Commutative Queries

(Extended Abstract)

Richard Beigel[*]
Yale University and
University of Maryland Institute
for Advanced Computer Studies

Richard Chang[†]
University of Maryland Baltimore County

Abstract. This paper considers polynomial-time Turing machines which have access to two oracles and investigates when the order of oracle access is significant. The oracles used here are complete languages for the Polynomial Hierarchy (PH). For language classes, the results in this paper show that the order of the oracle queries does not matter, even when the queries are truth table queries. This improves upon the previous results of Hemaspaandra, Hemaspaandra and Hempel who showed that the order of the queries does not matter if the base machine asks one query to each oracle. Furthermore, the results in this paper extend to polynomial-time machines which make more than two rounds of truth table queries to the oracles.

Let $k > j$ and let H and E be complete languages for Σ_k^{P} and Σ_j^{P} respectively. Then, the new results show that for all polynomial bounded $r(n)$ and $s(n)$,

$$\mathrm{P}^{H_{r(n)\text{-tt}};E_{s(n)\text{-tt}}} = \mathrm{P}^{E_{s(n)\text{-tt}};H_{r(n)\text{-tt}}} = \mathrm{P}^{E_{s(n)\text{-tt}}\|H_{r(n)\text{-tt}}},$$

where $\mathrm{P}^{A_{a(n)\text{-tt}};B_{b(n)\text{-tt}}}$ denotes the class of languages recognized by polynomial-time Turing machines that ask $a(n)$ parallel queries to A followed by $b(n)$ parallel queries to B, and $\mathrm{P}^{A_{a(n)\text{-tt}}\|B_{b(n)\text{-tt}}}$ denotes the class of languages recognized by polynomial-time Turing machines that ask $a(n)$ parallel queries to A simultaneous with $b(n)$ parallel queries to B. In contrast, for function classes the order of access is critical. In particular, for all constants r and s,

$$\mathrm{PF}^{H_{r\text{-tt}};E_{s\text{-tt}}} \not\subseteq \mathrm{PF}^{E_{s\text{-tt}};H_{r\text{-tt}}}$$

unless $\mathrm{PH} = \Sigma_{j+1}^{\mathrm{P}}$. This result also extends to several rounds of queries to H and E. For example, for all constants r, s and t,

$$\mathrm{PF}^{H_{r\text{-tt}};H_{s\text{-tt}};E_{t\text{-tt}}} \not\subseteq \mathrm{PF}^{H_{r\text{-tt}};E_{t\text{-tt}};H_{s\text{-tt}}}$$

unless $\mathrm{PH} = \Sigma_{j+1}^{\mathrm{P}}$.

---

[*]On sabbatical from the Yale University Department of Computer Science. Address: Dept. of Computer Science, University of Maryland at College Park, College Park, MD 20742-3251. Research supported in part by the National Science Foundation under grant CCR-9415410. Also affiliated with the Human–Computer Interaction Laboratory and supported in part by NASA under grant NAG 52895. Email: `beigel@cs.umd.edu`

[†]Address: Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA. Supported in part by National Science Foundation grant CCR-9309137 and by the University of Maryland Institute for Advanced Computer Studies. Email: `chang@umbc.edu`

1 Introduction

In this paper we examine the complexity of languages and functions computed by polynomial-time Turing machines which have access to two oracles. We ask whether the order of the queries is significant. That is, given oracles E and H, does it matter if we ask the queries to E first or to H first? For this question to be nontrivial, the complexity of E and H must be significantly different. Otherwise, the queries to E and H would be trivially interchangeable. In this paper, we choose our oracles E and H to be complete languages for different levels of the Polynomial Hierarchy (PH) — for example, E might be complete for NP and H complete for Σ_2^{P}. (We use E for the "easier" oracle and H for the "harder" one.) Our results show that when a polynomial-time machine is allowed parallel queries to E and H, then the order of the queries does not matter when the machine is recognizing a language — only the number of queries matters. In particular, we show that for all polynomial bounded $r(n)$ and $s(n)$,

$$\mathrm{P}^{H_{r(n)\text{-tt}};E_{s(n)\text{-tt}}} = \mathrm{P}^{E_{s(n)\text{-tt}};H_{r(n)\text{-tt}}},$$

where $\mathrm{P}^{A_{a(n)\text{-tt}};B_{b(n)\text{-tt}}}$ denotes the languages recognized by polynomial-time Turing machines that ask $a(n)$ parallel queries to A followed by $b(n)$ parallel queries to B. This result improves upon the previous results of Hemaspaandra, Hemaspaandra and Hempel [HHH96b] who showed that the order of the queries does not matter if the base machine asks just one query to each oracle. The techniques they use do not generalize to computations that involve more than two queries in total. Furthermore, our new results extend to machines that ask several rounds of queries to E and H. For example, we show that

$$\mathrm{P}^{H_{a\text{-tt}};E_{b\text{-tt}};H_{c\text{-tt}};E_{d\text{-tt}}} = \mathrm{P}^{H_{a\text{-tt}};H_{c\text{-tt}};E_{b\text{-tt}};E_{d\text{-tt}}}.$$

In the proofs of these results, it is simple to show that the queries to the easy oracle E can be delayed — i.e., we can always ask the hard questions first. The difficulty is in showing that the queries to the hard oracle H can also be delayed.

