Query Order and NP-Completeness [1]

Jack J. Dai
Department of Mathematics
Iowa State University
Ames, Iowa 50011
U.S.A.

Jack H. Lutz
Department of Computer Science
Iowa State University
Ames, Iowa 50011
U.S.A.

Abstract

The effect of query order on NP-completeness is investigated. A sequence $\vec{D} = (D_1, \ldots, D_k)$ of decision problems is defined to be *sequentially complete* for NP if each $D_i \in \mathrm{NP}$ and every problem in NP can be decided in polynomial time with one query to each of $D_1, \ldots, D_k$ *in this order.* It is shown that, if NP contains a language that is p-generic in the sense of Ambos-Spies, Fleischhack, and Huwig [3], then for every integer $k \geq 2$, there is a sequence $\vec{D} = (D_1, \ldots, D_k)$ such that $\vec{D}$ is sequentially complete for NP, but no nontrivial permutation $(D_{i_1}, \ldots, D_{i_k})$ of $\vec{D}$ is sequentially complete for NP. It follows that such a sequence $\vec{D}$ exists if there is any strongly positive, p-computable probability measure ν such that $\nu_{\mathrm{p}}(\mathrm{NP}) \neq 0$.

1 Introduction

The success or efficiency of a computation sometimes depends—and sometimes does not depend—upon the order in which it is allowed access to the various pieces of information that it may require. Although this truism is an important reality in many areas of computing, including information security, machine learning, game playing, and intelligent planning, it has only very recently come under complexity-theoretic scrutiny.

The natural way to begin investigation of this phenomenon is to focus on the effect of *query order* on the efficiency of oracle computations. Indeed, most of the work to date has focused on situations in which a polynomial-time computation has oracle access to complete problems for several different complexity classes. The typical sort of question is then whether various constraints on the order in which these complete problems must be queried confer differing amounts of power on the underlying computations. The results of such investigations, some of which are at first surprising, have recently been surveyed by Hemaspaandra, Hemaspaandra, and Hempel [9].

[1]This research was supported in part by National Science Foundation Grant CCR-9610461.

In this paper, we examine a related, but somewhat different aspect of query order, namely its effect on NP-completeness. To be more precise, we say that a sequence $\vec{D} = (D_1, \ldots, D_k)$ of decision problems is *sequentially complete* for NP if each $D_i \in$ NP and every problem in NP can be decided in polynomial time with one query to each of $D_1, \ldots, D_k$ *in this order*. The question we ask is whether, for $k \geq 2$, there is a sequence $\vec{D} = (D_1, \ldots, D_k)$ such that $\vec{D}$ is sequentially complete for NP, but no nontrivial permutation $(D_{i_1}, \ldots, D_{i_k})$ of $\vec{D}$ is sequentially complete for NP. In this case we say that $\vec{D}$ is *strictly sequentially complete* for NP.

It is immediately clear that the existence of a sequence $\vec{D}$ that is strictly sequentially complete for NP implies that P $\neq$ NP. The converse may also be true, but appears to be difficult to prove. We thus turn to a stronger hypothesis than P $\neq$ NP.

A natural hypothesis to consider is the hypothesis that NP does not have p-measure 0 (briefly, $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$). This hypothesis, which was proposed by the second author, implies that P $\neq$ NP and has been proven to have many plausible consequences not known to be provable from P $\neq$ NP. The recent surveys [5, 11, 8] discuss a number of such consequences.

As will be seen below, the existence of a sequence $\vec{D}$ that is strictly sequentially complete for NP does indeed follow from the $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$ hypothesis. However, our main theorem establishes more, namely, that if NP contains a language that is p-generic (in the sense of Ambos-Spies, Fleischhack, and Huwig [3, 4]), then there is a sequence $\vec{D}$ that is strictly sequentially complete for NP.

