The Computational Complexity Column

Eric Allender
Rutgers University, Department of Computer Science
Piscataway, NJ 08855 USA
allender@cs.rutgers.edu

There are two regular columns in complexity theory; this one and the one that Lane Hemaspaandra edits for SIGACT News. In an effort to avoid overlap in topics, Lane and I exchanged e-mail shortly after I became editor of this column, to find out what articles were planned or were already in the pipeline.

One very pleasant outcome of this e-mail exchange is the following "guest appearance" by Lane and his co-authors. The topic, *Query Order*, is an excellent example of how careful consideration of an overlooked aspect of reducibility (query order) can have consequences in diverse areas of complexity theory. In addition to the surprising list of topics mentioned below, the techniques discussed in this article have been used recently to establish connections among the complexity of several NP-hard approximation problems ([Richard Chang, personal communication], for more information, consult http://umbc.edu/~chang/papers/bqabh).

(On a sad note, recall that last month's column contained a report on the workshop held in celebration of Ron Book's sixtieth birthday. Ron passed away just as the Bulletin was going to press. More details appear elsewhere in this issue of the Bulletin.)

An Introduction to Query Order[1]

Edith Hemaspaandra[2] Lane A. Hemaspaandra[3] Harald Hempel[4]

Abstract: Hemaspaandra, Hempel, and Wechsung [HHW95] raised the following questions: If one is allowed one question to each of two different information sources, does the order in which one asks the questions affect the class of problems that one can solve with the given access? If so, which order yields the greater computational power?

The answers to these questions have been learned—inasfar as they can be learned without resolving whether or not the polynomial hierarchy

[1]Supported in part by grants NSF-CCR-9322513 and NSF-INT-9513368/DAAD-315-PRO-fo-ab.

[2]edith@bamboo.lemoyne.edu. Department of Mathematics, Le Moyne College, Syracuse, NY 13214, USA. Work done in part while visiting Friedrich-Schiller-Universität Jena.

[3]lane@cs.rochester.edu. Department of Computer Science, University of Rochester, Rochester, NY 14627, USA. Work done in part while visiting Friedrich-Schiller-Universität Jena.

[4]hempel@informatik.uni-jena.de. Institut für Informatik, Friedrich-Schiller-Universität Jena, 07740 Jena, Germany. Work done in part while visiting Le Moyne College.

collapses—for both the polynomial hierarchy and the boolean hierarchy. In the polynomial hierarchy, query order never matters. In the boolean hierarchy, query order sometimes does not matter and, unless the polynomial hierarchy collapses, sometimes does matter. Furthermore, the study of query order has yielded dividends in seemingly unrelated areas, such as bottleneck computations and downward translation of equality.

In this article, we present some of the central results on query order. The article is written in such a way as to encourage the reader to try his or her own hand at proving some of these results. We also give literature pointers to the quickly growing set of related results and applications.

1 Introduction

So, you're at a theory conference and the coming session strikes you as potentially boring. You walk into the lobby in search of more coffee and some theoretical chit-chat, and you get more than you bargained for. Poof! A well-dressed stranger appears seemingly from nowhere. His name tag is hidden under a lapel. Under his arm is a stack of books. A second edition of Garey and Johnson? *The Polynomial Hierarchy Does Not Collapse* by M. Sipser and A. Yao? Who is this guy? You don't have a clue. Wait a moment... is that Volume 4 of Knuth under his arm? Aha! Clearly, this is a time-traveller from the distant future!

Stranger: These books will all be yours if you can solve the following small puzzle. Here are two black boxes. One is marked SAT, and in unit time it solves satisfiability (using magical means not widely available in the 20th century). The other black box is marked IEI, and in unit time it solves the well-known problem IntegerExpressionInequivalence (again using magic)—a problem that Stockmeyer and Meyer, in a STOC '73 paper, proved $\mathrm{NP}^{\mathrm{NP}}$-complete. I'll ask you to write a polynomial-time algorithm that accepts the language FOO, and that on each input makes at most one query to each of the black boxes. Actually, at this moment I don't demand that you write the algorithm. I just want you to decide on a fixed order in which you will access the black boxes. Either you must (right now) commit to accessing SAT first on each input, or you must (right now) commit to accessing IEI first on each input.
You: Well, before I commit, will you tell me a bit about what problem FOO is?
Stranger: No.
You: Oh. Wait a second. You're trying to trick me. $\mathrm{NP}^{\mathrm{NP}}$ (as captured by its complete set, IEI) is *much* more powerful than NP (as captured by its complete set, SAT). So for both classes you are speaking of the query to SAT is superfluous. That is: $\mathrm{P}^{\mathrm{IEI}[1]} = \mathrm{P}^{\mathrm{IEI:SAT}} = \mathrm{P}^{\mathrm{SAT:IEI}}$, where "[1]" indicates one query and $\mathrm{P}^{A:B}$ indicates a P machine making at most one query to (the set) A set followed by at most one query to (the set) B.