In contrast, when we consider *functions* computable by polynomial-time Turing machines with queries to E and H, the order of the queries becomes critical. We show that every function computed by a machine that queries E first has an equivalent machine that queries H first. That is, for polynomial bounded $r(n)$ and $s(n)$,

$$\mathrm{PF}^{E_{s(n)\text{-tt}};H_{r(n)\text{-tt}}} \subseteq \mathrm{PF}^{H_{r(n)\text{-tt}};E_{s(n)\text{-tt}}}.$$

However, asking queries to H first is strictly more powerful unless PH collapses, because for all constants r and s

$$\mathrm{PF}^{H_{r\text{-tt}};E_{s\text{-tt}}} \subseteq \mathrm{PF}^{E_{s\text{-tt}};H_{r\text{-tt}}} \Longrightarrow \mathrm{PH} \subseteq \mathrm{NP}^{E}.$$

This result also extends to more than two rounds of parallel queries. For example, for all constants r, s and t

$$\mathrm{PF}^{H_{r\text{-tt}};H_{s\text{-tt}};E_{t\text{-tt}}} \subseteq \mathrm{PF}^{H_{r\text{-tt}};E_{t\text{-tt}};H_{s\text{-tt}}} \Longrightarrow \mathrm{PH} \subseteq \mathrm{NP}^{E}.$$

Several other studies have examined the effect of the order of access to multiple oracles. Hemaspaandra, Hempel and Wechsung [HHW95] determined when the order of queries to complete languages for the Boolean Hierarchy can be reversed. Stronger results were obtained independently by Agrawal, Beigel and Thierauf [ABT96]. Also, Gasarch and McNicholl [GM96] have investigated the order of oracle queries in a recursion theoretic setting.

2 Definition and Notation

For bounded queries to a single oracle, we use standard notation from the literature. For multiple oracle queries, new notation is needed.

Definition 1

- Let $\mathrm{P}^{A_{a(n)\text{-tt}};B_{b(n)\text{-tt}}}$ denote the class of languages recognized by polynomial-time Turing machines that ask $a(n)$ parallel (or truth table) queries to the oracle A followed by $b(n)$ parallel queries to the oracle B on inputs of length n.
- Let $\mathrm{P}^{A_{a(n)\text{-tt}}||B_{b(n)\text{-tt}}}$ denote the class of languages recognized by polynomial-time Turing machines that ask $a(n)$ parallel queries to A simultaneous with $b(n)$ parallel queries to B.
- Let $\mathrm{PF}^{A_{a(n)\text{-tt}};B_{b(n)\text{-tt}}}$ denote the class of functions recognized by polynomial-time Turing machines that ask $a(n)$ parallel queries to A followed by $b(n)$ parallel queries to B.
- Let $\mathrm{PF}^{A_{a(n)\text{-tt}}||B_{b(n)\text{-tt}}}$ denote the class of functions recognized by polynomial-time Turing machines that ask $a(n)$ parallel queries to A simultaneous with $b(n)$ parallel queries to B.

Note that $\mathrm{P}^{A_{a(n)\text{-tt}}||B_{b(n)\text{-tt}}}$ is trivially contained in both $\mathrm{P}^{A_{a(n)\text{-tt}};B_{b(n)\text{-tt}}}$ and $\mathrm{P}^{B_{b(n)\text{-tt}};A_{a(n)\text{-tt}}}$. Classes of languages and functions defined by machines that ask more than two rounds of parallel queries are defined similarly. For example, $\mathrm{P}^{A_{a\text{-tt}};B_{b\text{-tt}};C_{c\text{-tt}};D_{d\text{-tt}}}$ is the class of languages accepted by polynomial-time Turing machines that ask a queries to A, b queries to B, c queries to C and d queries to D in that order.

Notation 2

- $A(x)$ is the characteristic function of the set A at x
- $\chi_k^A(x_1, \ldots, x_k) = A(x_1) \cdots A(x_k)$, where juxtaposition means concatenation
- $\#_k^A(x_1, \ldots, x_k) = \|A \cap \{\, x_1, \ldots, x_k \,\}\|$
- $Y^{[m]} = \{S : S \subseteq Y \text{ and } |S| \leq m\}$
- $A \times A = \{(x, y) : x \in A \text{ and } y \in A\}$.

Definition 3 A function g from X to Y is m-enumerable if there is a polynomial-time computable function f from X to $Y^{[m]}$ such that $(\forall x)[g(x) \in f(x)]$.

Note that if g can be computed by a polynomial-time machine that makes k queries to A then g is 2^k-enumerable. The function χ_k^A can be computed using k parallel queries to A. In many cases, it has been shown that k queries to A are necessary. In particular, if A is disjunctively self-reducible and χ_k^A is $(2^k - 1)$-enumerable, then $A \in \mathrm{P}$ using a tree pruning procedure [BKS95]. We will use the tree pruning techniques of Amir, Beigel and Gasarch [ABG90] and of Beigel, Kummer and Stephan [BKS95] in the proofs below. It will also be helpful if the reader is familiar with mind-change proofs, which have been used to show the relationships between serial and parallel queries [Bei91], and with hard/easy arguments, which have been used to show that a collapse of the Boolean Hierarchy implies a collapse of the Polynomial Hierarchy [Kad88, BCO93, CK96, HHH96a, BF96].

3 Language Classes

In this section we consider classes of languages recognized by polynomial-time Turing machines which have access to a Σ_k^{P} oracle and a Σ_j^{P} oracle. The results in this section show that for language classes, the order of the queries does not matter — in fact, the queries can be made in parallel. In Lemma 4, we show that when the easier questions are asked first, then the queries can be made in parallel. This relationship even holds for function classes. This is the simple direction of our results; the difficult direction handles the case where the harder questions are asked first. The proof of this lemma is a modification of the techniques used by Hemaspaandra *et al.* [HHH96b].