This notion of p-genericity (which is defined precisely in Section 2 below) has recently been the subject of several investigations, most of which have been discussed in the paper [1] and/or the survey [5]. The hypothesis that NP contains a p-generic language is known to imply the P $\neq$ NP conjecture, and to be implied by the $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$ hypothesis [6, 5], and neither of the converse implications has been proven. Many (but not all) of the consequences of $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$ have also shown to be consequences of NP containing a p-generic language. One concrete advantage of using the latter hypothesis (when possible) is that it weakens the $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$ hypothesis from the uniform probability measure μ to a wide variety of probability measures ν. Specifically, Lorentz and Lutz [10] have shown that, if there is *any* strongly positive, p-computable probability measure ν such that $\nu_{\mathrm{p}}(\mathrm{NP}) \neq 0$, then NP contains a p-generic language. Thus, by our main theorem, the existence of such a probability measure ν implies the existence of a sequence $\vec{D}$ that is strictly sequentially complete for NP. (Note: Breutzmann and Lutz [7] have recently shown that $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$ is equivalent to $\nu_{\mathrm{p}}(\mathrm{NP}) \neq 0$ whenever ν is a strongly positive, P-computable, coin-toss probability measure, but this is a much stronger restriction on ν than the result of [10].)

The proof of our main result uses some ideas of Lutz and Mayordomo [12] and Ambos-Spies and Bentzien [2] but is somewhat more involved.

2 Preliminaries

In this paper, all *languages* are sets of binary strings, i.e., sets $A \subseteq \{0,1\}^*$. We identify each language A with its characteristic sequence $\chi_A \in \{0,1\}^\infty$ defined by

$$\chi_A = A(s_o)A(s_1)\cdots A(s_n)\cdots$$

where $s_0 = \lambda, s_1 = 0, s_2 = 1, \ldots$ is the standard enumeration of $\{0,1\}^*$, and

$$A(s_n) = \begin{cases} 1 & \text{if } s_n \in A \\ 0 & \text{otherwise.} \end{cases}$$

When $x = s_n$, $y = s_m$, we say "$x < y$" if and only if $n < m$. If $A = a_0a_1\cdots \in \{0,1\}^\infty$ and $x = s_n$, then we write $A\restriction x = a_0a_1\cdots a_{n-1}$. For $A, B \subseteq \{0,1\}^*$ and $n \in \mathbb{N}$, we define

$$A \oplus B = \{x1 \mid x \in A\} \cup \{y0 \mid y \in B\}$$

and

$$A_{(n)} = \{x \in \{0,1\}^* \mid x10^n \in A\}.$$

We briefly review the notion of p-genericity, introduced by Ambos-Spies, Fleischhack, and Huwig [3, 4]. We refer the reader to [1] or [5] for more detailed discussions.

Definition.

1. An *extension function* is a partial function f such that dom $f \subseteq \{0,1\}^*$ and, for all $A\restriction x \in$ dom f, $f(A\restriction x)$ is of the form

   $$f(A\restriction x) = ((x_0, i_0), \ldots, (x_m, i_m)),$$

   where $m \in \mathbb{N}$, $x_0, \ldots, x_m \in \{0,1\}^*$, $x < x_0 < x_1 \cdots < x_m$, and $i_0, \ldots, i_m \in \{0,1\}$.

2. A $t(n)$-*extension function* is an extension function that is computable in $O(t(n))$ time.

3. An extension function is *simple* if $f(A\restriction x) \in \{(x,0),(x,1)\}$ for all $A\restriction x \in$ dom f.

4. If f is an extension function and k is a positive integer, then f is *k-bounded* if, for all $A{\upharpoonright}x \in \text{dom } f$,
$$f(A{\upharpoonright}x) = ((x_0, i_0), \ldots, (x_l, i_l)),$$
for some $l < k$.

Definition. Let f be an extension function, and let $A \subseteq \{0,1\}^*$.

1. f is *dense along* A if there are infinitely many $x \in \{0,1\}^*$ for which $f(A{\upharpoonright}x)$ is defined.
2. A *meets* f *at* $x \in \{0,1\}^*$ if $f(A{\upharpoonright}x)$ is defined, say $f(A{\upharpoonright}x) = ((x_0, i_0), \ldots, (x_m, i_m))$ and, for all $0 \leq l \leq m$, $A(x_l) = i_l$.
3. A *meets* f if A meets f at some string x. Otherwise, A *avoids* f.

