

ON THE STRUCTURE OF BOUNDED QUERIES TO ARBITRARY NP SETS*

RICHARD CHANG[†]

Abstract. Kadin [6] showed that if the Polynomial Hierarchy (PH) has infinitely many levels, then for all k , $\text{P}^{\text{SAT}[k]} \subset \text{P}^{\text{SAT}[k+1]}$. This paper extends Kadin’s technique and shows that a proper query hierarchy is not an exclusive property of NP complete sets. In fact, for any $A \in \text{NP} - \widehat{\text{low}}_3$, if PH has infinitely many levels, then $\text{P}^{A[k]} \subset \text{P}^{A[k+1]}$. Moreover, for the case of parallel queries, $\text{P}^{A\parallel[k+1]}$ is not even contained in $\text{P}^{\text{SAT}\parallel[k]}$. These same techniques can be used to explore some other questions about query hierarchies. For example, if there exists any A such that $\text{P}^{A[2]} = \text{P}^{\text{SAT}[1]}$, then PH collapses to Δ_3^{P} .

Key words. Polynomial-time Hierarchy, Boolean Hierarchy, bounded queries, sparse sets, non-uniform computation.

AMS(MOS) subject classifications. 68Q15, 03D15, 03D20

1. Introduction. Soon after the Boolean Hierarchy (BH) was defined, the field of computational complexity theory experienced an exciting period when many hierarchies previously believed to be proper were shown to have collapsed [5]. These collapses led many to wonder if BH might also collapse. However, Kadin [6] brought some respect to the Boolean Hierarchy by showing that if BH collapses then so does the Polynomial Hierarchy (PH). Thus, if the Polynomial Hierarchy has infinitely many levels, then so would the Boolean Hierarchy and the intertwined query hierarchies.

The Boolean Hierarchy and the Query Hierarchy (QH) are built on top of SAT and $\overline{\text{SAT}}$. These hierarchies are contained in the Δ_2^{P} level of the Polynomial Hierarchy and their basic properties have been fully explored [1, 2, 6, 13]. BH is defined by combining SAT and $\overline{\text{SAT}}$ with logical “and” and “or”. QH is defined by allowing any polynomial time computable combination of SAT and $\overline{\text{SAT}}$.

In this paper, we look at hierarchies built on top of arbitrary sets in NP. We want to know when these hierarchies are proper, and how they relate to BH and QH. We will study these questions in the setting of Schöning’s high-low sets and produce some results about these hierarchies similar to ones already known about BH and QH. In particular, we conclude that if a set is high and the query hierarchy over this set collapses, then PH collapses. We also show that if PH is infinite, then bounded query hierarchies built from many languages in NP are also proper. (By “many”, we mean all sets in $\text{NP} - \widehat{\text{low}}_3$.) We interpret this as evidence that a proper query hierarchy is not an exclusive property of NP-hard sets. To demonstrate this claim, we show that if PH is infinite, then we can construct a language B in NP so that for all k , $\text{P}^{B[k+1]} \not\subseteq \text{P}^{\text{SAT}[k]}$. Moreover, this language B is not a high set, so it is not hard for NP in any sense (e.g., not NP-hard under many-one, Turing, or strong nondeterministic reductions).

2. The Boolean Hierarchy and the Query Hierarchy. In this section, we define the Boolean and Query Hierarchies and summarize some previous work in this area. This is neither a complete nor a chronological summary. We encourage the reader to consult the literature for completeness [1, 2, 3, 6, 13].

* This research was supported in part by by NSF research grants DCR-8520597 and CCR-88-23053.

[†] Department of Computer Science, Cornell University, Ithaca, New York 14853.

We assume that the reader is familiar with P, NP, PH and oracle Turing machines. We quickly define some familiar notation and concepts. For any language A , we write $A^{=n}$ for the set of strings in A of length n . Recall that a set S is *sparse* if $\|S^{=n}\|$ is bounded by a polynomial. A *padded prefix* of a string x is a string w such that $|w| = |x|$, $w = y\#^k$ and y is a prefix of x . A set S is *self-p-printable* if there is a polynomial time oracle Turing machine, D_i^S , which prints out $S^{=n}$ on input 1^n . All self-p-printable sets are sparse. A set A is closed under padded prefixes if $w \in A$ whenever $x \in A$ and w is a padded prefix of x . If a set is sparse and closed under padded prefixes, it is self-p-printable.