Lemma 4 For $k > j$, let H and E be $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for Σ_k^{P} and Σ_j^{P} respectively. Then, for all polynomial bounded $r(n)$ and $s(n)$,

$$\mathrm{PF}^{E_{s(n)\text{-tt}};H_{r(n)\text{-tt}}} \subseteq \mathrm{PF}^{E_{s(n)\text{-tt}}\|H_{r(n)\text{-tt}}} \subseteq \mathrm{PF}^{H_{r(n)\text{-tt}};E_{s(n)\text{-tt}}}.$$

Proof: The second containment is obvious. Let M be a polynomial-time bounded Turing machine that asks $s(n)$ parallel queries to the oracle E followed by $r(n)$ parallel queries to the oracle H. Let $e_1, \ldots, e_{s(n)}$ be the queries that M asks the oracle E on a particular input x. Note that the queries $e_1, \ldots, e_{s(n)}$ can be generated in polynomial time. Since $k > j$, a Σ_k^{P} machine can generate the set of queries $e_1, \ldots, e_{s(n)}$, determine the answers to these queries and then generate the second set of queries $h_1, \ldots, h_{r(n)}$ that $M(x)$ would ask to the oracle H. Thus, M' does not have to query E before asking H about the answers to $h_1, \ldots, h_{r(n)}$. The machine M' can simply ask the oracle H the following question h_i': "Compute h_i, the ith query that $M(x)$ asks H and determine whether $h_i \in H$." The oracle H can answer such queries because H is complete for Σ_k^{P}. Clearly, $h_i' \in H$ if and only if $h_i \in H$. In parallel with the queries to H, $M'(x)$ also asks the oracle E for answers to $e_1, \ldots, e_{s(n)}$, the same questions that $M(x)$ asked originally. Thus, $M'(x)$ has answers to all of the oracle queries that $M(x)$ asked and $M'(x)$ can complete the simulation of $M(x)$ step by step. □

Note that we do not really need H to be complete for Σ_k^{P}. The condition

$$\leq_{\mathrm{m}}^{\mathrm{NP}}(H) = \leq_{\mathrm{conj}}^{\mathrm{P}}(H) = \leq_{\mathrm{m}}^{\mathrm{P}}(H),$$

is sufficient to prove Lemma 4. Here $\leq_{\mathrm{m}}^{\mathrm{NP}}(H)$, $\leq_{\mathrm{conj}}^{\mathrm{P}}(H)$ and $\leq_{\mathrm{m}}^{\mathrm{P}}(H)$ are respectively the classes of languages that are nondeterministically, conjunctively and many-one reducible to H. For constant $r(n)$ and $s(n)$, we only need $E \leq_{\mathrm{m}}^{\mathrm{P}} H$, $\overline{E} \leq_{\mathrm{m}}^{\mathrm{P}} H$, $H \times H \leq_{\mathrm{m}}^{\mathrm{P}} H$ and $\overline{H} \times \overline{H} \leq_{\mathrm{m}}^{\mathrm{P}} \overline{H}$. Also, by restricting Lemma 4 to characteristic functions, we obtain the following corollary for language classes.

Corollary 5 For $k > j$, let H and E be $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for Σ_k^{P} and Σ_j^{P} respectively. Then, for all polynomial bounded $r(n)$ and $s(n)$,

$$\mathrm{P}^{E_{s(n)\text{-tt}};H_{r(n)\text{-tt}}} \subseteq \mathrm{P}^{E_{s(n)\text{-tt}}\|H_{r(n)\text{-tt}}} \subseteq \mathrm{P}^{H_{r(n)\text{-tt}};E_{s(n)\text{-tt}}}.$$

Lemma 4 and Corollary 5 show that we can always postpone the easy questions (the queries to E). In the next theorem, we show somewhat surprisingly that — when recognizing languages — we can also postpone the hard questions. In fact, in either case, all the questions can be asked in parallel. The proof of this theorem uses the mind-change technique. For those familiar with this technique, the basic structure of this proof is the same as the proof which shows that every language in $\mathrm{P}^{H_{r(n)\text{-tt}}}$ can be reduced to computing $\#^H_{r(n)}$, except in this case the polynomial-time machine is allowed $s(n)$ parallel queries to E [Bei91]. As before, we can prove Theorem 6 using $\leq^{\mathrm{NP}}_{\mathrm{m}}(H) = \leq^{\mathrm{P}}_{\mathrm{conj}}(H) = \leq^{\mathrm{P}}_{\mathrm{m}}(H)$ instead of the Σ^{P}_k-completeness of H. When $r(n)$ and $s(n)$ are constants, we really only need $E \leq^{\mathrm{P}}_{\mathrm{m}} H$, $\overline{E} \leq^{\mathrm{P}}_{\mathrm{m}} H$, $H \times H \leq^{\mathrm{P}}_{\mathrm{m}} H$ and $\overline{H} \times \overline{H} \leq^{\mathrm{P}}_{\mathrm{m}} \overline{H}$.