Stranger: False! Your reasoning is tempting, but is not valid. In fact, that weak SAT query provably gives strictly more computational power, as you would know if you had read a certain *BEATCS* article [HHH97a] and had applied it in light of the Sipser-Yao book under my arm.
You: I've got you now, as I *did* read that obscure little article! I retract the $\mathrm{P}^{\mathrm{IEI}[1]} = \mathrm{P}^{\mathrm{IEI:SAT}}$ claim I just made, but I reassert the $\mathrm{P}^{\mathrm{IEI:SAT}} = \mathrm{P}^{\mathrm{SAT:IEI}}$ claim! Order does not matter here, and so either order I commit to now will leave it equally likely that FOO can be solved via the given access order!
Stranger: Sheesh... I thought no one except the authors and the editor had read that column. Well, you've beaten me so here are your books.
You: But this Sipser-Yao book is from the year 2010. If I just steal their result and proof, and publish them now, won't that change the future and cause the book never to be written, in which case how can I be meeting you and receiving this book in the first place?
Stranger: (Suddenly starts fading away, but before he disappears you hear) I commend to you the Nebula-Award-winning book *Timescape* [Ben80], which also has quite a bit to say about curiosity-driven research, scientific ethics, and...

You suddenly feel a bit disoriented and—distractedly tossing the books into a trash bin and mumbling about swearing off that ninth cup of coffee—you go back into the lecture hall and listen with half-hearted attention to an in-progress lecture on query order.

2 Query Order in the Polynomial Hierarchy

2.1 Results

Recall from the introduction that, for any sets A and B, $\mathrm{P}^{A:B}$ denotes the class of languages that can be accepted via P machines making at most one query to A *followed by* at most one query to B. Similarly, for any classes $\mathcal{C}$ and $\mathcal{D}$, $\mathrm{P}^{\mathcal{C}:\mathcal{D}}$ denotes the class of languages that can be accepted via P machines making at most one query to a set from $\mathcal{C}$ *followed by* at most one query to a set from $\mathcal{D}$. That is,

$$\mathrm{P}^{\mathcal{C}:\mathcal{D}} = \bigcup_{C\in\mathcal{C},\, D\in\mathcal{D}} \mathrm{P}^{C:D}.$$

These notions and notations were introduced by Hemaspaandra, Hempel, and Wechsung [HHW95], who studied them for the case in which $\mathcal{C}$ and $\mathcal{D}$ are levels of the boolean hierarchy (see Section 3); in brief, they proved that query order usually does matter in the boolean hierarchy.

Hemaspaandra, Hemaspaandra, and Hempel [HHH96b,HHH97b], following up on the concept of query order, asked whether query order also matters in the polynomial hierarchy. What they found was that query order *never* matters in the polynomial hierarchy. This is stated formally below as Theorem 2.2.

Definition 2.1 [Sto77]

1. $\Sigma_0^p = \mathrm{P}$.
2. For each $k > 0$, $\Sigma_k^p = \mathrm{NP}^{\Sigma_{k-1}^p}$. (For example, $\Sigma_1^p = \mathrm{NP}$ and $\Sigma_2^p = \mathrm{NP}^{\mathrm{NP}}$.)
3. $\mathrm{PH} = \bigcup_{k \geq 0} \Sigma_k^p$.

Theorem 2.2 [HHH97b] For each $i, j \geq 0$,

$$\mathrm{P}^{\Sigma_i^p : \Sigma_j^p} = \mathrm{P}^{\Sigma_j^p : \Sigma_i^p}.$$

In fact, in all but the "diagonal" cases of this theorem (where order elimination is impossible unless the polynomial hierarchy collapses), one can eliminate order entirely:

Theorem 2.3 [HHH97b] For each $i, j \geq 0$ with $i \neq j$,

$$\mathrm{P}_{1,1\text{-tt}}^{(\Sigma_i^p, \Sigma_j^p)} = \mathrm{P}^{\Sigma_i^p : \Sigma_j^p} = \mathrm{P}^{\Sigma_j^p : \Sigma_i^p},$$

where $\mathrm{P}_{1,1\text{-tt}}^{(\Sigma_i^p, \Sigma_j^p)}$ denotes the class of languages accepted by machines that, in parallel (i.e., simultaneously), ask at most one question to a Σ_i^p set and at most one question to a Σ_j^p set.

However, we should now address the "tempting" worry you, the reader, raised during the introduction. Let $i < j$. Clearly, Σ_j^p can beat Σ_i^p into a pulp. For example, it is well known that $\mathrm{P}^{\Sigma_i^p} \subseteq \Sigma_j^p$. That is, Σ_j^p is more powerful than even a polynomially long series of adaptive queries to Σ_i^p. So it would indeed be tempting to assert: $\mathrm{P}^{\Sigma_i^p : \Sigma_j^p} = \mathrm{P}^{\Sigma_j^p[1]}$ for $i < j$, and indeed there is an obvious (but flawed) "proof" of this, involving having Σ_j^p simulate the Σ_i^p query of $\mathrm{P}^{\Sigma_i^p : \Sigma_j^p}$, get the answer, and then simulate the Σ_j^p query of $\mathrm{P}^{\Sigma_i^p : \Sigma_j^p}$. The flaw is that though Σ_j^p can do this, it cannot pass to the base P machine the information on which truth-table to use to process its answer; there is a 1-bit information bottleneck! Indeed, the tempting equality—$\mathrm{P}^{\Sigma_i^p : \Sigma_j^p} = \mathrm{P}^{\Sigma_j^p[1]}$ for $i < j$—is outright false unless the polynomial hierarchy collapses. This follows immediately from the more general fact that all "ordered access to the polynomial hierarchy" classes are either trivially equal or are truly different (unless the polynomial hierarchy collapses).