We now define the Parallel Query Hierarchy (QH_{\parallel}). This hierarchy is defined by restricting a P^{SAT} machine's access to the SAT oracle in two ways. First, the number of queries is limited to a constant (i.e., does not depend on input length). Second, all the queries must be made at the same time. We allow the P base machine to do some computation to determine what the queries are, but we do not allow the query strings to depend on the results (oracle answers) of previous queries. We write $\text{P}^{\text{SAT}}_{\parallel}[k]$ for the class of languages accepted by polynomial time Turing machines that make at most k parallel queries to SAT on input strings of any length. Of course,

$$\text{P}^{\text{SAT}}_{\parallel}[1] \subseteq \text{P}^{\text{SAT}}_{\parallel}[2] \subseteq \dots \subseteq \text{P}^{\text{SAT}}_{\parallel}[k] \subseteq \text{P}^{\text{SAT}}_{\parallel}[k+1] \subseteq \dots$$

This nested sequence has the upward collapsing property, in the sense that if $\text{P}^{\text{SAT}}_{\parallel}[k] = \text{P}^{\text{SAT}}_{\parallel}[k+1]$, then the entire Parallel Query Hierarchy ($\text{QH}_{\parallel} = \bigcup_{j=1}^{\infty} \text{P}^{\text{SAT}}_{\parallel}[j]$) is contained in $\text{P}^{\text{SAT}}_{\parallel}[k]$. If PH has infinitely many levels, then QH_{\parallel} does too [6]:

$$\text{P}^{\text{SAT}}_{\parallel}[1] \subset \text{P}^{\text{SAT}}_{\parallel}[2] \subset \dots \subset \text{P}^{\text{SAT}}_{\parallel}[k] \subset \text{P}^{\text{SAT}}_{\parallel}[k+1] \subset \dots$$

When we remove the parallel query restriction from $\text{P}^{\text{SAT}}_{\parallel}[k]$ and allow subsequent queries to depend on answers to previous queries, we obtain the serial Query Hierarchy. We write $\text{P}^{\text{SAT}}[k]$ for the class of languages accepted by polynomial time Turing machines which ask at most k serial queries to SAT for input strings of any length, and QH for $\bigcup_{j=1}^{\infty} \text{P}^{\text{SAT}}[j]$. Beigel [1] showed by a clever binary search routine (the mind change technique) that

$$\text{P}^{\text{SAT}}_{\parallel}[2^k - 1] = \text{P}^{\text{SAT}}[k].$$

Thus, the levels of QH are simply levels of QH_{\parallel} exponentially far apart. Serial queries are generally considered more natural and relevant than parallel queries, but the finer structure of QH_{\parallel} makes it more amenable to analysis.

Next, we examine the Boolean Hierarchy [2]. Like PH, each level of BH is composed of two complementary language classes BH_k and co-BH_k , such that

$$\text{co-BH}_k = \{ L \mid \overline{L} \in \text{BH}_k \}.$$

BH_k is defined inductively, starting with NP and building up with unions and intersections, as follows:

$$\begin{aligned} \text{BH}_1 &\stackrel{\text{def}}{=} \text{NP} \\ \text{BH}_{2k} &\stackrel{\text{def}}{=} \{ L_1 \cap \overline{L_2} \mid L_1 \in \text{BH}_{2k-1} \text{ and } L_2 \in \text{NP} \} \\ \text{BH}_{2k+1} &\stackrel{\text{def}}{=} \{ L_1 \cup L_2 \mid L_1 \in \text{BH}_{2k} \text{ and } L_2 \in \text{NP} \}. \end{aligned}$$

The levels of BH are interlaced, much like PH:

$$\begin{aligned} \text{BH}_k &\subseteq \text{BH}_{k+1} \cap \text{co-BH}_{k+1} \subseteq \text{BH}_{k+1} \\ \text{co-BH}_k &\subseteq \text{BH}_{k+1} \cap \text{co-BH}_{k+1} \subseteq \text{co-BH}_{k+1}. \end{aligned}$$