Definition. Let $A \subseteq \{0,1\}^*$.

1. A is $t(n)$-*generic* if A meets every simple $t(n)$-extension function that is dense along A.
2. A is p-*generic* if A is n^c-generic for every positive integer c.

The proof of our main result uses the following lemma.

Lemma 2.1 (Ambos-Spies [1]). If A is p-generic, then for all positive integers k and c, A meets every k-bounded n^c-extension function that is dense along A.

3 Sequential Reductions

In this section we introduce the notion of strictly sequential completeness for NP. Our development uses the following special type of polynomial-time reduction.

Definition. Let $A, B_1, \ldots, B_k \subseteq \{0,1\}^*$, and write $\vec{B} = (B_1, \ldots, B_k)$.

1. A *sequential, polynomial-time reduction* (briefly, a *$\leq_{\mathrm{S}}^{\mathrm{P}}$-reduction*) of A to $\vec{B}$ is a polynomial time-bounded k-oracle Turing machine M with the following two properties.

   (a) $A = L(M^{\vec{B}})$.

   (b) For all $x \in \{0,1\}^*$, the computation of $M^{\vec{B}}(x)$ makes exactly one query to each of the oracles $B_1, \ldots, B_k$, *with the queries occurring in this order*.

2. A is *sequentially polynomial-time reducible* (briefly, *$\leq_{\mathrm{S}}^{\mathrm{P}}$-reducible*) to $\vec{B}$, and we write $A \leq_{\mathrm{S}}^{\mathrm{P}} \vec{B}$, if there exists a $\leq_{\mathrm{S}}^{\mathrm{P}}$-reduction of A to $\vec{B}$.

We now use sequential reductions to define sequential completeness and strictly sequential completeness.

<u>Definition.</u> Let $\mathcal{C}$ be a class of languages, and let $\vec{B} = (B_1, \ldots, B_k)$ be a sequence of languages.

1. $\vec{B}$ is *sequentially hard* (briefly, *$\leq_{\mathrm{S}}^{\mathrm{P}}$-hard*) for $\mathcal{C}$ if, for all $A \in \mathcal{C}$, $A \leq_{\mathrm{S}}^{\mathrm{P}} \vec{B}$.

2. $\vec{B}$ is *sequentially complete* (briefly, *$\leq_{\mathrm{S}}^{\mathrm{P}}$-complete*) for $\mathcal{C}$ if $B_1, \ldots, B_k \in \mathcal{C}$ and $\vec{B}$ is $\leq_{\mathrm{S}}^{\mathrm{P}}$-hard for $\mathcal{C}$.

3. $\vec{B}$ is *strictly sequentially hard* (briefly, *$\leq_{\mathrm{SS}}^{\mathrm{P}}$-hard*) for $\mathcal{C}$ if $k \geq 2$, $\vec{B}$ is $\leq_{\mathrm{S}}^{\mathrm{P}}$-hard for $\mathcal{C}$, and no nontrivial permutation $(B_{i_1}, \ldots, B_{i_k})$ of $\vec{B}$ is $\leq_{\mathrm{S}}^{\mathrm{P}}$-hard for $\mathcal{C}$.

4. $\vec{B}$ is *strictly sequentially complete* (briefly, *$\leq_{\mathrm{SS}}^{\mathrm{P}}$-complete*) for $\mathcal{C}$ if $B_1, \ldots, B_k \in \mathcal{C}$ and $\vec{B}$ is $\leq_{\mathrm{SS}}^{\mathrm{P}}$-hard for $\mathcal{C}$.

We now make two observations regarding these definitions. The first is obvious.

<u>Observation 3.1.</u> If $\vec{B}$ is $\leq_{\mathrm{SS}}^{\mathrm{P}}$-complete for $\mathcal{C}$, then $\mathcal{C} \not\subseteq \mathrm{P}$.

In particular, Observation 3.1 implies that the existence of $\leq_{\mathrm{SS}}^{\mathrm{P}}$-complete languages for NP implies that P $\neq$ NP.

Our second observation is also easily verified.

Our second observation is an obvious characterization of $\leq_{\mathrm{S}}^{\mathrm{P}}$-reducibility that is used in the proof of our main result.