Theorem 2.4 [HHH97b] Let $i, j, \ell, m \geq 0$. If $\mathrm{P}^{\Sigma_i^p : \Sigma_j^p} = \mathrm{P}^{\Sigma_\ell^p : \Sigma_m^p}$, then either $\{i, j\} = \{\ell, m\}$ or the polynomial hierarchy collapses.

The just-stated theorem merely concludes that the polynomial hierarchy collapses (unless $\{i, j\} = \{\ell, m\}$). In fact, in almost all cases, the polynomial hierarchy collapses to an alarmingly low level—one that *a priori* seems lower than either of the classes mentioned in the theorem (this can be seen easily from [HHH,BF96,HHH97d,HHH97b], see especially [HHH97b, Section 3.2]). For example,

$$\mathrm{P}^{\Sigma^{\mathrm{p}}_{1997}:\Sigma^{\mathrm{p}}_{1999}} = \mathrm{P}^{\Sigma^{\mathrm{p}}_{1998}:\Sigma^{\mathrm{p}}_{1999}} \Longrightarrow \mathrm{PH} = \Sigma^{\mathrm{p}}_{1999},$$

even though *a priori* one would suspect that $\Sigma^{\mathrm{p}}_{1999}$ is strictly contained in $\mathrm{P}^{\Sigma^{\mathrm{p}}_{1997}:\Sigma^{\mathrm{p}}_{1999}}$.

Note that in all the results we have discussed so far, we have a P machine doing the querying, i.e., $\mathrm{P}^{\mathrm{SAT:IEI}} = \mathrm{P}^{\mathrm{IEI:SAT}}$. In fact, Hemaspaandra et al. [HHH96b] have shown that all standard complexity classes (in particular, all leaf-definable classes) automatically inherit all query-order containments that hold for P machines. Thus, for example, since $\mathrm{P}^{\mathrm{NP:NP^{NP}}} = \mathrm{P}^{\mathrm{NP^{NP}:NP}}$, we may conclude immediately that $\mathrm{PP}^{\mathrm{NP:NP^{NP}}} = \mathrm{PP}^{\mathrm{NP^{NP}:NP}}$.

2.2 Proof by Example

Our goal here is just to give the general flavor of a proof related to query order in the polynomial hierarchy. We will prove part of an instance of Theorem 2.2. That is, we will partially prove:

$$\mathrm{P}^{\Sigma^{\mathrm{p}}_2:\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{NP}:\Sigma^{\mathrm{p}}_2}.$$

In particular, we will prove that $X \subseteq \mathrm{P}^{\mathrm{NP}:\Sigma^{\mathrm{p}}_2}$, where X is the class of languages that are in $\mathrm{P}^{\Sigma^{\mathrm{p}}_2:\mathrm{NP}}$ via a $\mathrm{P}^{\Sigma^{\mathrm{p}}_2:\mathrm{NP}}$ machine in which the P machine on each input asks *exactly* one question to each of its oracles, and in which the P machine accepts if and only if *exactly one of its two queries gets the answer "yes."* That is, we will do the parity case.

The proof is not hard, and finding it for oneself will help one gain a feeling for what it is like to study query order. Thus, we urge the reader to try to prove this him- or herself before reading the proof we include below.

Proof: Let $L \in \mathrm{P}^{\Sigma^{\mathrm{p}}_2:\mathrm{NP}}$. Let M be a P machine, $A \in \Sigma^{\mathrm{p}}_2$, and $B \in \mathrm{NP}$ be such that M accepts L and, on each input, M makes exactly one query to A followed by exactly one query to B. (This does not rule out the possibility that were $M(x)$ to be given an incorrect answer to its first query it would not ask a second query. However, without loss of generality we can assume that it always asks exactly one query to each oracle, regardless of the answer to the first query. We do assume this both here and in the proof of Section 3.2.) We will partially describe a P machine M' that accepts L with one query to an NP set followed by one query to a Σ^{p}_2 set.

On input x, M' determines the first query of $M(x)$ and the two potential second queries of $M(x)$. We will write q for the first query asked by $M(x)$, q_Y for the second query that would be asked by $M(x)$ were it to receive a "yes" answer to the first query, and q_N for the second query that would be asked by $M(x)$ were it to receive a "no" answer to the first

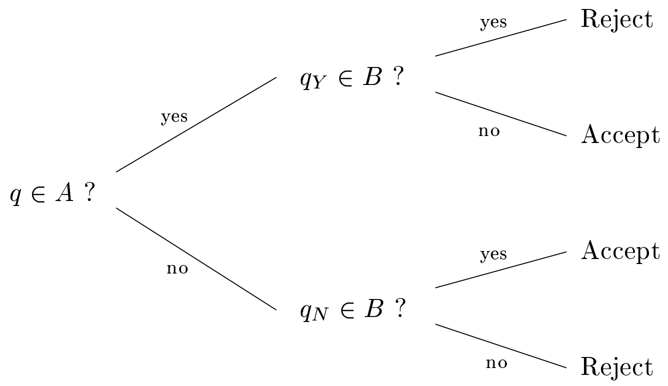

Figure 1: A possible behavior of $M(x)$ (applies to Section 2.2 and Section 3.2).

