Inverting Onto Functions and Polynomial Hierarchy

Harry Buhrman[1], Lance Fortnow[2], Michal Koucký[3], John D. Rogers[4], and Nikolay Vereshchagin[5]

[1] CWI, Amsterdam,
buhrman@cwi.nl
[2] University of Chicago,
fortnow@cs.uchicago.edu
[3] Institute of Mathematics of Czech Academy of Sciences,
mkoucky@cs.mcgill.ca
[4] DePaul University, Chicago,
jrogers@cs.depaul.edu
[5] Lomonosov Moscow State University,
ver@mech.math.msu.su

Abstract. The class **TFNP**, defined by Megiddo and Papadimitriou, consists of multivalued functions with values that are polynomially verifiable and guaranteed to exist. Do we have evidence that such functions are hard, for example, if **TFNP** is computable in polynomial-time does this imply the polynomial-time hierarchy collapses? By computing a multivalued function in deterministic polynomial-time we mean on every input producing one of the possible values of that function on that input.
We give a relativized negative answer to this question by exhibiting an oracle under which **TFNP** functions are easy to compute but the polynomial-time hierarchy is infinite. We also show that relative to this same oracle, $\mathbf{P} \neq \mathbf{UP}$ and $\mathbf{TFNP}^{\mathbf{NP}}$ functions are not computable in polynomial-time with an **NP** oracle.

1 Introduction

Megiddo and Papadimtriou [MP91] defined the class **TFNP**, the class of multivalued functions with values that are polynomially verifiable and guaranteed to exist. A function from **TFNP** is specified by a polynomial time computable binary relation $R(x, y)$ and a polynomial p such that for every string x there is a string y of length at most $p(|x|)$ such that $R(x, y)$ holds. It maps x to any y of length at most $p(|x|)$ such that $R(x, y)$. This class of functions includes Factoring, Nash Equilibrium, finding solutions of Sperner's Lemma, and finding collisions of hash functions.

Fenner, Fortnow, Naik and Rogers [FFNR03] consider the hypothesis, which they called "Q", that for every function in **TFNP** there is a polynomial-time procedure that will output a value of that function. That is, Proposition Q states that for every R and p defining a **TFNP**-function there is a polynomial time computable function f such that $R(x, f(x))$ holds for all x.

Fenner et. al. showed that Q is equivalent to a number of different hypotheses including

- Given an **NP** machine M with $L(M) = \Sigma^*$, there is a polynomial-time computable function f such that $f(x)$ is an accepting computation of $M(x)$.
- Given an honest onto polynomial-time computable function g there is a polynomial-time computable function f such that $g(f(x)) = x$. (A function $g(x)$ is called honest if there is a polynomial $p(n)$ such that $|x| \leq p(|g(x)|)$ for all x.)
- For all polynomial-time computable subsets S of **SAT** there is a polynomial-time computable function f such that for all ϕ in S, $f(\phi)$ is a satisfying assignment to ϕ.
- For all **NP** machines M such that $L(M) = \mathbf{SAT}$, there is a polynomial-time computable function f such that for every ϕ in **SAT** and accepting path c of $M(\phi)$, $f(\phi, c)$ is a satisfying assignment of ϕ.

Proposition Q implies that all disjoint **coNP**-sets are **P**-separable (which implies that $\mathbf{NP} \cap \mathbf{coNP} = \mathbf{P}$) and is implied by $\mathbf{P} = \mathbf{NP}$. Fenner et. al. ask whether we can draw any stronger complexity collapses from Q, in particular if Q implies that the polynomial-time hierarchy collapses. We give a relativized negative answer to this question by exhibiting an oracle relative to which Q holds and the polynomial-time hierarchy is infinite. Our proof uses a new "Kolmogorov generic" oracle.

Proposition Q naturally generalizes to other levels of the polynomial hierarchy. Namely, define the class $\mathbf{TF}\Sigma_k^p$ as follows. A $\mathbf{TF}\Sigma_k^p$-function is specified by a binary relation $R(x, y)$ computable in polynomial time with an oracle from Σ_{k-1}^p and a polynomial p such that for every string x there is a string y of length at most $p(|x|)$ such that $R(x, y)$ holds. For $k \geq 1$ we will label Σ_k^pQ the statement

> For every R and p defining a $\mathbf{TF}\Sigma_k^p$-function there is a function f computable in polynomial time with an oracle from Σ_{k-1}^p such that $R(x, f(x))$ holds for all x.

For $k = 1$ we obtain the class **TFNP** and Proposition Q.