Theorem 6 For $k > j$, let H and E be $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for Σ^{P}_k and Σ^{P}_j respectively. Then, for all polynomial bounded $r(n)$ and $s(n)$,

$$\mathrm{P}^{H_{r(n)\text{-tt}};E_{s(n)\text{-tt}}} = \mathrm{P}^{H_{r(n)\text{-tt}}\|E_{s(n)\text{-tt}}}.$$

Proof: $\mathrm{P}^{H_{r(n)\text{-tt}}\|E_{s(n)\text{-tt}}} \subseteq \mathrm{P}^{H_{r(n)\text{-tt}};E_{s(n)\text{-tt}}}$ is obvious. Let L belong to $\mathrm{P}^{H_{r(n)\text{-tt}};E_{s(n)\text{-tt}}}$ via a polynomial-time Turing machine M. Let $Q = \{h_1, \ldots, h_{r(n)}\}$ be the set of queries to H made by M on a particular input x, and let $Z = Q \cap H$. Given the input x and a set $Z' \subseteq Q$, let the value of $f(Z', x)$ be true if $M(x)$ accepts when the queries to H were answered according to Z'. Note that $f(Z', x)$ can be computed in polynomial time using $s(n)$ parallel queries to E. Let $g(m, x)$ be true if and only if M can make m "mind changes" on input x with respect to the queries to H — i.e.,

$$g(m, x) = (\exists Z_0, \ldots, Z_m)[Z_0 = \emptyset \wedge (\forall 1 \leq i \leq m)[(Z_{i-1} \subset Z_i \subseteq Z) \wedge (f(Z_{i-1}, x) \neq f(Z_i, x))]].$$

Since a Σ^{P}_k machine has an oracle that can answer E queries, a Σ^{P}_k machine can determine whether $g(m, x)$ is true by guessing an increasing sequence $Z_0, \ldots, Z_m$, checking that $Z_i \subseteq Z$, and confirming that $f(Z_{i-1}, x) \neq f(Z_i, x)$ for $1 \leq i \leq m$. Thus, the maximum number of mind changes, $\max\{\, m : g(m, x) \wedge (1 \leq m \leq r(n)) \,\}$, can be computed using $r(n)$ parallel queries to H. Furthermore, $M(x) = f(\emptyset, x)$ if and only if the maximum number of mind changes is even. Therefore, using $s(n)$ parallel queries to E to compute $f(\emptyset, x)$ and $r(n)$ parallel queries to H to compute the maximum number of mind changes, we can determine whether $M(x)$ accepts. Hence, L can be recognized in $\mathrm{P}^{H_{r(n)\text{-tt}}\|E_{s(n)\text{-tt}}}$. □

Corollary 5 shows that for language classes, asking the easy questions first is equivalent to asking the hard questions first. In fact, this observation generalizes to several rounds of parallel queries to the oracles H and E. Again, there is an equivalent machine that asks all the parallel queries to H before the parallel queries to E. Thus, for example,

$$\mathrm{P}^{H_{a\text{-tt}};E_{b\text{-tt}};H_{c\text{-tt}};E_{d\text{-tt}}} \subseteq \mathrm{P}^{H_{a\text{-tt}};H_{c\text{-tt}};E_{b\text{-tt}};E_{d\text{-tt}}}.$$

In the case of languages, we can use a result of Beigel [Bei91, Theorem 4.9] to combine consecutive rounds of parallel queries to the same oracle into a single round. Thus,

$$\mathrm{P}^{H_{a\text{-tt}};H_{c\text{-tt}};E_{b\text{-tt}};E_{d\text{-tt}}} \subseteq \mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}},$$

where $r = (a+1)(c+1) - 1$ and $s = (b+1)(d+1) - 1$. Furthermore, by Theorem 6, $\mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}} = \mathrm{P}^{E_{s\text{-tt}};H_{r\text{-tt}}} = \mathrm{P}^{H_{r\text{-tt}}\|E_{s\text{-tt}}}$. We can also show that

$$\mathrm{P}^{H_{r\text{-tt}}\|E_{s\text{-tt}}} \subseteq \mathrm{P}^{H_{a\text{-tt}};E_{b\text{-tt}};H_{c\text{-tt}};E_{d\text{-tt}}}.$$

(The proof is omitted in this extended abstract.) Therefore, the classes $\mathrm{P}^{H_{a\text{-tt}};E_{b\text{-tt}};H_{c\text{-tt}};E_{d\text{-tt}}}$, $\mathrm{P}^{H_{a\text{-tt}};H_{c\text{-tt}};E_{b\text{-tt}};E_{d\text{-tt}}}$, $\mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}}$, $\mathrm{P}^{E_{s\text{-tt}};H_{r\text{-tt}}}$ and $\mathrm{P}^{H_{r\text{-tt}}\|E_{s\text{-tt}}}$ are all equal. Thus, for language classes, the complexity is characterized by the number of queries asked to the oracles H and E and not by the order of the queries.

In fact, the following lemma shows that the $\mathrm{P}^{H_{r\text{-tt}}\|E_{s\text{-tt}}}$ classes form a nice linear hierarchy where additional queries to E are nested inside the additional queries to H:

$$\mathrm{P}^{H_{r\text{-tt}}} \subseteq \mathrm{P}^{H_{r\text{-tt}}\|E_{1\text{-tt}}} \subseteq \mathrm{P}^{H_{r\text{-tt}}\|E_{2\text{-tt}}} \subseteq \cdots \subseteq \mathrm{P}^{H_{r+1\text{-tt}}}$$

Lemma 7 For $k > j$, let H and E be $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for Σ_k^{P} and Σ_j^{P} respectively. Then, for all polynomial bounded $r(n)$ and $s(n)$, $\mathrm{P}^{H_{r(n)\text{-tt}}\|E_{s(n)\text{-tt}}} \subseteq \mathrm{P}^{H_{r(n)+1\text{-tt}}}$.