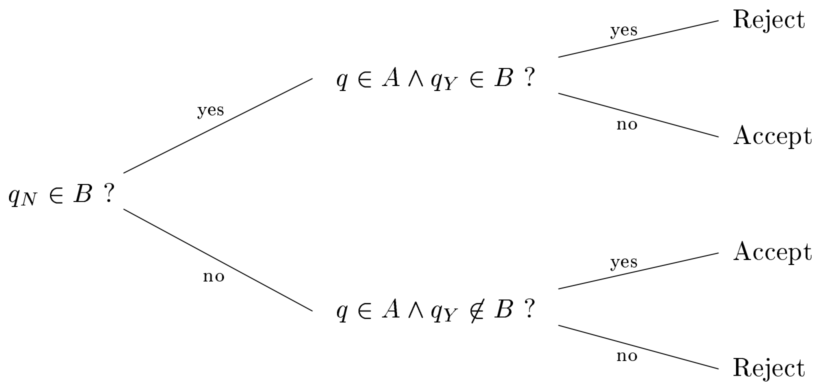

Figure 2: Query structure of the $\mathrm{P}^{\mathrm{NP}:\Sigma_2^{\mathrm{p}}}$ machine corresponding to Figure 1.

query. $M'(x)$ then determines for which of the four possible answers to two sequential queries (namely, "no/no," "no/yes," "yes/no," and "yes/yes") $M(x)$ accepts. All this can be done in polynomial time without querying any strings. Since $M(x)$ asks two queries, there are sixteen different possibilities for Accept/Reject behavior of $M(x)$. As an example, suppose that $M(x)$ accepts if and only if the answer to the first query differs from the answer to the second query (see Figure 1 for a pictorial representation of this case).

In this case, $M'(x)$ proceeds as shown in Figure 2. It is easy to see that M' accepts x if and only if M accepts x. In addition, note that the first query in Figure 2 is a query to an NP set, namely B, and that the two potential second queries "$q \in A \ \wedge \ q_Y \in B$?" and "$q \in A \ \wedge \ q_Y \notin B$?" are both Σ_2^{p} predicates. Since Σ_2^{p} is closed under disjoint union we are done for the case corresponding to Figure 1. There are of course fifteen other cases to consider, but all of these are similar to or easier than the case we just treated. We mention in passing that the full proof of Theorem 2.2 in [HHH97b] is more elegant than working through the sixteen different subcases and splicing them together dynamically; the present proof fragment is merely intended to convey some of the flavor of how one can prove things about query order. ∎

3 Query Order in the Boolean Hierarchy

3.1 Results

The boolean hierarchy [CGH+88,CGH+89] was introduced in the 1980s, and captures and classifies those languages that can be computed via finite hardware operating over NP predicates (equivalently, that can be computed via bounded access to SAT).

Definition 3.1 1. Let $\mathcal{C} \ominus \mathcal{D} = \{L_1 - L_2 \mid L_1 \in \mathcal{C} \wedge L_2 \in \mathcal{D}\}$.

2. [CGH+88]
   (a) $\mathrm{BH}_0 = \mathrm{P}$.
   (b) For each $k > 0$, $\mathrm{BH}_k = \mathrm{NP} \ominus \mathrm{BH}_{k-1}$.
   (c) BH, the boolean hierarchy, is defined as $\bigcup_{k \geq 0} \mathrm{BH}_k$.

So $\mathrm{BH}_1 = \mathrm{NP}$, and BH_2 equals Papadimitriou and Yannakakis's [PY84] class DP, namely the class of all sets that can be written as the intersection of some NP set with some coNP set. NP, DP, and the other levels of the boolean hierarchy contain a large variety of complete problems (see, e.g., [GJ79, CM87,CGH+88,Bor95]).

Hemaspaandra, Hempel, and Wechsung [HHW95] raised the topic of query order by asking whether $\mathrm{P}^{\mathrm{BH}_i:\mathrm{BH}_j} = \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_i}$. They resolved this question as follows. They noted that equality trivially holds if $i = j \vee i = 0 \vee j = 0$. They proved that equality (not so trivially) holds if i is even $\wedge$ $j = i + 1$, or if j is even $\wedge$ $i = j + 1$. They proved that for all other cases inequality holds unless the polynomial hierarchy collapses.

Theorem 3.2 [HHW95] For each i and j, the relationship between $\mathrm{P}^{\mathrm{BH}_i:\mathrm{BH}_j}$ and $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_i}$ is as shown in Figure 3.

The most strikingly odd feature of this theorem is that the just-off-diagonal entries alternate between equality and inequality (e.g., $\mathrm{P}^{\mathrm{BH}_2:\mathrm{BH}_3} = \mathrm{P}^{\mathrm{BH}_3:\mathrm{BH}_2}$, yet unless the polynomial hierarchy collapses $\mathrm{P}^{\mathrm{BH}_3:\mathrm{BH}_4} \neq \mathrm{P}^{\mathrm{BH}_4:\mathrm{BH}_3}$). The curious asymmetry becomes a bit less opaque if one looks at what is actually underpinning Theorem 3.2. The key result on which Theorem 3.2 rests is Lemma 3.3 below, which states that ordered access to the boolean hierarchy's levels can without loss of generality be restructured as parallel access to NP. As is standard, let $\mathrm{R}^{\mathrm{p}}_{\ell\text{-tt}}(\mathrm{NP})$ denote $\{L \mid (\exists A \in \mathrm{NP})[L \leq^{\mathrm{p}}_{\ell\text{-tt}} A]\}$, where $\leq^{\mathrm{p}}_{\ell\text{-tt}}$ is the standard reduction allowing ℓ parallel queries [LLS75].