Observation 3.2. Let $A \subseteq \{0,1\}^*$, and let $\vec{B} = (B_1, \ldots B_k)$ be a sequence of languages. Then $A \leq_{\mathrm{S}}^{\mathrm{P}} \vec{B}$ if and only if there exist a family $\mathcal{F} = \{f_\alpha \mid \alpha \in \{0,1\}^{\leq k-1}\}$ of polynomial time-computable functions $f_\alpha : \{0,1\}^* \longrightarrow \{0,1\}^*$ and a polynomial time-computable function $h : \{0,1\}^* \times \{0,1\}^k \longrightarrow \{0,1\}$ such that, for all $x \in \{0,1\}^*$, if we define $b_1, \ldots, b_k \in \{0,1\}$ by the recursion $b_i = B_i(f_{b_1 \cdots b_{i-1}}(x))$, then

$$A(x) = h(x, b_1 \cdots b_k).$$

(Intuitively, if M is the $\leq_{\mathrm{S}}^{\mathrm{P}}$-reduction of A to $\vec{B}$, then $f_{b_1 \cdots b_{i-1}}(x)$ is the i^{th} query of $M^{\vec{B}}(x)$ when the first $i-1$ query answers are $b_1, \ldots, b_{i-1}$; and $h(x, b_1 \cdots b_k)$ is the final accept/reject decision of $M^{\vec{B}}(x)$.) In this case, we say that $A \leq_{\mathrm{S}}^{\mathrm{P}} \vec{B}$ *via* $(\mathcal{F}, h)$.

4 Main Result

In this section, we prove our main result, which says that if NP contains a p-generic language, then for every $k \geq 2$ there is a sequence $\vec{D} = (D_1, \ldots, D_k)$ that is $\leq_{\mathrm{SS}}^{\mathrm{P}}$-complete for NP. The following construction will be used.

Construction 4.1. Given $A, S \subseteq \{0,1\}^*$ and an integer $k \geq 2$, define the sequence $\vec{D}(A, S, k) = (D_1, \ldots, D_k)$ as follows.

(i) $D_1 = A_{(4k-4)}$.

(ii) For $1 < i < k$,

$$D_i = [A_{(4k-4i+4)} \cap A_{(4k-4i)}] \oplus [A_{(4k-4i+4)} \cup A_{(4k-4i)}].$$

(iii) $D_k = [A_{(4)} \cap S] \oplus [A_{(4)} \cup S]$.

Most of our work is done by the following lemma.

Lemma 4.2 (Main Lemma). Let $A \subseteq \{0,1\}^*$, $S \in \mathrm{DTIME}(2^n)$, and let $\vec{D} = \vec{D}(A, S, k) = (D_1, \ldots, D_k)$. If A is p-generic and $\vec{D}' = (D_{i_1}, \ldots, D_{i_k})$ is a nontrivial permutation of $\vec{D}$, then $A \not\leq_{\mathrm{S}}^{\mathrm{P}} \vec{D}'$.

Proof. Let $A, S, \vec{D}$, and $\vec{D}'$ be as given, and assume that $A \leq_{\mathrm{S}}^{\mathrm{P}} \vec{D}'$. It suffices to show that A is not p-generic.

Since $A \leq_{\mathrm{S}}^{\mathrm{P}} \vec{D}'$, there exist $\mathcal{F}$ and h as in Observation 3.2 such that $A \leq_{\mathrm{S}}^{\mathrm{P}} \vec{D}'$ via $(\mathcal{F}, h)$. Our argument proceeds as follows. When we use $(\mathcal{F}, h)$ to answer the question "$x \in A$?", if at some stage we encounter a question "$y \in A_{(4l)}$?", then when $y10^{4l} > x$, we make a guess about the answer of this question. Using these guesses, we can obtain the value of $A(x)$. Then the opposite value of A(x) is impossible, provided that all the guesses are correct. This will induce a $(k+1)$-bounded n^2-extension function F that is dense along A while A avoids F. It will follow by Lemma 2.1 that A is not p-generic.