BH also has the upward collapsing property:

$$\text{BH}_k = \text{co-BH}_k \implies \text{BH} = \text{BH}_k \cap \text{co-BH}_k.$$

Many results in this area depend on the fact that BH and QH_{\parallel} are intertwined [1, 2]

$$\text{BH}_k \cup \text{co-BH}_k \subseteq \text{P}^{\text{SAT}_{\parallel}[k]} \subseteq \text{BH}_{k+1} \cap \text{co-BH}_{k+1} \subseteq \text{P}^{\text{SAT}_{\parallel}[k+1]}.$$

Clearly BH is a proper hierarchy if and only if QH_{\parallel} is a proper hierarchy. Kadin showed that if BH collapses, then PH collapses. His proof depends on the structure of the canonical complete languages for the levels of BH. For BH_2 and BH_3 these complete languages are:

$$\begin{aligned} \text{SAT} \wedge \overline{\text{SAT}} &\stackrel{\text{def}}{=} \{ \langle F_1, F_2 \rangle \mid F_1 \in \text{SAT} \text{ and } F_2 \in \overline{\text{SAT}} \}, \\ (\text{SAT} \wedge \overline{\text{SAT}}) \vee \text{SAT} &\stackrel{\text{def}}{=} \{ \langle F_1, F_2, F_3 \rangle \mid \langle F_1, F_2 \rangle \in \text{SAT} \wedge \overline{\text{SAT}} \text{ or } F_3 \in \text{SAT} \}. \end{aligned}$$

3. High and Low Sets. Schöning [10] defined the high and low hierarchies for NP to classify sets between P and NP. Very roughly, the high-low classification measures the amount of information an NP language (acting as an oracle) can give to a Σ_k^{P} machine. If $A \in \text{NP}$, then for all k

$$\Sigma_k^{\text{P}} \subseteq \Sigma_k^{\text{P},A} \subseteq \Sigma_k^{\text{P},\text{SAT}} = \Sigma_{k+1}^{\text{P}}.$$

If $\Sigma_k^{\text{P},A} = \Sigma_k^{\text{P},\text{SAT}}$, then one might say that the oracle A tells the Σ_k^{P} base machine a lot—as much as SAT does. If $\Sigma_k^{\text{P}} = \Sigma_k^{\text{P},A}$, then one could say that A does not tell the Σ_k^{P} base machine anything that it could not compute itself. In the first case we call A a high set, in the latter a low set. More formally, we define:

$$\begin{aligned} \text{high}_k &\stackrel{\text{def}}{=} \{ A \mid A \in \text{NP} \text{ and } \Sigma_k^{\text{P},A} = \Sigma_k^{\text{P},\text{SAT}} \}, \\ \text{low}_k &\stackrel{\text{def}}{=} \{ A \mid A \in \text{NP} \text{ and } \Sigma_k^{\text{P}} = \Sigma_k^{\text{P},A} \}. \end{aligned}$$

Clearly, $\text{low}_k \subseteq \text{low}_{k+1}$ and $\text{high}_k \subseteq \text{high}_{k+1}$. However, the high and low hierarchies are not known to have the familiar upward collapsing behavior. For example, it is not known whether $\text{low}_2 = \text{low}_3$ would imply that $\text{low}_2 = \text{low}_4$.

Schöning also defined a refinement of the low hierarchy [11]

$$\widehat{\text{low}}_k \stackrel{\text{def}}{=} \{ A \mid A \in \text{NP} \text{ and } \Delta_k^{\text{P}} = \Delta_k^{\text{P},A} \}.$$

These classes lie between the levels of the low hierarchy,

$$\text{low}_0 \subseteq \widehat{\text{low}}_1 \subseteq \text{low}_1 \subseteq \widehat{\text{low}}_2 \subseteq \text{low}_2 \subseteq \widehat{\text{low}}_3 \subseteq \text{low}_3$$

and allow a finer classification of language classes in the low hierarchy. In this paper, we will pay special attention to the class $\widehat{\text{low}}_3$.

High and low sets have many interesting properties. We briefly mention some here. Again, we ask the reader to consult the literature for the proofs [7, 9, 10, 11, 12].