i

:	etc.	etc.	etc.	etc.	etc.	etc.	etc.	etc.	etc.
7	$=$	$<$	$<$	$<$	$<$	$<$	$=$	$=$	etc.
6	$=$	$<$	$<$	$<$	$<$	$<$	$=$	$=$	etc.
5	$=$	$<$	$<$	$<$	$=$	$=$	$>$	$>$	etc.
4	$=$	$<$	$<$	$<$	$=$	$=$	$>$	$>$	etc.
3	$=$	$<$	$=$	$=$	$>$	$>$	$>$	$>$	etc.
2	$=$	$<$	$=$	$=$	$>$	$>$	$>$	$>$	etc.
1	$=$	$=$	$>$	$>$	$>$	$>$	$>$	$>$	etc.
0	$=$	$=$	$=$	$=$	$=$	$=$	$=$	$=$	etc.
	0	1	2	3	4	5	6	7	··

j

Figure 3: Relationship between $\mathrm{P}^{\mathrm{BH}_i:\mathrm{BH}_j}$ and $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_i}$.

Key: $=$ in row i and column j means $\mathrm{P}^{\mathrm{BH}_i:\mathrm{BH}_j} = \mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_i}$.
$<$ in row i and column j means $\mathrm{P}^{\mathrm{BH}_i:\mathrm{BH}_j}$ is a strict subset of $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_i}$ unless the polynomial hierarchy collapses.
$>$ in row i and column j means $\mathrm{P}^{\mathrm{BH}_i:\mathrm{BH}_j}$ is a strict superset of $\mathrm{P}^{\mathrm{BH}_j:\mathrm{BH}_i}$ unless the polynomial hierarchy collapses.

Lemma 3.3 For each $i, j \geq 1$,

$$\mathrm{P}^{\mathrm{BH}_i:\mathrm{BH}_j} = \begin{cases} \mathrm{R}^{\mathrm{p}}_{(i+2j-1)\text{-tt}}(\mathrm{NP}) & \text{if } i \text{ is even and } j \text{ is odd} \\ \mathrm{R}^{\mathrm{p}}_{(i+2j)\text{-tt}}(\mathrm{NP}) & \text{otherwise.} \end{cases}$$

This lemma is the source of the asymmetry between, for example, 2-versus-3 and 3-versus-4. Of course, this in some way begs the question, as the reader may well ask about the asymmetry between the first and second cases of Lemma 3.3. Briefly and informally put, when i is even and j is odd, a certain underlying graph modeling the computation of $\mathrm{P}^{\mathrm{BH}_i:\mathrm{BH}_j}$ becomes non-bipartite, and by doing so allows one to guarantee a savings of one parallel query to NP (see [HHW95] for full details).

3.2 Proof by Example

In this subsection we will give a partial proof for an instance of Lemma 3.3, namely, we will give a partial proof for

$$\mathrm{P}^{\mathrm{DP}:\mathrm{NP}} \subseteq \mathrm{R}^{\mathrm{p}}_{3\text{-tt}}(\mathrm{NP}).$$

In particular, we will show that $X \subseteq \mathrm{R}^{\mathrm{p}}_{3\text{-tt}}(\mathrm{NP})$, where X is the class of languages that are in $\mathrm{P}^{\mathrm{DP}:\mathrm{NP}}$ via a $\mathrm{P}^{\mathrm{DP}:\mathrm{NP}}$ machine in which the P machine on each input asks *exactly* one question to each of its oracles, and in which the P machine accepts if and only if *exactly one of its two queries gets the* *answer "yes."* That is, as was the case also in Section 2.2, we will do the parity case.

We warn the reader that the proof approach taken here is not suited to be elegantly generalized to eventually yield Lemma 3.3. For a complete and unified proof of $\mathrm{P}^{\mathrm{DP:NP}} \subseteq \mathrm{R}^{\mathrm{p}}_{3\text{-tt}}(\mathrm{NP})$, we refer the reader to [HHW95]. However, the proof given below provides a good starting point for understanding how these proofs work in the context of the boolean hierarchy. Note that the proof picks one of the interesting cases of Lemma 3.3 ("i even and j odd"). As hands-on experience is the best way to get a feel for an area, we urge the reader to come up with his or her own proof before reading the proof below.

Proof: Let $L \in \mathrm{P}^{\mathrm{DP:NP}}$. Let P machine M, $A \in \mathrm{DP}$, and $B \in \mathrm{NP}$ be such that M accepts L and, on each input, M makes exactly one query to A followed by exactly one query to B. We will partially describe a P machine M' that accepts L with three parallel queries to an NP set.

On input x, M' determines the first query and the two potential second queries of $M(x)$. As in the proof of Section 2.2, we will write q for the first query asked by $M(x)$, q_Y for the second query that would be asked by $M(x)$ were it to receive a "yes" answer to the first query, and q_N for the second query that would be asked by $M(x)$ were it to receive a "no" answer to the first query. $M'(x)$ then determines for which answers to its two queries $M(x)$ accepts. All this can be done in polynomial time without querying any strings. Again, we will consider the case pictured in Figure 1, i.e., the case that M accepts x if and only if the answer to the first query differs from the answer to the second query.