Proposition $\mathbf{TF}\Sigma_k^p$ implies that $\mathbf{P}^{\Sigma_{k-1}^p} = \Sigma_k^p \cap \Pi_k^p$ and is implied by $\Sigma_{k-1}^p = \Sigma_k^p$. A natural question is whether, similar to the implication

$$\Sigma_{k-1}^p = \Sigma_k^p \Longrightarrow \Sigma_k^p = \Sigma_{k+1}^p,$$

Proposition $\mathbf{TF}\Sigma_k^p$ implies Proposition $\mathbf{TF}\Sigma_{k+1}^p$. We give a relativized negative answer to this question in the case $k = 1$: For any Kolmogorov generic G the Proposition $\mathbf{TF}\Sigma_2^{p,G}$ does not hold.

In addition we show that for any Kolmogorov generic G, $\mathbf{P}^G \neq \mathbf{UP}^G$.

2 Definitions and preliminaries

Let Σ denote the alphabet $\{0, 1\}$. The set of all finite-length binary strings is denoted Σ^*.

2.1 Complexity classes

Our model of computation is the oracle Turing machine, both deterministic (DTM) and nondeterministic (NTM). Unless otherwise noted, all machines in this paper run in polynomial time. We assume that the reader is familiar with the complexity classes **P**, **NP**, **UP**, **PSPACE**, Σ_k^p, and Π_k^p for $k \geq 0$. The class Δ_k^p is defined as $\mathbf{P}^{\Sigma_{k-1}^p}$, and $\mathbf{PH} = \bigcup_k \Sigma_k^p$ stands for the Polynomial hierarchy. The class $\mathbf{F}\Delta_k^p$ is defined as the class of all functions from Σ^* to Σ^* that are computable in polynomial time with an oracle from Σ_{k-1}^p.

We say that disjoint sets B and C are **P**-separable if there is a set $D \in \mathbf{P}$ such that $B \subseteq D$ and $C \subseteq \Sigma^* - D$.

Proposition Q and its generalizations Σ_k^pQ are defined in the Introduction. It is easy to see that Σ_k^pQ is equivalent to the following statement:

> For every nondeterministic polynomial-time Turing machine M with oracle from Σ_{k-1}^p that accepts Σ^*, there is a function f in $\mathbf{F}\Delta_k^p$ such that, for all x, $f(x)$ is an accepting computation of $M(x)$.

It is easy to see the following:

Proposition 1. *If $\Sigma_{k-1}^p = \Sigma_k^p$ then $\Sigma_k^p Q$ is true. If $\Sigma_k^p Q$ is true then $\Delta_k^p = \Sigma_k^p \cap \Pi_k^p$.*

2.2 Kolmogorov complexity and randomness

An excellent introduction to Kolmogorov complexity can be found in the textbook by Li and Vitányi [LV97]. We will state here the definitions and results relevant to our work. Roughly speaking, the Kolmogorov complexity of a binary string x is defined as the minimal length of a program that generates x; the conditional complexity $C(x|y)$ of x conditional to y is the minimal length of a program that produces x with y as input.

A *conditional description method* is a partial computable function Φ (that is, a Turing machine) mapping pairs of binary strings to binary strings. A string p is called a *description of x conditional to y* with respect to Φ if $\Phi(p, y) = x$. The complexity of x conditional to y with respect to Φ is defined as the minimal length of a description of x conditional to y with respect to Φ:

$$C_\Phi(x|y) = \min\{|p| \mid \Phi(p, y) = x\}.$$

A conditional description method Ψ is called *universal* if for all other conditional description methods Φ there is a constant k such that

$$C_\Psi(x|y) \leq C_\Phi(x|y) + k$$

for all x, y. The Solomonoff–Kolmogorov theorem [Sol64,Kol65] states that universal methods exist. We fix a universal Ψ and define *conditional Kolmogorov complexity* $C(x|y)$ as $C_\Psi(x|y)$. We call this Ψ the reference universal Turing machine. The (unconditional) Kolmogorov complexity $C(x)$ is defined as the

Kolmogorov complexity of x conditional to the empty string. Comparing the universal function Ψ with the function $\Phi(p, y) = \Psi(p, \text{empty string})$ we see that the conditional Kolmogorov complexity does not exceed the unconditional one:

$$C(x|y) \ \leq \ C(x) + O(1).$$

Comparing the universal function Ψ with the function $\Phi(p, y) = p$ we see that the Kolmogorov complexity does not exceed the length:

$$C(x) \leq |x| + k \tag{1}$$

for some k and all x. For most strings this inequality is close to an equality: the number of strings x of length n with

$$C(x) < n - m$$

is less than 2^{n-m}. Indeed, the total number of descriptions of length less than $n - m$ is equal to

$$1 + 2 + \cdots + 2^{n-m-1} = 2^{n-m} - 1.$$

In particular, for every n there is a string x of length n and complexity at least n. Such strings are called *incompressible, or random.*