1. $\text{low}_k = \text{high}_k \iff \text{PH} \subseteq \Sigma_k^{\text{P}}$.
2. If PH is infinite, then $\exists I \in \text{NP} \forall k [I \notin \text{high}_k \text{ and } I \notin \text{low}_k]$.
3. If S is sparse and $S \in \text{NP}$, then $S \in \widehat{\text{low}}_2$.
4. If $A \in \text{NP}$ and for some sparse set S , $A \in \text{P}^S$, then $A \in \text{low}_3$.
5. $\text{R} \subseteq \text{BPP} \cap \text{NP} \subseteq \text{low}_2$.
6. $\text{high}_0 = \{ A \mid A \text{ is } \leq_{\text{T}}^{\text{P}}\text{-complete for NP} \}$.
7. $\text{low}_0 = \text{P}$.
8. $\text{low}_1 = \text{NP} \cap \text{co-NP}$.
9. Graph Isomorphism $\in \text{low}_2$.

4. Main Theorem. The three hierarchies QH, QH_{\parallel} , and BH are built on top of SAT and $\overline{\text{SAT}}$. We now consider analogous query hierarchies built on top of an arbitrary set in NP. For any set $A \in \text{NP}$ we can consider the classes $\text{P}^{A\parallel[k]}$ and $\text{P}^{A[k]}$, defined analogously to $\text{P}^{\text{SAT}\parallel[k]}$ and $\text{P}^{\text{SAT}[k]}$. We immediately have

$$\text{P}^{A\parallel[k]} \subseteq \text{P}^{A\parallel[k+1]}, \quad \text{P}^{A[k]} \subseteq \text{P}^{A[k+1]} \quad \text{and} \quad \text{P}^{A\parallel[k]} \subseteq \text{P}^{A[k]}.$$

However, in general we do not know how to repeat Beigel's mind change technique, so we can only relate parallel and serial queries loosely

$$\text{P}^{A\parallel[k]} \subseteq \text{P}^{A[k]} \subseteq \text{P}^{A\parallel[2^k-1]}.$$

The serial query hierarchy over an arbitrary set has the upward collapsing property, too. That is,

$$\text{P}^{A[k+1]} = \text{P}^{A[k]} \implies \forall j > k, \quad \text{P}^{A[j]} = \text{P}^{A[k]}.$$

However, it is not known if the corresponding property holds for the parallel query hierarchy over A . Because of these difficulties, we do not attempt to define a boolean hierarchy based on $A \in \text{NP}$. Instead we define *boolean languages* analogous to the canonical complete languages for BH_k and co-BH_k .

DEFINITION 4.1. For each language A we define a sequence of boolean languages

$$\begin{aligned} \text{BL}_1(A) &\stackrel{\text{def}}{=} A \\ \text{BL}_{2k}(A) &\stackrel{\text{def}}{=} \{ \langle x_1, \dots, x_{2k} \rangle \mid \langle x_1, \dots, x_{2k-1} \rangle \in \text{BL}_{2k-1}(A) \text{ and } x_{2k} \in \overline{A} \} \\ \text{BL}_{2k+1}(A) &\stackrel{\text{def}}{=} \{ \langle x_1, \dots, x_{2k+1} \rangle \mid \langle x_1, \dots, x_{2k} \rangle \in \text{BL}_{2k}(A) \text{ or } x_{2k+1} \in A \} \\ \text{co-BL}_1(A) &\stackrel{\text{def}}{=} \overline{A} \\ \text{co-BL}_{2k}(A) &\stackrel{\text{def}}{=} \{ \langle x_1, \dots, x_{2k} \rangle \mid \langle x_1, \dots, x_{2k-1} \rangle \in \text{co-BL}_{2k-1}(A) \text{ or } x_{2k} \in A \} \\ \text{co-BL}_{2k+1}(A) &\stackrel{\text{def}}{=} \{ \langle x_1, \dots, x_{2k+1} \rangle \mid \langle x_1, \dots, x_{2k} \rangle \in \text{co-BL}_{2k}(A) \text{ and } x_{2k+1} \in \overline{A} \}. \end{aligned}$$

Note that $\text{BL}_2(\text{SAT}) = \text{SAT} \wedge \overline{\text{SAT}}$ and $\text{BL}_3(\text{SAT}) = (\text{SAT} \wedge \overline{\text{SAT}}) \vee \text{SAT}$. In the general case, $\text{BL}_k(\text{SAT})$ is $\leq_{\text{m}}^{\text{P}}$ -complete for BH_k and $\text{co-BL}_k(\text{SAT})$ is $\leq_{\text{m}}^{\text{P}}$ -complete for co-BH_k . Clearly, $\text{BL}_k(A) \in \text{P}^{A\parallel[k]}$ and $\text{co-BL}_k(A) \in \text{P}^{A\parallel[k]}$. So, there is some interaction between the boolean languages over A and the query hierarchies over A .