Proof: Corollary 5 and Theorem 6 show that under this lemma's hypotheses:

$$\mathrm{P}^{E_{s(n)\text{-tt}};H_{r(n)\text{-tt}}} = \mathrm{P}^{H_{r(n)\text{-tt}};E_{s(n)\text{-tt}}} = \mathrm{P}^{H_{r(n)\text{-tt}}\|E_{s(n)\text{-tt}}}.$$

Let L be a language in $\mathrm{P}^{H_{r(n)\text{-tt}};E_{s(n)\text{-tt}}}$ via a polynomial-time Turing machine M. In the proof of Theorem 6, we showed that $M(x) = f(\emptyset, x)$ if and only if the maximum number of mind changes, $\max\{\, m : g(m, x) \wedge 1 \leq m \leq r(n)\,\}$, is even. Recall that $f(\emptyset, x)$ is a characteristic function computable using $s(n)$ parallel queries to E and that $r(n)$ parallel queries to H is sufficient to calculate the maximum number of mind changes. Since H is Σ_k^{P} complete, one query to H is also sufficient to compute the value of $f(\emptyset, x)$. Thus, using $r(n) + 1$ parallel queries to H, we can determine whether $M(x)$ accepted or rejected. □

Finally, we note that even when two oracles are used, each additional query adds additional computational power (unless PH collapses). The proof of the following lemma uses a fairly straightforward hard/easy argument and is omitted from this extended abstract.

Lemma 8 For $k > j$, let H and E be $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for Σ_k^{P} and Σ_j^{P} respectively. Then, for all $0 < \epsilon < 1$ and for all $r(n)$ and $s(n)$ in $O(n^\epsilon)$, $\mathrm{P}^{H_{r(n)\text{-tt}}\|E_{s(n)\text{-tt}}} = \mathrm{P}^{H_{r(n)\text{-tt}}\|E_{s(n)+1\text{-tt}}}$ implies that PH collapses.

Corollary 9 For $k > j$, let H and E be $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for Σ_k^{P} and Σ_j^{P} respectively. Assuming that PH does not collapse, for all $0 < \epsilon < 1$ and for all $r(n)$ and $s(n)$ in $O(n^\epsilon)$, we have the strict containments:

$$\mathrm{P}^{H_{r(n)\text{-tt}}\|E_{s(n)\text{-tt}}} \subset \mathrm{P}^{H_{r(n)\text{-tt}}\|E_{s(n)+1\text{-tt}}} \subset \mathrm{P}^{H_{r+1\text{-tt}}}.$$

4 Function Classes

In this section we show that for *functions* computed by polynomial-time Turing machines with access to two oracles, the order of the queries is critical. Lemma 4 showed that if the easy questions are asked first, then there is an equivalent machine that asks the hard questions and the easy questions in parallel. The following theorem shows that the converse does not hold unless the Polynomial Hierarchy collapses.

Theorem 10 For $k > j \geq 1$, let H and E be $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for Σ_k^{P} and Σ_j^{P} respectively. Then, for all constant r and s, $\mathrm{PF}^{H_{r\text{-tt}};E_{s\text{-tt}}} \not\subseteq \mathrm{PF}^{E_{s\text{-tt}};H_{r\text{-tt}}}$ unless $\mathrm{PH} \subseteq \Sigma_{j+1}^{\mathrm{P}}$.

The proof of Theorem 10 can be best illustrated by first proving the restricted case where H is Σ_2^{P} complete, E is NP complete and just one query is asked to each oracle.

Theorem 11 Let H and E be $\leq_m^P$-complete for Σ_2^P and NP respectively. Suppose that $\mathrm{PF}^{H_{1\text{-tt}};E_{1\text{-tt}}} \subseteq \mathrm{PF}^{E_{1\text{-tt}};H_{1\text{-tt}}}$. Then, $\mathrm{PH} \subseteq \Sigma_2^P$.

Proof: Consider the function:

$$f(x,y,z) = \begin{cases} H(x)E(y) & \text{if } x \notin H \\ H(x)E(z) & \text{if } x \in H \end{cases}$$

Recall that $H(\cdot)$ and $E(\cdot)$ denote the characteristic functions of H and E and that $H(x)E(y)$ represents the concatenation of $H(x)$ and $E(y)$. Note that $f(x,y,z)$ is easily computable by a $\mathrm{PF}^{H_{1\text{-tt}};E_{1\text{-tt}}}$ machine M that asks whether $x \in H$ followed by the appropriate query to E. However, there is no obvious way to compute f in $\mathrm{PF}^{E_{1\text{-tt}};H_{1\text{-tt}}}$ since it is not clear which of $y \in E$ and $z \in E$ to ask.

If $\mathrm{PF}^{H_{1\text{-tt}};E_{1\text{-tt}}} \subseteq \mathrm{PF}^{E_{1\text{-tt}};H_{1\text{-tt}}}$, then $f \in \mathrm{PF}^{E_{1\text{-tt}};H_{1\text{-tt}}}$ via some polynomial-time Turing machine M'. Consider the branches of the oracle query trees of M and M' on input (x,y,z) where the queries to E are answered correctly. Let $\mathrm{OUT}^E(M(x,y,z))$ and $\mathrm{OUT}^E(M'(x,y,z))$ denote the set of outputs of the machines on these paths. Since the queries to H may be correctly or incorrectly answered, the sets $\mathrm{OUT}^E(M(x,y,z))$ and $\mathrm{OUT}^E(M'(x,y,z))$ have at most two values each. From our construction of M, we know that

$$\mathrm{OUT}^E(M(x,y,z)) = \{\, 0E(y), 1E(z) \,\}.$$

Now, suppose that $\mathrm{OUT}^E(M'(x,y,z)) \neq \mathrm{OUT}^E(M(x,y,z))$. Since both M and M' compute f, the correct value of $f(x,y,z)$ must appear in both sets. Thus,

$$\begin{aligned} &\mathrm{OUT}^E(M'(x,y,z)) \neq \mathrm{OUT}^E(M(x,y,z)) \\ &\quad \Longrightarrow \mathrm{OUT}^E(M'(x,y,z)) \cap \mathrm{OUT}^E(M(x,y,z)) = \{f(x,y,z)\}. \end{aligned}$$