Let $f(x, y)$ be a computable function mapping strings to strings. To describe the string $f(x, y)$ it is enough to concatenate x and y. Thus we obtain:

$$C(f(x, y)) \ \leq \ 2|x| + |y| + k. \tag{2}$$

where k depends on f and on the reference universal machine but not on x, y. The factor of 2 is needed, as we have to separate x from y. To this end we write the former in a self-delimiting form. As a self-delimiting encoding of a string u we take the string $\bar{u}$ obtained from u by doubling all its bits and appending the pattern 01. For instance, $\overline{001} = 00001101$. A similar inequality holds for computable functions of more than 2 strings:

$$C(f(x_1, x_2, \ldots, x_n)) \ \leq \ |x_1| + 2|x_2| + \cdots + 2|x_n| + O(1). \tag{3}$$

2.3 Generic oracles

An oracle is a subset of Σ^*. The oracles we use are *generic* oracles. What does this mean? To explain this we need to recall some definitions from category theory.

A *condition* on an oracle A is a finite set of *requirements* having the form $x \in A$ and $y \notin A$, where $x, y \in \Sigma^*$. We say that an oracle A *satisfies* a condition α if all the requirements in α are satisfied by A. Let U be a family of oracles and let U_α denote the set of all $A \in U$ satisfying α. An *interval* in U is a non-empty subset of U having the form U_α. A subset S of U is called *dense in* U if every non-empty interval I in U has a sub-interval J in U included in S. Countable intersections of dense sets are called *large* subsets of U.

Let P be a property of a set of oracles. We say that P *holds for a generic oracle* if the set P is large in the set of all oracles. We say that P *holds for a generic oracle in* U if the set $P \cap U$ is large in U. Assume that U has the following property: the intersection of every infinite descending chain

$$I_1 \supset I_2 \supset I_3 \supset \dots$$

of intervals in U is non-empty. Then by the usual diagonalization we can show that every large subset of U is non-empty. Using a metaphor, we can say that "generic oracles exist." Our usage of the term "generic" oracle is similar to the usage of the term "random" oracle. Indeed, we say that a property P holds for a random oracle if P is a measure 1 set.

Note that if a property P_0 holds for a generic oracle in U and P_1 holds for a generic oracle in U then so does $P_0 \cap P_1$. Therefore if we want to prove that Proposition Q holds but that the polynomial hierarchy is infinite relative to a generic oracle in U we can prove these things separately. The same applies to countable families of properties. If each P_i holds for a generic oracle in U then the property $\bigcap_i P_i$ also holds for a generic oracle in U. For example, if we want to prove that $\mathbf{P} \neq \mathbf{NP}$ relative to a generic oracle in U we can define a relativized language L that is in $\mathbf{NP}$ for generic oracle in U and then define a set of requirements R_i, where R_i is the statement "DTM number i does not accept L." Then it is enough to prove, for every i, that R_i holds relative to a generic oracle in U. To this end it suffices to prove that the set $U \cap R_i$ is dense in U: every interval I in U has a subinterval J in U such that $J \subset R_i$.

Good introductions and several applications of the approach we are using here may be found in the papers [BGS75,FFKL03,FR03,MV96].

Previous results

The method we are using was essentially designed in [BGS75]. Let U be a family of oracles. Fix a **PSPACE**-complete set H and consider oracles of the form

$$A \oplus H = \{0x \mid x \in A\} \cup \{1x \mid x \in H\},$$

where $A \in U$. We will denote the set of all such oracles by $U \oplus$PSPACE-complete. Consider "tower" numbers, that is, natural numbers $1, 2, 2^2, 2^{2^2}, \dots$. The next tower number from n is 2^n. Let

$$U_0 = \{A \mid A \cap \Sigma^n = \emptyset \text{ for all non-tower } n\}.$$

Most of the oracle constructions use ($U \oplus$ PSPACE-complete)-genericity where U is a subfamily of U_0. We survey three such particular families U, which are relevant to the results of this paper:

- Relative to a generic oracle in $U_0 \oplus$PSPACE-complete (called Cohen-generic in [FR03]) the following holds: $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$, Q is false [IN88], there are disjoint $\mathbf{P}$-inseparable $\mathbf{coNP}$-sets, $\mathbf{P} = \mathbf{UP}$ [FR03], and, moreover, the polynomial hierarchy $\mathbf{PH}$ is infinite (this is a direct corollary of lower bounds for constant depth circuits from [Hås89]).