With all these definitions in place we can pose some questions originally posed for SAT. For example, we would like to know what happens if $\text{P}^{A\parallel[k]} = \text{P}^{A\parallel[k+1]}$, if $\text{P}^{A[k]} = \text{P}^{A[k+1]}$, or if $\text{BL}_k(A) \leq_{\text{m}}^{\text{P}} \text{co-BL}_k(A)$. Also, we would like to know the relationship between queries to SAT and queries to A . We know $\text{P}^{A\parallel[k]} \subseteq \text{P}^{\text{SAT}\parallel[k]}$, but what is

the relationship between $P^{A\| [k]}$ and $P^{\text{SAT}\| [k-1]}$? Instead of asking “Is one query to SAT as powerful as two?” we ask “Is one query to SAT as powerful as two queries to some other oracle?” The following theorem answers some of these questions.

THEOREM 4.2. *If $A, B \in \text{NP}$ and $\text{BL}_k(A) \leq_m^P \text{co-BL}_k(B)$, then $A \in \widehat{\text{low}}_3$.*

We prove this theorem in two parts. Lemma 4.3 is a rather technical lemma, and follows the same lines as Kadin’s original proof that $\text{BL}_k(\text{SAT}) \leq_m^P \text{co-BL}_k(\text{SAT})$ implies $\text{PH} \subseteq \Delta_3^P$. We relegate the proof of Lemma 4.3 to the next section.

LEMMA 4.3. *If $B \in \text{NP}$ and $\text{BL}_k(A) \leq_m^P \text{co-BL}_k(B)$, then there exists a self-p-printable set $S \in \Delta_3^P$ such that $\overline{A} \in \text{NP}^S$. (N.B. In Lemma 4.3, we do not assume $A \in \text{NP}$.)*

LEMMA 4.4. *If $A \in \text{NP}$ and there exists a self-p-printable set $S \in \Delta_3^P$ such that $\overline{A} \in \text{NP}^S$, then $A \in \widehat{\text{low}}_3$.*

Proof. If $\overline{A} \in \text{NP}^S$, then $\text{NP}^A \subseteq \text{NP}^S$. (To answer a query to A , the NP^S machine runs the NP algorithm for A and the NP^S algorithm for \overline{A} in parallel. One of the algorithms will be correct.) So, we have

$$P^{\text{NP}^{\text{NP}^A}} \subseteq P^{\text{NP}^{\text{NP}^S}}$$

by replacing the NP^A oracle with an NP^S oracle. However, $S \in \Delta_3^P$ and is self-p-printable, so a Δ_3^P machine can write down an initial segment of S that includes all queries to S in a $\Delta_3^{P,S}$ computation. (The length of this initial segment is bounded by a polynomial.) Since Δ_3^P consists of a P base machine and an NP^{NP} oracle, the P base machine (with the help of the NP^{NP} oracle) can write down this initial segment and send it with subsequent oracle queries to NP^{NP} . With this extra advice, the NP^{NP} oracle does not need to consult an S oracle, so

$$\Delta_3^{P,A} \subseteq \Delta_3^{P,S} \subseteq \Delta_3^P.$$

Therefore, $A \in \widehat{\text{low}}_3$. \square

The theorem gives us a sufficient technical condition for a set to be in $\widehat{\text{low}}_3$. The following corollary clarifies the picture somewhat.

COROLLARY 4.5. *Let A be any language in NP. If one of the following conditions holds, then $A \in \widehat{\text{low}}_3$.*

1. $P^{A[2]} \subseteq P^{\text{SAT}[1]}$.
2. $P^{A\| [k+1]} \subseteq P^{\text{SAT}\| [k]}$, for some $k \geq 1$.
3. $P^{A\| [k+1]} = P^{A\| [k]}$, for some $k \geq 1$.
4. $P^{A[k+1]} = P^{A[k]}$, for some $k \geq 1$.