Let

$$T = \{x \mid x \text{ is not of the form } y10^{4l} \text{ for } y \in \{0,1\}^*,\ l \in \mathbb{Z}^+\}.$$

Clearly $T \in \mathrm{P}$ and T is an infinite set. Given a string $a_0 a_1 \cdots a_{m-1} \in \{0,1\}^*$, let $x = s_m$. If $x \notin T$, the extension function F is undefined at x. If $x \in T$, before we can define $F(x)$, there are k stages.

At stage j ($j = 1, 2, \ldots, k$), suppose the assigned values of the answers for the first $j-1$ questions are $b_1, b_2, \ldots, b_{j-1}$, respectively, where $b_i \in \{0,1\}$, for $i = 1, 2, \ldots, j-1$. Then we assign the jth question to be "$f_{\alpha_j}(x) \in D_{i_j}$?", where $\alpha_j = b_1 b_2 \cdots b_{j-1}$. At this stage, we shall assign a value to $D_{i_j}(f_{\alpha_j}(x))$. If we make a guess at this stage, we shall record it as $A(q_j) = c_j$, where $q_j \in \{0,1\}^*$, $c_j \in \{0,1\}$.

Let $f_{\alpha_j}(x) = u_{j0}1$ if $f_{\alpha_j}(x)$ is of the form $y1$ for $y \in \{0,1\}^*$; let $f_{\alpha_j}(x) = u_{j1}0$ if $f_{\alpha_j}(x)$ is of the form $y0$. We may write $f_{\alpha_j}(x) = u_{jl}\bar{l}$, where $l \in \{0,1\}$ and $\bar{l} = 1 - l$.

There are three cases.

Case 1. $f_{\alpha_j}(x)$ is small. There are three subcases.

Subcase 1.1. $i_j = 1$ (so $D_{i_j} = A_{(4k-4)}$) and $f_{\alpha_j}(x)10^{4k-4} = s_{m_1}$, where $m_1 < m$. Then

$$\begin{aligned} a_{m_1} &= A(f_{\alpha_j}(x)10^{4k-4}) \\ &= A_{(4k-4)}(f_{\alpha_j}(x)). \end{aligned}$$

We assign $b_j = a_{m_1}$. (Note that $b_j = D_{i_j}(f_{\alpha_j}(x))$.)

Subcase 1.2. $2 \leq i_j \leq k-1$, so

$$D_{i_j} = [A_{(4k-4i_j+4)} \cap A_{(4k-4i_j)}] \oplus [A_{(4k-4i_j+4)} \cup A_{(4k-4i_j)}]$$

and $u_{jl}10^{4k-4i_j+4} < s_m$. In this case, let

$$\begin{aligned} u_{jl}10^{4k-4i_j} &= s_{m_1}, \\ u_{jl}10^{4k-4i_j+4} &= s_{m_2}. \end{aligned}$$

Then $m_1 < m_2 < m$, so

$$\begin{aligned} A_{(4k-4i_j)}(u_{jl}) &= a_{m_1} \\ \text{and } A_{(4k-4i_j+4)}(u_{jl}) &= a_{m_2}. \end{aligned}$$

Using these values, we obtain the value of $D_{i_j}(f_{\alpha_j}(x))$. Assign b_j the value of $D_{i_j}(f_{\alpha_j}(x))$ we obtained.

Subcase 1.3. $i_j = k$ (so $D_k = [A_{(4)} \cap S] \oplus [A_{(4)} \cup S]$) and $u_{jl}10^4 < x$. In this case, let $u_{jl}10^4 = s_{m_1}$; then $A_{(4)}(u_{jl}) = a_{m_1}$. Since $S \in \text{DTIME}(2^n)$ and $u_{jl} < s_m$, we can compute $S(u_{jl})$ within $(m+1)$ steps. Then we use a_{m_1} and the value of $S(u_{jl})$ to obtain the value of $D_k(f_{\alpha_j}(x))$. We assign b_j this value.

In Case 2 and Case 3, $f_{\alpha_j}(x)$ is not small. In Case 2, no guess is made, while in most of the setups in Case 3, a guess is made.

Case 2. There are two subcases.