– Let $U_1 = \{A \in U_0 \mid |A \cap \Sigma^n| \leq 1 \text{ for all tower } n\}$. Relative to a generic oracle in $U_1 \oplus$ PSPACE-complete (**UP**-generic, according to [FR03]), $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$, all disjoint **coNP**-sets are **P**-separable [BGS75], Q is true and $\mathbf{P} \neq \mathbf{UP} = \mathbf{PH}$ [FR03].
– Let $U_2 = \{A \in U_0 \mid |A \cap \Sigma^n| = 1 \text{ for all tower } n\}$. Relative to a generic oracle in $U_2 \oplus$ PSPACE-complete we have $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{coNP} = \mathbf{UP} = \mathbf{PH}$ [MV96].

2.4 Kolmogorov-generic oracles

No genericity notion known from the literature is suitable to construct an oracle such that Q is true but the **PH** is infinite. This is easily seen for the three genericity notions surveyed above. So we have to define a new one, which we call *Kolmogorov generic*.

For each n fix a binary string Z_n of length $n2^n$ that is incompressible, that is, $C(Z_n) \geq |Z_n| - O(1)$. Divide Z_n into substrings $z_1, \ldots, z_{2^n}$, each of length n. Let Y_n be the set $\{\langle i, z_i\rangle | i \in \{0,1\}^n\}$. (Here we identify i with the integer binary represented by i. The pair $\langle u, v\rangle$ is encoded be the string $\bar{u}v$, where $\bar{u}$ stands for the self-delimiting encoding of u defined in Section 2.2.) Let U be the set of all subsets of $\bigcup_n Y_n$ where the union is over all tower n.

When proving that a certain property holds for a Kolmogorov generic oracle $A = G \oplus H \in U \oplus$ PSPACE-complete we use the fact that every two different lengths of strings in G are exponentially far apart. When discussing a particular polynomial-time computation, we only have to worry about strings at exactly one length in the oracle. Longer strings cannot be queried by the computation and so cannot affect it. Shorter strings can all be queried and found by the computation.

3 Results

Theorem 1. *Relative to a generic oracle in $U \oplus$PSPACE-complete (a Kolmogorov-generic oracle), Proposition Q is true.*

Proof. We first assume that $\mathbf{P} = \mathbf{PSPACE}$ and prove that Proposition Q is true under a generic oracle $G \in U$.

As explained above it suffices to show that for every polynomial-time oracle NTM M and relative to a Kolmogorov-generic oracle,

$$\begin{aligned}&\text{If } M \text{ accepts } \Sigma^* \text{ then there is a polynomial time machine}\\ &\text{finding for each input an accepting computation of } M.\end{aligned} \tag{4}$$

Fix M. Without loss of generality, M on an input x runs in time $|x|^k + k$, for some constant k independent of its oracle. Indeed, for each oracle nondeterministic Turing machine M (not necessarily polynomial time) and natural k we can construct an NTM that acts as M supplied with a clock that prevents it to

run more than in $|x|^k + k$ steps. If M^A runs in polynomial time then for some k the machine M^A supplied with the clock $|x|^k + k$ is equivalent to M^A.

We will show that the set of oracles satisfying (4) is dense. Let $I = U_\alpha$ be an interval in U. We need to construct a sub-interval J of I such that (4) is true for all $G \in J$. Consider two cases.

Case 1. There is a sub-interval of I such that for all A in that sub-interval, M^A does not accept Σ^*. Then let J be equal to that sub-interval of I.

Case 2. There is no such sub-interval. Consider the following polynomial-time deterministic algorithm A that, given an input x of length at least two, finds an accepting path of the computation $M^G(x)$. Let n be the largest tower number smaller or equal to $4|x|^{2k}$. The algorithm A will try to collect enough information about the oracle G so that it could find an accepting path of $M^G(x)$. The algorithm A starts by asking the value of G on all the strings in Y_i for $i \leq \log n$. This can be done in time polynomial in $|x|$.