Proof. To prove Part 1, assume $P^{A[2]} \subseteq P^{\text{SAT}[1]}$. Then, $\text{BL}_2(A) \leq_m^P \text{co-BL}_2(\text{SAT})$ because $\text{BL}_2(A) \in P^{A\| [2]}$, $P^{A\| [2]} \subseteq P^{A[2]} \subseteq P^{\text{SAT}[1]} \subseteq \text{co-BH}_2$, and $\text{co-BL}_2(\text{SAT})$ is \leq_m^P -complete for co-BH_2 . Thus, by the theorem, $A \in \widehat{\text{low}}_3$.

To prove Part 2, assume that $P^{A\| [k+1]} \subseteq P^{\text{SAT}\| [k]}$. Then

$$\text{BL}_{k+1}(A) \in P^{A\| [k+1]} \subseteq P^{\text{SAT}\| [k]} \subseteq \text{co-BH}_{k+1}.$$

Since $\text{co-BL}_{k+1}(\text{SAT})$ is \leq_m^P -complete for co-BH_{k+1} , there exists a many-one reduction from $\text{BL}_{k+1}(A)$ to $\text{co-BL}_{k+1}(\text{SAT})$. Again, by the theorem, $A \in \widehat{\text{low}}_3$. Part 3 follows from Part 2, since $P^{A\| [k+1]} = P^{A\| [k]}$ implies $P^{A\| [k+1]} \subseteq P^{\text{SAT}\| [k]}$.

To show Part 4, note that $P^{A[k+1]} = P^{A[k]}$ implies $P^{A[2^k]} \subseteq P^{A[k]}$ because the whole query hierarchy over A collapses. Thus, $P^{A\| [2^k]} \subseteq P^{A[2^k]} \subseteq P^{A[k]} \subseteq P^{A\| [2^k-1]}$. Then, $A \in \widehat{\text{low}}_3$ follows from Part 3. \square

Parts 3 and 4 of Corollary 4.5 state that if the parallel or serial query hierarchy over A collapses, then A is not very hard. Part 2 says that if a set A is not in $\widehat{\text{low}}_3$, then not only is the parallel query hierarchy over A proper, it also rises in lock step with QH_{\parallel} (see Figure 1). For NP-hard sets, we can relate Theorem 4.2 to the collapse of PH. In particular, if the query hierarchy over an NP \leq_T^P -complete set collapses, then PH collapses to Δ_3^P .

COROLLARY 4.6. *If $A \in \text{high}_j$ for some j and one of the following holds, then PH is finite.*

1. $\text{P}^{A[2]} \subseteq \text{P}^{\text{SAT}[1]}$.
2. $\text{P}^{A\parallel[k+1]} \subseteq \text{P}^{\text{SAT}\parallel[k]}$, for some $k \geq 1$.
3. $\text{P}^{A\parallel[k+1]} = \text{P}^{A\parallel[k]}$, for some $k \geq 1$.
4. $\text{P}^{A[k+1]} = \text{P}^{A[k]}$, for some $k \geq 1$.

Proof. Let $i = \max(3, j)$. By Corollary 4.5, $A \in \widehat{\text{low}}_3 \subseteq \text{low}_i$, so $\Sigma_i^P = \Sigma_i^{P,A}$. However, $A \in \text{high}_j \subseteq \text{high}_i$ means $\Sigma_i^{P,A} = \Sigma_{i+1}^P$. Thus, $\Sigma_i^P = \Sigma_i^{P,A} = \Sigma_{i+1}^P$ and $\text{PH} \subseteq \Sigma_i^P$. \square

The next corollary generalizes Kadin's result for SAT and answers the question "Is one query to SAT as powerful as two queries to some other oracle?"