Subcase 2.1. $i_j = 1$, $f_{\alpha_j}(x)10^{4k-4} > x$, and the value of $A(f_{\alpha_j}(x)10^{4k-4})$ has been assigned in the previous stages. In this case, assign b_j the assigned value of $A(f_{\alpha_j}(x)10^{4k-4})$.

Subcase 2.2. $2 \le i_j \le k-1$, $u_{jl}10^{4k-4i_j+4} > x$, and the values of both $A(u_{jl}10^{4k-4i_j+4})$ and $A(u_{jl}10^{4k-4i_j})$ have been assigned. (We assign $A(y)$ a value only when $y \ge x$, so in Subcase 2.2, $u_{jl}10^{4k-4i_j} > x$.) In this case, use the assigned values of $A(u_{jl}10^{4k-4i_j+4})$ and $A(u_{jl}10^{4k-4i_j})$ to compute $D_{i_j}(f_{\alpha_j}(x))$ and assign b_j the value of $D_{i_j}(f_{\alpha_j}(x))$ we obtained.

Case 3. There are three subcases.

Subcase 3.1. $i_j = 1$, $f_{\alpha_j}(x)10^{4k-4} > x$, and the value of $A(f_{\alpha_j}(x)10^{4k-4})$ has not been assigned. We define $q_j = f_{\alpha_j}(x)10^{4k-4}$ and $c_j = 0$. We assign $A(q_j) = 0$ and $b_j = 0$.

Subcase 3.2. $2 \le i_j \le k-1$, $u_{jl}10^{4k-4i_j+4} > x$, and either $A(u_{jl}10^{4k-4i_j+4})$ or $A(u_{jl}10^{4k-4i_j})$ has not been assigned a value. There are three Subsubcases.

Subsubcase 3.2.1 $A(u_{jl}10^{4k-4i_j+4})$ has not been assigned. We define $q_j = u_{jl}10^{4k-4i_j+4}$ and $c_j = l$. We assign $A(q_j) = l$. If $A(q_j) = l$, then $D_{i_j}(f_{\alpha_j}(x)) = l$, so we assign $b_j = l$.

Subsubcase 3.2.2. The value of $A(u_{jl}10^{4k-4i_j+4})$ has been assigned, and $u_{jl}10^{4k-4i_j} < x$. In this case, let $u_{jl}10^{4k-4i_j} = s_{m_1}$, where $m_1 < m$. Then $A_{(4k-4i_j)}(u_{jl}) = a_{m_1}$. Using this and the assigned value of $A(u_{jl}10^{4k-4i_j+4})$, we obtain the value of $D_{i_j}(f_{\alpha_j}(x))$. We assign b_j this value.

Subsubcase 3.2.3. The value of $A(u_{jl}10^{4k-4i_j+4})$ has been assigned, $u_{jl}10^{4k-4i_j} > x$, and the value of $A(u_{jl}10^{4k-4i_j})$ has not been assigned. In this case we do the same work as in Subsubcase 3.2.1 but replace $(4k-4i_j+4)$ by $(4k-4i_j)$.

Subcase 3.3. $i_j = k$, and $u_{jl}10^4 > x$. In this case, we assign $A(u_{jl}10^4) = l$ and $b_j = l$. We define $q_j = u_{jl}10^4$ and $c_j = l$.

After k stages, we have assigned the values for $b_1, b_2, \dots, b_k$. Now we define $b = \overline{h(x, b_1 b_2 \cdots b_k)}$.

Claim 1. If $j_1 \neq j_2$, q_{j_1} and q_{j_2} are defined, then $q_{j_1} \neq q_{j_2}$.

Proof. Since q_j is defined only when $A(q_j)$ has not yet been assigned a value (with the only possible exception in Subcase 3.3), we have $i_{j_1} = k$ or $i_{j_2} = k$. Without loss of generality, we assume that $i_{j_1} = k$. If $j_2 > j_1$ in Subcase 3.2, the value of $A(u_{j_2 l}10^4)$ is not assigned, and q_{j_2} is not defined, so $j_2 < j_1$. In stage j_2 of Subcase 3.2, when we assign the value of $A(u_{j_2 l}10^4)$, the value of $A(u_{j_2 l}10^8)$ must have been assigned in some previous stage, say stage j_3, where $j_3 < j_2$.