After that it will iteratively build a set Q of strings from $G \cap Y_n$ starting from an empty set Q. Using the assumption that $\mathbf{P} = \mathbf{PSPACE}$ and the information about G collected so far, the algorithm finds the lexicographically first accepting path of M^G on x under the assumption that $G \cap \{\langle i, u\rangle | i, u \in \{0,1\}^n\} = Q$. (Note, M on x cannot query any string in Y_m, for $m > n$.) Such path does exist, as otherwise, the sub-interval J of I, consisting of all G' with $G' \cap \{\langle i, u\rangle | i, u \in \{0,1\}^n\} = Q$ and $G' \cap Y_i = G \cap Y_i$ for all $i \leq \log n$ would qualify case (1).

If this path is indeed an accepting path of the computation $M^G(x)$, A is done. If not then there is a string $w \in (G \cap \{\langle i, u\rangle | i, u \in \{0,1\}^n\}) \setminus Q$ that is queried along this path. Clearly such w is from Y_n. The algorithm picks the first such w, adds it to the set Q and iterates the process. Clearly, A eventually finds a correct accepting path of $M^G(x)$. We claim that A will find it within polynomially many iterations.

Observe, given M, x, $G \cap Y_{\leq \log n}$ each string in Q can be described by $k \log |x|$ bits by its order number among the queries of M on x on the accepting path described above. The set $G \cap Y_{\leq \log n}$ has at most $n + \log n + \log\log n + \ldots$ strings, each of length at most $\log n$. Thus $G \cap Y_{\leq \log n}$ can be described in at most $O(n \log n)$ bits. Hence if Q reaches size ℓ, we can describe Q by $\ell k \log |x| + O(n \log n) + 2|x| + O(1)$ bits (by Equation (3)).

Recall that all of the strings in Y_n are derived from Z_n. Because of the way Y_n is defined any set A of ℓ strings from Y_n has Kolmogorov complexity at least $\ell n/2 - O(1)$.

Indeed, each element of Y_n is a pair $\langle i, y\rangle$. Let p denote the concatenation of all y's from all pairs $\langle i, y\rangle$ outside A arranged according to the order on i's. The length of p is $n(2^n - \ell)$. The initial string Z_n can be obtained from p by inserting the second components of pairs from A, their first components specify the places where to insert. Thus given p and the shortest description q of A we can find Z_n, and Equation (2) implies

$$n2^n - O(1) \leq C(Z_n) \leq |p| + 2|q| + O(1) = n(2^n - \ell) + 2C(A) + O(1).$$

Since $2+2k\log|x| < n \leq 4|x|^{2k}$, the Kolmogorov complexity of ℓ strings from Y_n is at least $\ell k \log|x| + \ell - O(1)$. Thus Q cannot grow bigger than $O(n\log n) + 2|x| = O(|x|^{2k}\log|x|)$.

We can remove the hypothesis that $\mathbf{P} = \mathbf{PSPACE}$ by first relativizing to an oracle making $\mathbf{P} = \mathbf{PSPACE}$. It is known that relative to every **PSPACE**-complete set H we have $\mathbf{P} = \mathbf{PSPACE}$. Thus, relative to H, Q-property holds relative to a generic oracle in U. As H is computable, the Kolmogorov complexity relativized by H coincides with the unrelativized Kolmogorov complexity (up to an additive constant), and relativization does not change the notion of Kolmogorov genericity. In other words, Property Q holds relative a generic oracle in $U \oplus$ PSPACE-complete. □

Theorem 2. *Relative to a generic oracle in $U \oplus$PSPACE-complete (a Kolmogorov-generic oracle), for all $k \geq 0$ we have $\Sigma_k^p \neq \Sigma_{k+1}^p$.*

Proof. Meyer and Stockmeyer [MS72] show that if $\Sigma_k^p = \Sigma_{k+1}^p$ then $\Sigma_k^p = \Sigma_j^p$ for all $j \geq k$. So it is sufficient for us to show that $\Sigma_{k-2}^p \neq \Sigma_{k+1}^p$ for all $k \geq 3$ relative to a Kolmogorov generic oracle $G \oplus H$, where H is an arbitrary set.

We use the Sipser [Sip83] functions as defined by Håstad [Hås89]. The function f_k^m is represented by a depth k alternating circuit tree with an OR gate at the top with fan-in $\sqrt{m/\log m}$, bottom fan-in $\sqrt{km\log m/2}$ and all other fan-ins are m. Each variable occurs just once at each leaf.

Theorem 3 (Håstad). *Depth $k-1$ circuits computing f_k^m are of size at least $2^{\Omega(\sqrt{m/(k\log m)})}$.*

Pick a tower n. Set $m = 2^{n/k}$. The number of variables of f_k^m is $m^{k-1}\sqrt{k/2} < 2^n$ for large n. For each of the variables of this formula assign a unique $i \in \{0,1\}^n$ so we can in polynomial-time find i from the variable and vice-versa.