By definition, the first bit of $f(x,y,z)$ is $H(x)$. So, if there exists (y,z) such that the sets $\mathrm{OUT}^E(M'(x,y,z))$ and $\mathrm{OUT}^E(M(x,y,z))$ differ, we can determine whether $x \in H$ or $x \in \overline{H}$ by guessing (y,z) and answering queries to E. When such a (y,z) pair exists, we call x an *easy string*. Otherwise, no such (y,z) pair exists and we call x a *hard string*. For each length n, we will consider two cases. Either all of the strings of length n are easy or there exists a hard string for length n. We will show below that we can collapse PH in either case. Formally, the two cases are:

Case 1: (All strings of length n are easy.)

$$(\forall x, |x| = n)(\exists y, z, |y| = |z| = n)[\mathrm{OUT}^E(M(x,y,z)) \neq \mathrm{OUT}^E(M'(x,y,z))].$$

Case 2: (There exists a hard string for length n.)

$$(\exists x, |x| = n)(\forall y, z, |y| = |z| = n)[\mathrm{OUT}^E(M(x,y,z)) = \mathrm{OUT}^E(M'(x,y,z))].$$

For each length n, we can provide an $\mathrm{NP}^{\mathrm{NP}}$ machine with 1 bit of advice saying which of Case 1 or 2 holds. If Case 1 holds, then the $\mathrm{NP}^{\mathrm{NP}}$ machine can determine whether $x \in \overline{H}$ by guessing (y,z) and checking whether $\mathrm{OUT}^E(M(x,y,z)) \neq \mathrm{OUT}^E(M'(x,y,z))$. When the "correct" (y,z) is guessed, $\mathrm{OUT}^E(M(x,y,z)) \cap \mathrm{OUT}^E(M'(x,y,z))$ contains one string ab where $a = H(x)$. Thus, when all strings are easy, there exist $\mathrm{NP}^{\mathrm{NP}}$ machines which can recognize H and $\overline{H}$. Since $\overline{H}$ is Π_2^P complete, this would collapse PH.

If there exists a hard string for length n, the advice function also provides the hard string x. Note that $\mathrm{OUT}^{E}(M'(x,y,z))$ can be computed using only 1 query to E. However, in Case 2 we have $\mathrm{OUT}^{E}(M'(x,y,z)) = \mathrm{OUT}^{E}(M(x,y,z)) = \{\, 0E(y), 1E(z) \,\}$. Thus, a P machine with the advice and 1 query to E can compute χ_2^E. Using the tree pruning technique of Amir, Beigel and Gasarch [ABG90], we can show that E can be solved in P with advice (proof omitted).

So, for each length n, given polynomial advice, we either have $H \in \mathrm{NP}^{E}$ (Case 1) or $E \in \mathrm{P}$ (Case 2). If we combined the two cases carefully, we have PH collapsing to Σ_4^{P} using the fact that $\Pi_2^{\mathrm{P}} \subseteq \Sigma_2^{\mathrm{P}}/\mathrm{poly}$ and Yap's theorem [Yap83]. We can improve upon the collapse of PH in two ways. First, we can modify the definitions of Case 1 and 2 to get $H \in \mathrm{P}_{\mathrm{tt}}^{E}$ in Case 1 and $E \in \mathrm{P}$ in Case 2. This method uses the techniques of Amir, Beigel and Gasarch [ABG90] to construct a better polynomial advice function. This improvement would collapse PH to Σ_3^{P}. The second method uses the latest refinements of the hard/easy argument [HHH96a, BF96] to show that PH actually collapses to Σ_2^{P}. One key difference in the new approach is that we do not have to look for a hard string x; we simply guess whether the input string is a hard string. We sketch the proof of the second method below.

We construct an $\mathrm{NP}^{\mathrm{NP}}$ machine N to recognize $\overline{H}$ as follows. First, we rewrite $\overline{H}$ as $\{\, x : (\forall u)(\exists v)[R(x,u,v)] \,\}$ for some polynomial-time computable predicate R. On input x, the computation of N is divided into two "parallel" strategies. The first strategy supposes that x is an easy string. Thus, in the first strategy, N guesses (y,z) and checks whether $\mathrm{OUT}^{E}(M(x,y,z)) \neq \mathrm{OUT}^{E}(M'(x,y,z))$ using its NP oracle. Note that when the correct pair (y,z) is found, the machine N "knows" that x is an easy string. In this case, let ab be the single string in $\mathrm{OUT}^{E}(M(x,y,z)) \cap \mathrm{OUT}^{E}(M'(x,y,z))$. Then, $x \in \overline{H}$ if and only if $a = 0$. If no such (y,z) pair exists, then all computation paths following the first strategy will reject.