By induction, we can show that there exist $j_1 > j_2 > \cdots > j_k$ such that the value of $A(u_{j_r l}10^{4r-4})$ is assigned at stage j_r $(2 \leq r \leq k-1)$, $D_{j_r} = D_{k+1-r}$, the value of $A(f_{\alpha_j}(x)10^{4k-4})$ is assigned at stage j_k of Subcase 3.1, and $D_{j_k} = D_1$. Since $j_1, \dots, j_k \in \{1, 2, \dots, k\}$, it follows that $j_t = k - t + 1$, for $t = 1, 2, \dots, k$, whence $D_{i_j} = D_j$ for $j = 1, 2, \dots, k$, contradicting the nontriviality of the permutation. This proves the Claim. □

Let
$$\{(q_j, c_j)\} \cup \{x, b\} = \{(y_0, d_0), \dots, (y_t, d_t)\},$$
where
$$y_0, y_1, \dots, y_t \in \{0,1\}^*,$$
$$y_0 < y_1 < \cdots < y_t,$$
$$d_0, \dots, d_t \in \{0,1\}.$$

Clearly $t \geq 0$ since $(y_0, d_0) = (x, b)$.

Now we define the extension (partial) function

$$F : \{0,1\}^* \longrightarrow (\{0,1\}^* \times \{0,1\})^*$$

as follows. For $w = a_0 a_1 \cdots a_{m-1} \in \{0,1\}^*$, if $s_m \notin T$, then $F(w)$ is undefined. If $s_m \in T$, then $F(w) = ((y_0, d_0), \ldots, (y_t, d_t))$. It is easy to verify that F is a $(k+1)$-bounded n^2-extension function. Since for every $x \in T$, $F(A\lceil x)$ is defined, and T is an infinite set, F is dense along A. From the fact that $b = \overline{h(x, b_1 b_2 \cdots b_k)}$, for every $x \in T$, it follows that A does not meet F at any $x \in \{0,1\}^*$, whence A avoids F. It follows by Lemma 2.1 that A is not p-generic. □

Theorem 4.3 (Main Theorem). If NP contains a p-generic language, then for every $k \geq 2$ there is a sequence $\vec{D} = (D_1, \ldots, D_k)$ that is strictly sequentially complete for NP.

Proof. Let $A \in \mathrm{NP}$ be p-generic, let $k \geq 2$, and let

$$\vec{D} = \vec{D}(A, \mathrm{SAT}, k) = (D_1, \ldots, D_k).$$

It is clear by inspection that, for all $x \in \{0,1\}^*$, the following three things are true.

(i) $\mathrm{SAT}(x) = D_k(xA_{(4)}(x))$.

(ii) For $1 < i < k$, $A_{(4k-4i)}(x) = D_i(xA_{(4k-4i+4)}(x))$.

(iii) $A_{(4k-4)}(x) = D_1(x)$.

It is easy to convert (i), (ii), and (iii) into a $\leq^{\mathrm{P}}_{\mathrm{S}}$-reduction of SAT to $\vec{D}$. (For example, when $k = 3$, we have $\mathrm{SAT}(x) = D_3(xA_{(4)}(x)) = D_3(xD_2(xA_{(8)}(x))) = D_3(xD_2(xD_1(x)))$.) Thus $\mathrm{SAT} \leq^{\mathrm{P}}_{\mathrm{S}} \vec{D}$. Since SAT is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for NP, it follows immediately that $\vec{D}$ is $\leq^{\mathrm{P}}_{\mathrm{S}}$-hard for NP. Also, since $A \in \mathrm{NP}$ and NP is closed under finite unions, finite intersections, $\oplus$, and projections $A \mapsto A_{(n)}$, we have $D_1, \ldots, D_k \in \mathrm{NP}$ whence $\vec{D}$ is $\leq^{\mathrm{P}}_{\mathrm{S}}$-complete for NP.