Let $A_1, A_2 \in \mathrm{NP}$ be such that $A = A_1 - A_2$. It is well-known that we may choose A_1 and A_2 to be such that $A_1 \supseteq A_2$ [CGH+88]. If we know the answers to the *four* NP queries, $q \in A_1$, $q \in A_2$, $q_N \in B$, and $q_Y \in B$, we can easily determine whether $M(x)$ accepts or rejects. Thus we have $\mathrm{P}^{\mathrm{DP:NP}} \subseteq \mathrm{R}^{\mathrm{p}}_{4\text{-tt}}(\mathrm{NP})$. How can we do better? In particular, how can we save one parallel NP query?

Let us redraw the query tree of $M(x)$ using the above-mentioned underlying four queries in the fashion shown in Figure 4. Recall that we are looking at the case in which M accepts x if and only if the answer to the first query differs from the answer to the second query. Since $A_1 \supseteq A_2$, if $q \notin A_1$ then $q \notin A$.

The refined query tree of $M(x)$ displays four regions of acceptance and rejection (see Figure 4). In order to correctly simulate $M(x)$, it suffices to find out in which region the correct branch ends. However, this can be done with just three questions, namely:

1. Does the correct branch end in region 2, 3, or 4? ($q \in A_1 \ \vee \ q_N \in B$?)

2. Does it end in region 3 or 4? ($q \in A_1 \ \wedge \ (q \in A_2 \ \vee \ q_Y \in B)$?)

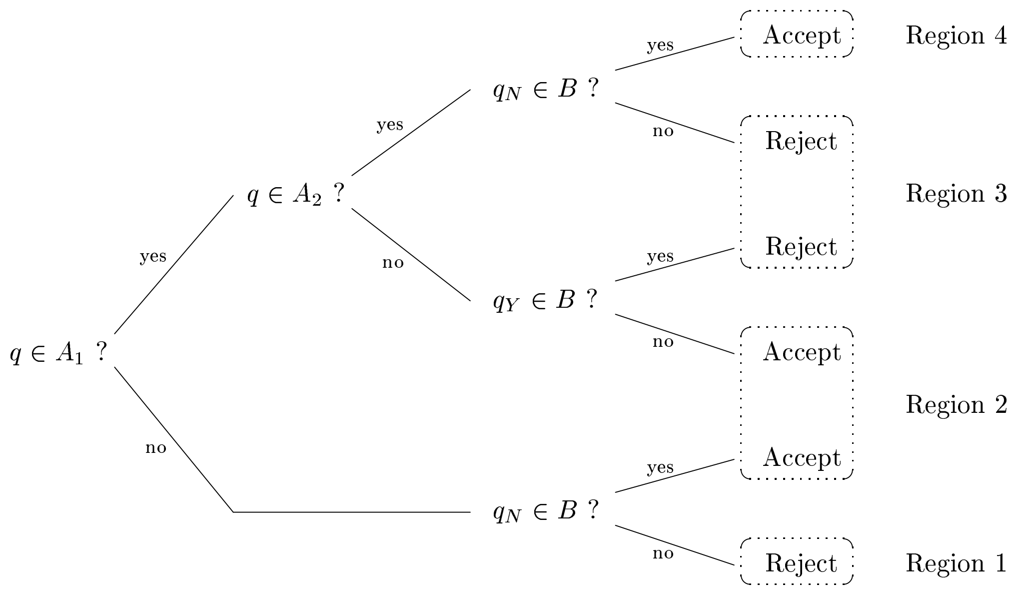

Figure 4: Refined query tree of $M(x)$—all queries drawn are NP queries.

3. Does the correct branch end in region 4? ($q \in A_1 \ \wedge \ q \in A_2 \ \wedge \ q_N \in B$?)

The answers to these three questions determine the region in which the correct branch ends and hence we know whether $M(x)$ rejects or accepts. In particular, $M'(x)$ should accept if and only if the correct branch ends either in region 2 or region 4 (that is, if and only if either only question (1) is answered "yes" or all three questions are answered "yes"). Note that we use three different NP sets and also various "and"s and "or"s in the above description of the questions, but since NP is closed under union, intersection, and disjoint union, the three questions can be transformed (in polynomial time) into three single queries that in turn can be asked (in parallel) to one NP set. ∎

4 Related Work

Sections 2 and 3 presented the basic results known about query order in the polynomial and boolean hierarchies. In a nutshell, query order never matters in the polynomial hierarchy, and in the boolean hierarchy we know in exactly which cases query order matters (assuming that the polynomial hierarchy does not collapse).

However, via the study of query order, a number of results have been obtained regarding topics that at first blush might seem totally unrelated, such as bottleneck computation and downward translation of equality. Also, a number of researchers have generalized from "one query to a given class" to more elaborate settings such as tree-like query structures, multiple queries, and multiple rounds of multiple queries. In this section, we briefly provide pointers to these related topics and generalized settings.

Translating Equalities Downwards

Suppose two questions to Σ_k^{p} yield no new languages beyond those already solvable via one query to Σ_k^{p}. What follows?

Until very recently, all one could conclude from this assumption was that the polynomial hierarchy collapses to a level slightly above one query to $\Sigma_{k+1}^{\mathrm{p}}$ (note the "+1" here) [Kad88,Wag87,Wag89,CK96,BCO93]. However, work growing directly out of the study of query-order classes—namely, out of the goal of showing that different ordered access to levels of the polynomial hierarchy yields different language classes (see Theorem 2.4)—led to a collapse a full level lower in the polynomial hierarchy. In particular, the one-two punch of [HHH96a,HHH]/[BF96] yielded the following theorem.