Now consider the language $L_k(G)$ such that input 1^n is in $L(G)$ if f_k^m is true when we set the variables corresponding to i to one if $\langle i, z_i\rangle$ is in G and zero otherwise.

We will show relative to a Kolmogorov generic oracle $G \oplus H$, $L_k(G) \in \Sigma_{k+1}^{p,G\oplus H} - \Sigma_{k-2}^{p,G\oplus H}$.

First notice that $L_k(G) \in \Sigma_{k+1}^{p,G\oplus H}$ for all $G \in U$: Consider an alternating Turing machine that uses k-alternations to simulate the circuit. To determine whether a variable corresponding to i is true the machine makes the **NP** query "is there a z such that $\langle i, z\rangle$ is in G." This gives us a $\Sigma_k^{\mathbf{NP},G} = \Sigma_{k+1}^{p,G}$ machine accepting $L_k(G)$.

Let M be an alternating Σ_{k-2}^p oracle Turing machine that runs in time n^j. Let $I = U_\alpha$ be an interval in U. We need to construct a subinterval J of I such that $M^{G\oplus H}$ does not accept $L(G)$ for all $G \in J$. Along the lines of Sipser [Sip83] we can convert the computation to a circuit of depth $k-1$ and size $2^{O(n^j)}$ whose input variables correspond to queries to G. Hardwire the queries not of the form $\langle i, z_i\rangle$ to one if they belong to H and zero otherwise to obtain a circuit whose variables are the same as those in f_k^m in the definition of $L_k(G)$ on 1^n. By

Theorem 3 for sufficiently large n this circuit cannot compute f_k^m so there must be some setting of the variables where the circuit and f_k^m have different outputs. Add to the condition α the requirement $\langle i, z_i \rangle \in G$ if variable i is assigned 1 in this setting and the requirement $\langle i, z_i \rangle \notin G$ otherwise. For all $G \in U$ satisfying the resulting condition, $M^{G \oplus H}(1^n)$ accepts iff 1^n is not in $L(G)$. □

We can also show that one-way functions exist relative to G.

Theorem 4. *Relative to a Kolmogorov generic oracle $G \oplus H$, $\mathbf{P} \neq \mathbf{UP}$.*

Proof. Define the relativized language L^X as $\{\langle i, 0^n \rangle : (\exists z)|z| = n \ \& \ \langle i, z \rangle \in X\}$. For a string z of length n, there is at most one string of the form $\langle i, z \rangle$ in G so the language is in $\mathbf{UP}^G$. A simple argument demonstrates that L^G is not in $\mathbf{P}^{G \oplus H}$. □

Can the proof that Q holds relative to a Kolmogorov-generic be lifted to show that Σ_k^pQ holds and we get the collapse of the Δ_k^p and $\Sigma_k^p \cap \Pi_k^p$? The answer is no for $k = 2$ and the proof of this shows that this is true for a broad class of finite extension oracles.

To show that Σ_2^pQ fails relative to a Kolmogorov-generic oracle G, let f be a function from Σ^* to Σ^* where for every x of length n

$$f(x) = y_1 \ldots y_{n-1}$$

and

$$y_j = 1 \iff (\exists u, z)\ |u| = n,\ |z| = 2n + \log n,\ \langle xju, z \rangle \in G.$$

No matter what strings are in G, the pigeonhole principle tells us that, for all n, there will always be a *collision*, that is, two different strings x_1 and x_2 of length n such that $f(x_1) = f(x_2)$.

Let M be a $\Sigma_2^{p,G}$ machine that on any input of length n guesses two different strings of length n in its existential step and then accepts iff those strings collide on f. It is clear from the definition of f that M can find these collisions and that it accepts Σ^*. A $\mathbf{P}^{\mathbf{NP}^G}$ machine that finds an accepting path of M could be modified to output the two colliding strings on that path so, without loss of generality, we will assume it does just that.

Theorem 5. *Relative to a Kolmogorov generic oracle $G \oplus H$, no $\mathbf{P}^{\mathbf{NP}}$ machine can find an accepting path of the computation $M(x)$ for every x.*

Proof. Let $\langle R, N \rangle$ be an arbitrary pair consisting of an oracle polynomial time DTM R and an oracle polynomial time NTM N. We will show that the set of all oracles G such that R with oracle $N^{G \oplus H}$ does not find any collision of f^G is dense in U.