To complete the proof, let $\vec{D}' = (D_{i_1}, \ldots, D_{i_k})$ be a nontrivial permutation of $\vec{D}$. Since A is p-generic, the Main Lemma tells us that $A \not\leq^{\mathrm{P}}_{\mathrm{S}} \vec{D}'$. Since $A \in \mathrm{NP}$, it follows that $\vec{D}'$ is not $\leq^{\mathrm{P}}_{\mathrm{S}}$-complete for NP. We have now shown that $\vec{D}$ is $\leq^{\mathrm{P}}_{\mathrm{SS}}$-complete for NP. □

A similar—and simpler—argument establishes the following absolute result.

Theorem 4.4. For every $k \geq 2$ there is a sequence $\vec{D} = (D_1, \ldots, D_k)$ that is strictly sequentially complete for E (hence also for E_2).

We conclude with a brief discussion of strict sequential completeness in NP under strong measure-theoretic hypotheses. Ambos-Spies, Neis, and Terwijn [6] have shown that, if NP does not have p-measure 0 (briefly, $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$), then NP contains a p-generic language. Lorentz and Lutz [10] have recently extended this argument to show that, if there is *any* strongly positive, p-computable probability ν such that $\nu_{\mathrm{p}}(\mathrm{NP}) \neq 0$, then NP contains a p-generic language. Theorem 4.3 thus has the following immediate consequence.

Corollary 4.5. If there exists a strongly positive, p-computable measure ν such that $\nu_{\mathrm{p}}(\mathrm{NP}) \neq 0$, then for every $k \geq 2$ there is a sequence $\vec{D} = (D_1, \ldots, D_k)$ that is strictly sequentially complete for NP.

Acknowledgments. We thank Giora Slutzki, Cliff Bergmann, Elvira Mayordomo, and Steve Kautz for useful discussions.

References

[1] K. Ambos-Spies. Resource-bounded genericity. In S. B. Cooper, et al., editor, *Computability, Enumerability, Unsolvability*, volume 224 of *London Mathematical Society Lecture Notes*, pages 1–59. Cambridge University Press, 1996.

[2] K. Ambos-Spies and L. Bentzien. Separating NP-completeness notions under strong hypotheses. In *Proceedings of the 12th IEEE Conference on Computational Complexity*, pages 121–127, 1997.

[3] K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalizations over polynomial time computable sets. *Theoretical Computer Science*, 51:177–204, 1987.

[4] K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalizing over deterministic polynomial time. In *Proceedings of Computer Science Logic '87*, pages 1–16. Springer-Verlag, 1988.

[5] K. Ambos-Spies and E. Mayordomo. Resource-bounded measure and randomness. In A. Sorbi, editor, *Complexity, Logic and Recursion Theory*, Lecture Notes in Pure and Applied Mathematics, pages 1–47. Marcel Dekker, New York, N.Y., 1997.

[6] K. Ambos-Spies, H.-C. Neis, and S. A. Terwijn. Genericity and measure for exponential time. *Theoretical Computer Science*, 168:3–19, 1996.

[7] J. M. Breutzmann and J. H. Lutz. Equivalence of measures of complexity classes. *SIAM Journal on Computing.* To appear. See also *Proceedings of the 14th Symposium on Theoretical Aspects of Computer Science*, Springer-Verlag, 1997, pp. 535–545.

[8] H. Buhrman and L. Torenvliet. Complete sets and structure in subrecursive classes. In *Proceedings of Logic Colloquium '96*, pages 45–78. Springer-Verlag, 1998.

[9] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. An introduction to query order. *Bulletin of the EATCS*, 63:93–107, 1997.

[10] A. K. Lorentz and J. H. Lutz. Genericity and randomness over feasible probability measures. *Theoretical Computer Science*, 207:245–259, 1998.

[11] J. H. Lutz. The quantitative structure of exponential time. In L.A. Hemaspaandra and A.L. Selman, editors, *Complexity Theory Retrospective II*, pages 225–254. Springer-Verlag, 1997.

[12] J. H. Lutz and E. Mayordomo. Cook versus Karp-Levin: Separating completeness notions if NP is not small. *Theoretical Computer Science*, 164:141–163, 1996.