Theorem 4.1 Let $k > 1$. If $\mathrm{P}^{\Sigma_k^{\mathrm{p}}[1]} = \mathrm{P}^{\Sigma_k^{\mathrm{p}}[2]}$, then $\mathrm{PH} = \Sigma_k^{\mathrm{p}}$.

In other words, if $\mathrm{P}^{\Sigma_k^{\mathrm{p}}[1]} = \mathrm{P}^{\Sigma_k^{\mathrm{p}}[2]}$, $k > 1$, the polynomial hierarchy crashes to a class that (before the crash) was seemingly even lower than that at which the hypothesis's equality stands.

It has been shown [HHH,HHH97d] that the above theorem in fact has an analog for the j versus $j+1$ queries case. In particular, we have the following, which like Theorem 4.1 was established in the literature via proving a more general theorem about query-order classes, and then deriving the stated result as a corollary to the more general theorem [HHH97d].

Theorem 4.2 Let $k > 1$ and $m \geq 1$. If $\mathrm{P}^{\Sigma_k^{\mathrm{p}}[m]} = \mathrm{P}^{\Sigma_k^{\mathrm{p}}[m+1]}$, then $\mathrm{DIFF}_m(\Sigma_k^{\mathrm{p}}) = \mathrm{coDIFF}_m(\Sigma_k^{\mathrm{p}})$.

Again, this says that, under the stated assumption, there is a collapse within the boolean hierarchy to a level that, *a priori*, was just *below* $\mathrm{P}^{\Sigma_k^{\mathrm{p}}[m]}$.

In a nutshell, in this setting smaller classes collapse if and only if larger classes collapse—a type of behavior people have been stalking ever since influential papers of Book [Boo74] and Hartmanis, Immerman, and Sewelson [Har83,HIS85] raised the issue of when classes stand and fall together.

Multiple Queries and Bottleneck Computations

In this survey, we have focused on the most natural case: one query to each of the two information sources. A number of papers building on those mentioned here have studied more elaborate settings.

In fact, the initial paper of Hemaspaandra, Hempel, and Wechsung [HHW95] already studied the case of general tree-like access to levels of the boolean hierarchy, and in doing so studied the case of multiple rounds of single queries; Beigel and Chang [BC97] study multiple rounds of multiple queries to the polynomial hierarchy, and show that here the order does not matter, and they also study the case of function classes; Wagner [Wag97] studies parallel rounds of one or more queries to the polynomial hierarchy and other classes and also tightly relates such classes to the refined hierarchy work of Selivanov [Sel94,Sel95] (see also the discussion in the final paragraph of Section 2 of [HHH97c]).

In a quite different direction, bottleneck machines are a model used to study whether a computational problem can be decomposed into a large number of simple, sequential, tasks, each of which passes on only a very limited amount of information to the next task, and all of which differ only in that input and in a "task number" input [CF91]. A surprising recent paper of Hertrampf [Her97] uses ordered access involving multiple queries, combined with quantifier-based and modulo-based computation, to completely characterize the languages accepted by certain bottleneck machine classes—classes that had long eluded crisp characterization.

Advice Classes, Self-Specifying Machines, and Completeness Types

A number of other seemingly different notions are related to query order. Hemaspaandra, Hempel, and Wechsung [HHW97] have studied self-specifying machines—nondeterministic machines that dynamically specify the path sets on which they will accept. They completely characterize the two most natural such classes in terms of query-order classes with a "positive final query" restriction. They show that the classes have equivalent characterizations as the #P-closures of P and NP, respectively, and they establish a query order result mixing function and language classes: $\mathrm{P}^{\#\mathrm{P}[1]} = \mathrm{P}^{\#\mathrm{P}:\mathrm{NP}} \iff \mathrm{P}^{\#\mathrm{P}[1]} = \mathrm{P}^{\#\mathrm{P}:\mathrm{NP}[\mathcal{O}(1)]}$ (where "$\mathcal{C} : \mathcal{D}$" access means one query each except when $\mathcal{O}(1)$ queries are explicitly stated for the queried class). They also show that the classes have characterizations in terms of the "input-specific advice" notation of Köbler and Thierauf [KT94].

Agrawal, Beigel, and Thierauf [ABT96], independently of [HHW95], also study input-specific-advice classes. As noted by Hemaspaandra et al. in the journal version of [HHW95], this can be seen as equivalent to studying query order with a "positive final query" restriction—i.e., the machines must "do" exactly what the response to their second query is. A detailed and careful discussion of the relationship between the two papers can be found in the journal version of [HHW95].

Finally, a long line of research has asked whether $\leq_{\mathrm{m}}^{\mathrm{p}}$-completeness and $\leq_{\mathrm{T}}^{\mathrm{p}}$-completeness stand or fall together for classes that potentially lack complete sets. Gurevich [Gur83] and Ambos-Spies [Amb86] have shown that, for all classes $\mathcal{C}$ closed downwards under Turing reductions, it holds robustly that: $\mathcal{C}$ has $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete sets if and only if $\mathcal{C}$ has $\leq_{\mathrm{T}}^{\mathrm{p}}$-complete sets. Nonetheless, by studying a strong nondeterministic closure of NP that, it turns out, exactly equals the query-order class $\mathrm{P}^{\mathrm{NP}\cap\mathrm{coNP}:\mathrm{NP}}$, Hemaspaandra et al. have recently shown that on some reducibility closures of NP, $\leq_{\mathrm{m}}^{\mathrm{p}}$-completeness and $\leq_{\mathrm{T}}^{\mathrm{p}}$-completeness do not robustly stand or fall together [HHH97c].