COROLLARY 4.7. *If there exists A such that $\text{P}^{A[2]} = \text{P}^{\text{SAT}[1]}$, then $\text{PH} \subseteq \Delta_3^P$.*

Proof. Suppose that $\text{P}^{A[2]} = \text{P}^{\text{SAT}[1]}$, then $A \in \text{P}^{A[2]} = \text{P}^{\text{SAT}[1]}$. So, we know that $A \leq_m^P \text{SAT} \oplus \overline{\text{SAT}}$ via some polynomial time function g , since $\text{SAT} \oplus \overline{\text{SAT}}$ is \leq_m^P -complete for $\text{P}^{\text{SAT}[1]}$, where $X \oplus Y$ is defined by

$$X \oplus Y \stackrel{\text{def}}{=} \{ 0x \mid x \in X \} \cup \{ 1y \mid y \in Y \}.$$

We split A into two sets,

$$\begin{aligned} A_0 &= \{ x \mid x \in A \text{ and } g(x) = 0F \text{ for some } F \}, \\ A_1 &= \{ x \mid x \in A \text{ and } g(x) = 1F \text{ for some } F \}. \end{aligned}$$

Clearly, $A_0 \in \text{NP}$ and $A_1 \in \text{co-NP}$. Now let $C = A_0 \oplus \overline{A_1}$. One can easily see that $C \in \text{NP}$ and that $\text{P}^{A[k]} = \text{P}^{C[k]}$, for all k . So, $\text{P}^{C[2]} = \text{P}^{A[2]} = \text{P}^{\text{SAT}[1]}$. However, $\text{SAT} \in \text{P}^{C[2]}$ implies C is Turing complete for NP. So, C is a high_0 set. Then, by the proof of Corollary 4.6, Part 1, $\text{P}^{C[2]} \subseteq \text{P}^{\text{SAT}[1]}$ implies PH collapses to Δ_3^P . \square

We can generalize Corollary 4.7 for parallel queries.

COROLLARY 4.8. *If there exists any set A such that $\text{P}^{A\parallel[r]} = \text{P}^{\text{SAT}\parallel[k]}$, and $r > k > 0$, then $\text{PH} \subseteq \Delta_3^P$.*

Proof. Since $r > k$, we have $\text{P}^{A\parallel[k+1]} \subseteq \text{P}^{A\parallel[r]} = \text{P}^{\text{SAT}\parallel[k]} \subseteq \text{co-BH}_{k+1}$. So, $\text{BL}_{k+1}(A) \leq_m^P \text{co-BL}_{k+1}(\text{SAT})$. Then, by Lemma 4.3 there is a self-p-printable sparse set $S \in \Delta_3^P$ such that $\overline{A} \in \text{NP}^S$. Since, $\text{P}^{\overline{A}\parallel[k+1]} = \text{P}^{A\parallel[k+1]} \subseteq \text{co-BH}_{k+1}$, repeating the argument for \overline{A} yields S' such that $A \in \text{NP}^{S'}$. Thus,

$$\text{NP}^A \subseteq \text{NP}^{S \oplus S'}.$$

However, $\text{P}^{A\parallel[r]} = \text{P}^{\text{SAT}\parallel[k]}$ implies $\text{SAT} \in \text{P}^A$. So, $\text{NP}^{\text{SAT}} \subseteq \text{NP}^A$. Thus,

$$\text{P}^{\text{NP}^{\text{NP}^{\text{SAT}}}} \subseteq \text{P}^{\text{NP}^{\text{NP}^A}} \subseteq \text{P}^{\text{NP}^{\text{NP}^{S \oplus S'}}} \subseteq \text{P}^{\text{NP}^{\text{NP}}}.$$

Therefore, $\text{PH} \subseteq \Delta_3^P$. \square

Corollaries 4.7 and 4.8 say that if PH is infinite, then there is no way to split queries to SAT into exact portions. For example, there would not be a set A where