The second strategy supposes that x is a hard string. In this case, N asks its NP oracle whether $(\forall u)(\exists v)[R(x,u,v)]$. Normally, an NP oracle cannot answer this Π_2^{P} question. However, using x as a hard string, there is a polynomial-time tree pruning procedure to find the witness v such that $R(x,u,v)$ is true (for any fixed u). Let, N' be an NP machine which guesses u and looks for the witness v deterministically using the tree pruning technique of Amir, Beigel and Gasarch [ABG90]. If such a witness v is found, then N' rejects. So, if $N'(x)$ rejects on all paths, $x \in \overline{H}$. Hence, in the second strategy, the base machine N will simply ask the NP oracle whether $N'(x)$ rejects. Note that in the second strategy, the base machine N will accept only if the search for witnesses succeeds for all u. Thus, acceptance by the second strategy is correct even if it turns out that x is not a hard string. Combining the two strategies, we have an $\mathrm{NP}^{\mathrm{NP}}$ algorithm for $\overline{H}$. Therefore, PH collapses to Σ_2^{P}. □

We are now ready to sketch a proof of Theorem 10. The outline of this proof is the same as the proof of the restricted case. That is, if all strings are easy, then $\overline{H} \subseteq \mathrm{NP}^{E}$. If there is a hard string x, then we can use x to run a tree pruning procedure on witnesses for E. The main difference in the proof below is that in the easy case $\mathrm{OUT}^{E}(M(\cdot)) \cap \mathrm{OUT}^{E}(M'(\cdot))$ may contain many strings. So, we cannot determine whether $x \in \overline{H}$ directly. Instead, we conclude that χ_r^H is $(2^r - 1)$-enumerable by an NP^{E} machine. Fortunately, this is sufficient for us to mimic the tree pruning technique of Beigel, Kummer and Stephan [BKS95] on the witnesses for H.

Proof Sketch (Theorem 10): To shorten our notation, let $m = 2^r$. Consider the function

$$f(x_1, \ldots, x_r, y_{0,1}, \ldots, y_{0,s}, \ldots, y_{m-1,1}, \ldots, y_{m-1,s}) = \sigma\chi_s^E(y_{\sigma,1}, \ldots, y_{\sigma,s}),$$

where $\sigma = \chi_r^H(x_1, \ldots, x_r)$. Clearly, f can be computed by a $\mathrm{PF}^{H_{r\text{-tt}};E_{s\text{-tt}}}$ machine M which uses r parallel queries to H to compute $\sigma = \chi_r^H(x_1, \ldots, x_r)$ and then uses s parallel queries to E to compute $\chi_s^E(y_{\sigma,1}, \ldots, y_{\sigma,s})$. Let M' be a $\mathrm{PF}^{E_{s\text{-tt}};H_{r\text{-tt}}}$ machine which computes f.

For $\vec{x} = (x_1, \ldots, x_r)$ and $\vec{y} = (y_{0,1}, \ldots, y_{m-1,s})$, let $\mathrm{OUT}^E(M(\vec{x}, \vec{y}))$ be the set of outputs of $M(\vec{x}, \vec{y})$ on branches of the oracle query tree where the queries to E are answered correctly. The set $\mathrm{OUT}^E(M'(\vec{x}, \vec{y}))$ is defined analogously. From the description of the machine M, we know that

$$\mathrm{OUT}^E(M(\vec{x}, \vec{y})) = \{\, \sigma\chi_s^E(y_{\sigma,1}, \ldots, y_{\sigma,s}) : \sigma \in \{0,1\}^r \,\}.$$

Fix a length n and consider only strings x_i and $y_{\sigma,i}$ of length n. Suppose that $\vec{x}$ is easy. That is, for some $\vec{y}$, $\mathrm{OUT}^E(M(\vec{x}, \vec{y})) \neq \mathrm{OUT}^E(M'(\vec{x}, \vec{y}))$. Since $\mathrm{OUT}^E(M'(\vec{x}, \vec{y}))$ has at most 2^r elements, for at least one $\sigma \in \{0,1\}^r$, $\sigma\chi_s^E(y_{\sigma,1}, \ldots, y_{\sigma,s}) \notin \mathrm{OUT}^E(M'(\vec{x}, \vec{y}))$. Then, we can eliminate σ as a possible value for $\chi_r^H(x_1, \ldots, x_r)$. That is, χ_r^H is $(2^r - 1)$-enumerable. Using the tree pruning technique of Beigel, Kummer and Stephan [BKS95], we can construct an NP^E machine to recognize either H or $\overline{H}$.

On the other hand, suppose that there exists a hard string $\vec{x}$. That is, for this $\vec{x}$, for every $\vec{y}$, we have $\mathrm{OUT}^E(M(\vec{x}, \vec{y})) = \mathrm{OUT}^E(M'(\vec{x}, \vec{y}))$. Note that $\mathrm{OUT}^E(M'(\vec{x}, \vec{y}))$ can be computed using just s parallel queries to E since the machine M' asks its queries to E first. Thus, using just s parallel queries to E, we can determine $\chi_{ms}^E(\vec{y})$ — the answers to $m \cdot s$ parallel queries to E. Again, using the tree pruning technique of Beigel, Kummer and Stephan, we can construct a P machine to recognize either E or $\overline{E}$.

It remains to show that the two tree pruning procedures can be combined into an NP^E algorithm for a language $L \in \Pi_{j+1}^{\mathrm{P}}$. Since $k > j$, $L \leq_{\mathrm{m}}^{\mathrm{P}} \overline{H}$. We also rewrite L as $\{\, x : (\forall u)(\exists v) R(x, u, v) \,\}$, where R is a $\Delta_{j-1}^{\mathrm{P}}$ predicate. Our NP^E machine carries out a top level tree pruning procedure on the witnesses of H to determine whether an input string $x \in L$. This tree pruning procedure will generate many instances of $\vec{x}$. For the top level tree pruning procedure, we will just assume that all such $\vec{x}$ are easy. For each instance, the correctness of this assumption can be confirmed by guessing a $\vec{y}$ such that $\mathrm{OUT}^E(M(\vec{x}, \vec{y})) \neq \mathrm{OUT}^E(M'(\vec{x}, \vec{y}))$. The NP^E machine can determine whether it has guessed the correct $\vec{y}$, because it can use the E oracle to compute both sets. Only the paths that guessed the correct $\vec{y}$ will "survive" and carry on the tree pruning procedure.