Without loss of generality we can assume that there are polynomial upper bounds of the running time of R and N that do not depend on their oracles. Let p_R and p_N stand for those polynomials, respectively.

Let I_α be an interval in U. We will show that for some n there is an interval $I_\beta \subset I_\alpha$ such that for all $G \in I_\beta$, $R^{N^{G \oplus H}}(0^n)$ does not find two strings that collide on f^G.

Pick a large n that is bigger than the maximal length of strings in the domain of α and such that $2n + \log n$ is a tower number. (We call the set of all y such that α contains a condition $y \in G$ or $y \notin G$ the domain of α and use the notation $\operatorname{dom}\alpha$.)

Note that the outcome of $R^{N^{G\oplus H}}$ on input 0^n depends only on membership in G of strings of length at most $p_N(p_R(n))$. First we add to α the requirements $y \notin G$ for all strings $y \notin Y_{2n+\log n} \cup \operatorname{dom}\alpha$ of length at most $p_N(p_R(n))$ and denote by β_0 the resulting condition. The condition β is obtained from β_0 in at most $p_R(n)$ iterations. In ith iteration we define a condition β_i obtained from β_{i-1} by adding some requirements of the form $y \in G$ and $y \notin G$ for $y \in Y_{2n+\log n}$.

Let us explain this in more detail. For $x \in \Sigma^n$ and $j = 1, \dots, n-1$ let

$$B_{jx} = \{\langle xju, z_{xju}\rangle \mid \ u \in \Sigma^n\}.$$

We call the set $B_x = \bigcup_{j=1}^{n-1} B_{jx}$ the *bag* corresponding to x. The value $f^G(x)$ depends only on $B_x \cap G$. More specifically, jth bit of $f^G(x)$ is 0 if the set $B_{jx} \cap G$ is empty.

On each iteration we choose a set $D \subset \Sigma^n$ of cardinality at most $p_N(p_R(n))$ and *set* oracle's value on the set $\bigcup_{x\in D} B_x$. This means that for every y in this set we include in β_i either the condition $y \notin G$, or the condition $y \in G$. The notation D_i will refer to the set of all strings x such that oracle's value is set on B_x during iterations $s = 1, \dots, i$. We will keep the following statement invariant: f^G is injective on D_i for all $G \in I_{\beta_i}$.

Additionally, on ith iteration we choose the *desired* answer a_i of $N^{G\oplus H}$ to the i-th query to $N^{G\oplus H}$ in the run of R on input 0^n.

On ith iteration we run R on input 0^n assuming the answers $a_1, \dots, a_{i-1}$ to oracle queries until R makes ith query q_i to the oracle or outputs a result. If the first option happens, we choose the desired answer of $N^{G\oplus H}$ on q_i as follows.

Assume that $G \in I_{\beta_{i-1}}$ and C is an accepting computation of $N^{G\oplus H}$ on input q_i. We say that $\langle G, C\rangle$ is a *good pair* if the following holds. Let D be the set of all $x \in \Sigma^n$ such that computation C queries a string in the bag of x. The pair $\langle G, C\rangle$ is good if f^G is injective on the set $D \cup D_{i-1}$.

Assume first that there is a good pair $\langle G, C\rangle$. In this case we pick a good pair $\langle \tilde{G}, \tilde{C}\rangle$, define D as explained above and choose YES as the desired answer to ith query. The condition β_i is obtained from β_{i-1} by adding the requirements $y \in G$ for all $y \in \bigcup_{x\in D} B_x \cap \tilde{G}$ and the requirements $y \notin G$ for all $y \in \bigcup_{x\in D} B_x \setminus \tilde{G}$. Note that $N^{G\oplus H}(q_i) = 1$ for all $G \in I_{\beta_i}$.

If there is no good pair $\langle G, C\rangle$ then we choose NO as the desired answer to ith query and set $\beta_i = \beta_{i-1}$, $D_i = D_{i-1}$.

On some iteration $k \le p_R(n)$, R makes no new queries and outputs two strings x_1 and x_2, where f presumably collides. At this point we set oracle's value on all remaining strings in $Y_{2n+\log n}$ as follows. Pick any oracle $\tilde{G} \in I_{\beta_{k-1}}$ such that $f^{\tilde{G}}$ is injective on the set $D_k = D_{k-1} \cup \{x_1, x_2\}$ and such that for all $x \in \Sigma^n \setminus D_k$, $f^{\tilde{G}}(x) = 00\dots0$. If n is large enough then there is such $\tilde{G}$. Indeed, the length of q_i is at most $p_R(n)$ and thus every computation of $N^{\tilde{G}\oplus H}$

on input q_i runs in time $p_N(p_R(n))$. Hence $|D_k|$ is bounded by the polynomial $p_R(n)p_N(p_R(n)) + 2$. If 2^{n-1} is bigger than this bound then there are enough strings in the range of f to avoid collision in D_k.