References

[ABT96] M. Agrawal, R. Beigel, and T. Thierauf. Modulo information from nonadaptive queries to NP. Technical Report 96-001, Electronic Colloquium on Computational Complexity, January 1996. Appears as "Pinpointing Computation with Modular Queries in the Boolean Hierarchy" in the 16th FSTTCS conference.

[Amb86] K. Ambos-Spies. A note on complete problems for complexity classes. *Information Processing Letters*, 23:227–230, 1986.

[BC97] R. Beigel and R. Chang. Commutative queries. In *Proceedings of the 5th Israeli Symposium on Theory of Computing and Systems*, pages 159–165. IEEE Computer Society Press, June 1997.

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

[Ben80] G. Benford. *Timescape.* Simon and Schuster, 1980.

[BF96] H. Buhrman and L. Fortnow. Two queries. Technical Report 96-20, University of Chicago, Department of Computer Science, Chicago, IL, September 1996.

[Boo74] R. Book. Tally languages and complexity classes. *Information and Control*, 26:186–193, 1974.

[Bor95] B. Borchert. The complexity of mind changes. Research Report 3, Universität Heidelberg, Mathematisches Institut, Heidelberg, Germany, December 1995.

[CF91] J. Cai and M. Furst. PSPACE survives constant-width bottlenecks. *International Journal of Foundations of Computer Science*, 2(1):67–76, 1991.

[CGH+88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, 1988.

[CGH+89] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy II: Applications. *SIAM Journal on Computing*, 18(1):95–111, 1989.

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

[CM87] J. Cai and G. Meyer. Graph minimal uncolorability is D^P-complete. *SIAM Journal on Computing*, 16(2):259–277, 1987.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[Gur83] Y. Gurevich. Algebras of feasible functions. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 210–214. IEEE Computer Society Press, November 1983.

[Har83] J. Hartmanis. On sparse sets in NP−P. *Information Processing Letters*, 16:55–60, 1983.

[Her97] U. Hertrampf. Acceptance by transformation monoids (with an application to local self-reductions). In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, pages 213–224. IEEE Computer Society Press, June 1997.

[HHH] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. A downward collapse within the polynomial hierarchy. *SIAM Journal on Computing.* To appear.

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

[HHH96b] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. Query order in the polynomial hierarchy. Technical Report TR-634, University of Rochester, Department of Computer Science, Rochester, NY, September 1996. Revised, August 1997.

[HHH97a] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. An introduction to query order. *Bulletin of the EATCS*, 63, October 1997.

[HHH97b] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. Query order in the polynomial hierarchy. In *Proceedings of the 11th Conference on Fundamentals of Computation Theory*, pages 222–232. Springer Verlag *Lecture Notes in Computer Science #1279*, September 1997.

[HHH97c] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. $\mathrm{R}^{SN}_{1\text{-}tt}(\mathrm{NP})$ distinguishes robust many-one and Turing completeness. In *Proceedings of the 3rd Italian Conference on Algorithms and Complexity*, pages 49–60. Springer-Verlag *Lecture Notes in Computer Science #1203*, March 1997.

[HHH97d] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. Translating equality downwards. Technical Report TR-657, University of Rochester, Department of Computer Science, Rochester, NY, April 1997.

[HHW95] L. Hemaspaandra, H. Hempel, and G. Wechsung. Query order and self-specifying machines. Technical Report TR-596, University of Rochester, Department of Computer Science, Rochester, NY, October 1995. To appear as "Query Order" in *SIAM Journal on Computing*.

[HHW97] L. Hemaspaandra, H. Hempel, and G. Wechsung. Self-specifying machines. Technical Report TR-654, University of Rochester, Department of Computer Science, Rochester, NY, April 1997.

[HIS85] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP−P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):159–181, 1985.

[Kad88] J. Kadin. The polynomial time hierarchy collapses if the boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, 1988. Erratum appears in the same journal, 20(2):404.

[KT94] J. Köbler and T. Thierauf. Complexity-restricted advice functions. *SIAM Journal on Computing*, 23(2):261–275, 1994.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.

[PY84] C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984.

[Sel94] V. Selivanov. Two refinements of the polynomial hierarchy. In *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 439–448. Springer-Verlag *Lecture Notes in Computer Science #775*, February 1994.

[Sel95] V. Selivanov. Fine hierarchies and boolean terms. *Journal of Symbolic Logic*, 60(1):289–317, 1995.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Wag87] K. Wagner. Number-of-query hierarchies. Technical Report 158, Universität Augsburg, Institut für Mathematik, Augsburg, Germany, October 1987.

[Wag89] K. Wagner. Number-of-query hierarchies. Technical Report 4, Universität Würzburg, Institut für Informatik, Würzburg, Germany, February 1989.

[Wag97] K. Wagner. A note on parallel queries and the difference hierarchy. Technical Report 173, Universität Würzburg, Institut für Informatik, Würzburg, Germany, June 1997.