On the other hand, we must guard against the possibility that some particular $\vec{x}$ might be hard. Thus, for each $\vec{x}$ we encounter in the top level tree pruning procedure, we initiate a separate tree pruning procedure to recognize L in P^E. Without any assumptions, L can be recognized in coNP^E. However, if $\vec{x}$ is indeed a hard string, we can use $\vec{x}$ and a polynomial-time procedure to search for a witness v such that $R(x, u, v)$ is true. The correctness of the witness v can be verified by a $\Delta_{j-1}^{\mathrm{P}}$ procedure, since R is a $\Delta_{j-1}^{\mathrm{P}}$ predicate. Thus, a $\mathrm{coNP}^{\Sigma_{j-1}^{\mathrm{P}}}$ procedure is sufficient to recognize L. Since E is Σ_j^{P} complete, one query to E is sufficient to determine whether $x \in L$, assuming that $\vec{x}$ is indeed a hard string. Note that even if x is not a hard string, this procedure will accept only if $x \in L$ because it must verify a witness v for every u. So, in both tree pruning procedures, the error is one-sided — the strategies might reject incorrectly, but never accept incorrectly. Therefore, either all the $\vec{x}$ are easy strings and the top level tree pruning succeeds, or some $\vec{x}$ is a hard string and the second tree pruning procedure succeeds. In either case, the overall procedure accepts x

when $x \in L$ and rejects on all branches when $x \notin L$. Thus, $\Pi^{\mathrm{P}}_{j+1} \subseteq \mathrm{NP}^E$ and PH collapses to $\Sigma^{\mathrm{P}}_{j+1}$. □

The proof of Theorem 10 can be extended to the case where more than two rounds of queries are made. For example, using a very similar proof, we can show that

$$\mathrm{PF}^{H_{r\text{-tt}};H_{s\text{-tt}};E_{t\text{-tt}}} \subseteq \mathrm{PF}^{H_{r\text{-tt}};E_{t\text{-tt}};H_{s\text{-tt}}} \Longrightarrow \mathrm{PH} \subseteq \mathrm{NP}^E.$$

In this proof, the two sets $\mathrm{OUT}^E(M(\vec{x}, \vec{y}))$ and $\mathrm{OUT}^E(M'(\vec{x}, \vec{y}))$ would be defined as before. If the two sets are not equal, then χ^H_{r+s} is $(2^{r+s} - 1)$-enumerable. If the two sets are equal, then χ^E_m is 2^{t2^r}-enumerable, where $m = t2^{r+s}$. Combining the two tree pruning procedures produces an NP^E algorithm for a coNP^E language.

It is interesting to note that in the proof of Theorem 10, the complexity of the language H is not used very much. The only requirement that we need for H is that it is hard for coNP^E. In fact, even when E is NP complete and H is Σ^{P}_{17} complete, we only use the fact that all Π^{P}_2 languages reduce to H. The same holds for H being PSPACE complete. It is only when the complexity of H is so high that a set bi-immune to NP^E reduces H that we are able exploit the properties of H. In this case, we know that there can only be finitely many easy strings, thus we can conclude that $\mathrm{PF}^{H_{r\text{-tt}};E_{s\text{-tt}}} = \mathrm{PF}^{E_{s\text{-tt}};H_{r\text{-tt}}} \Longrightarrow E \in \mathrm{P}$.

Acknowledgments

The authors would like to thank Bill Gasarch for proofreading this paper and for many fruitful discussions.

References

[ABG90] A. Amir, R. Beigel, and W. I. Gasarch. Some connections between bounded query classes and non-uniform complexity. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 232–243, 1990.

[ABT96] M. Agrawal, R. Beigel, and T. Thierauf. Modulo information from nonadaptive queries to NP. Technical Report TR 96-001, The Electronic Computational Complexity Colloquium, 1996. To appear in *Proceedings of the 16th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 1180 of *Lecture Notes in Computer Science*, pages 322–324. Springer-Verlag, 1996.

[BCO93] R. Beigel, R. Chang, and M. Ogiwara. A relationship between difference hierarchies and relativized polynomial hierarchies. *Mathematical Systems Theory*, 26(3):293–310, July 1993.

[Bei91] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Computer Science*, 84(2):199–223, July 1991.

[BF96] Harry Buhrman and Lance Fortnow. Two queries. Unpublished manuscript, September 1996.

[BKS95] R. Beigel, M. Kummer, and F. Stephan. Approximable sets. *Information and Computation*, 120(2):304–314, 1995.

[CK96] R. Chang and J. Kadin. The Boolean hierarchy and the polynomial hierarchy: a closer connection. *SIAM Journal on Computing*, 25(2):340–354, April 1996.

[GM96] W. I. Gasarch and T. McNicholl. Personal communications, 1996.

[HHH96a] E. Hemaspaandra, L. A. Hemaspaandra, and H. Hempel. Downward translation in the polynomial hierarchy. Technical Report TR-630, University of Rochester, Department of Computer Science, July 1996.

[HHH96b] E. Hemaspaandra, L. A. Hemaspaandra, and H. Hempel. Query order in the polynomial hierarchy. Technical Report TR-634, University of Rochester, Department of Computer Science, September 1996.

[HHW95] L. A. Hemaspaandra, H. Hempel, and G. Wechsung. Query order and self-specifying machines. Technical Report TR-596, University of Rochester, Department of Computer Science, October 1995.

[Kad88] J. Kadin. The polynomial time hierarchy collapses if the Boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, December 1988.

[Yap83] C. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26(3):287–300, 1983.