We let β be the condition containing the requirements $y \in G$ for all $y \in \tilde{G}$ of length at most $p_N(p_R(n))$ and the requirements $y \notin G$ for all $y \notin \tilde{G}$ of length at most $p_N(p_R(n))$.

We claim that for all $G \in I_\beta$, $R^{N^{G\oplus H}}$ on 0^n computes the way how we determined. Indeed, if R computes differently for some $G \in I_\beta$ then there must be a query answered in the opposite way than we desire. Let q_i be the first such query. Note that q_i coincides with the ith query in our construction, as all the previous queries are answered by $N^{G\oplus H}$ as we desire. If we have chosen YES as the desired answer to ith query then by construction $N^{G\oplus H}(q_i) = 1$ and thus the desired answer is correct. Therefore this may happen only if we have chosen NO as the ith answer and $N^{G\oplus H}(q_i) = 1$.

By way of contradiction, assume that this is the case. Pick then an accepting computation C of $N^{G\oplus H}$ on q_i. We will show that there is $G' \in I_{\beta_{i-1}}$ such that $\langle G', C\rangle$ is a good pair. Let D be the set of all $x \in \Sigma^n$ such that computation C queries a string in the bag of x. Note that by construction f^G is injective on D_k. (However, f^G may be not injective on $D_{i-1} \cup D$ thus $\langle G, C\rangle$ may be not a good pair.)

We will add to G some strings from $\bigcup_{x\in D\setminus D_k} B_x$ so that for the resulting oracle G' the pair $\langle G', C\rangle$ is good. We may assume that 2^n, the cardinality of every set B_{jx}, is greater than the number of queries along C. For every $x \in D\setminus D_k$ and every j we can change jth bit of $f^G(x)$ to 1 by adding to G a non-queried string from B_{jx}. All of the 2^{n-1} values in the range of f can be obtained in this way. Thus we can change $f^G(x)$ for all $x \in D \setminus D_k$ one by one so that for the resulting oracle G', C is an accepting computation and $f^{G'}(x)$ is injective on $D \cup D_k$ and hence on $D \cup D_{i-1}$. □

4 Conclusion and open problems

Is there an oracle relative to which the polynomial-time hierarchy is proper and $\Sigma_k^p\mathrm{Q}$ is true for all k? As a corollary we would get a relativized world where the hierarchy is proper and $\Delta_k^p = \Sigma_k^p \cap \Pi_k^p$. The second statement remains open even relative to Kolmogorov generics and, if true, would give a relativized version of the polynomial-time hierarchy that acts like the arithmetic hierarchy.

Acknowledgments

We thank Steve Fenner and Marcus Schaefer for helpful discussions. The work of N. Vereshchagin was in part supported by RFBR grant 06-01-00122 and the work of M. Koucký was supported in part by grant GA ČR 201/07/P276, project No. 1M0021620808 of MŠMT ČR and Institutional Research Plan No. AV0Z10190503.

References

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

[FFKL03] Lance Fortnow, Stephen Fenner, Stuart Kurtz, and Lide Li. An oracle builder's toolkit. *Information and Computation*, 182:95–136, 2003.

[FFNR03] Lance Fortnow, Stephen Fenner, Ashish Naik, and John Rogers. Inverting onto functions. *Information and Computation*, 186:90–103, 2003.

[FR03] Lance Fortnow and John Rogers. Separability and one-way functions. *Computational Complexity*, 11:137–157, 2003.

[Hås89] J. Håstad. Almost optimal lower bounds for small depth circuits. *Advances in Computing Research*, 5:143–170, 1989.

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

[Kol65] A.N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

[LV97] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Graduate Texts in Computer Science. Springer, New York, second edition, 1997.

[MP91] N. Megiddo and C. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

[MS72] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, pages 125–129. IEEE, New York, 1972.

[MV96] An. Muchnik and N. Vereshchagin. A general method to construct oracles realizing given relationships between complexity classes. *Theoretical Computer Science*, 157:227–258, 1996.

[Sip83] M. Sipser. Borel sets and circuit complexity. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 61–69. ACM, New York, 1983.

[Sol64] R.J. Solomonoff. A formal theory of inductive inference, part 1 and part 2. *Information and Control*, 7:1–22, 224–254, 1964.