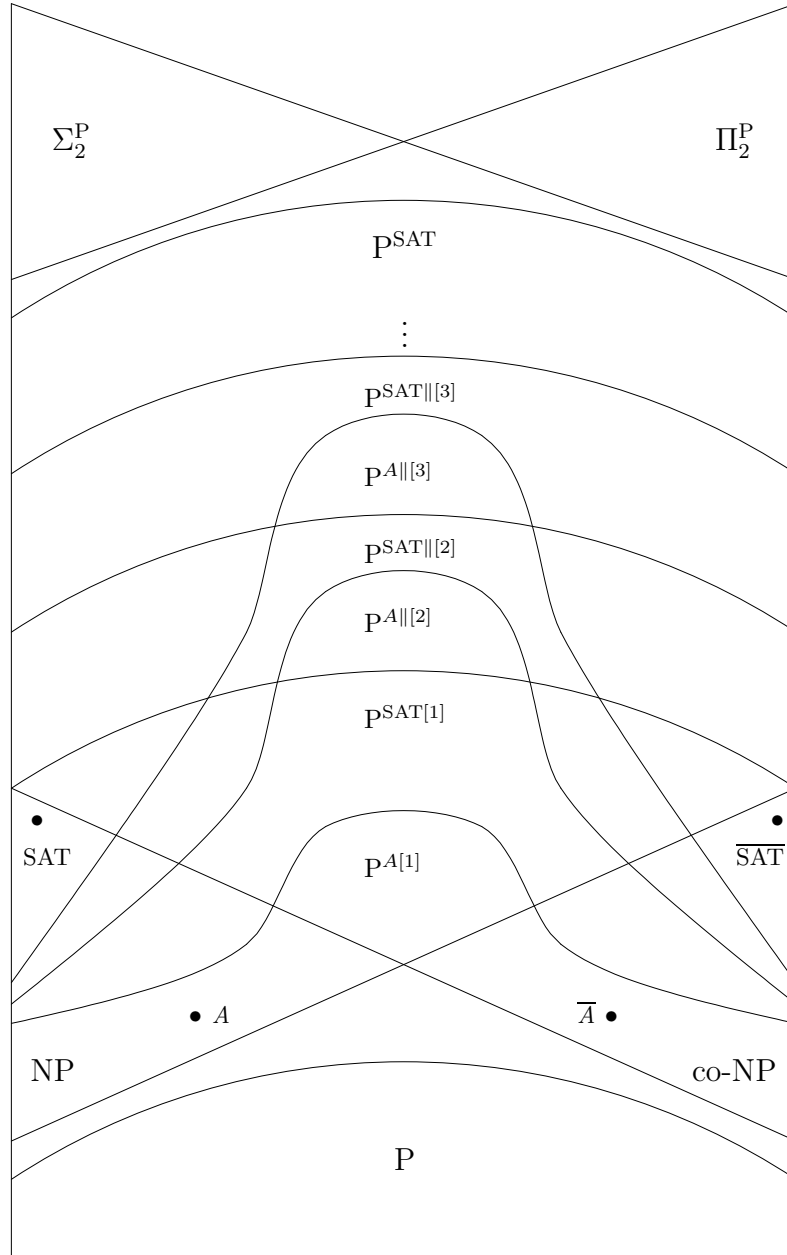


FIG. 1. Let A be an incomplete set in $\text{NP} - \widehat{\text{low}}_3$.

3 parallel queries to A is exactly 2 parallel queries to SAT. However, we still do not know if there can be sets B such that $P^{B\|2} = P^{\text{SAT}\|3}$. If we can show that such sets do not exist (under certain hypotheses), then we can put forward the thesis that parallel queries to SAT are “atomic” or “indivisible” in some sense.

Now we return to queries to sets in NP. In the next two corollaries, we explore technical conditions that allow us to strengthen the results about the serial query hierarchies.

COROLLARY 4.9. *If $A \in \text{NP} - \widehat{\text{low}}_3$ then there exist $B \in \text{NP}$ such that $A \leq_m^P B$, $P^A = P^B$ and for all k , $P^{B\|k+1} \not\subseteq P^{\text{SAT}\|k}$.*

Proof. Note that if $P^B = P^A$ then A and B are in the same high or low level. So, this corollary also says that for every high and low level above $\widehat{\text{low}}_3$ there is a set B whose serial query hierarchy rises in lock step with the serial query hierarchy for SAT.

To prove this lemma, we need to mimic Beigel’s mind change technique for sets other than SAT. SAT has some special properties that make the technique work.

DEFINITION 4.10.

$$\begin{aligned} \text{OR}_2(C) &= \{ \langle x, y \rangle \mid x \in C \text{ or } y \in C \}, \\ \text{AND}_2(C) &= \{ \langle x, y \rangle \mid x \in C \text{ and } y \in C \}. \end{aligned}$$

(The “2” in the subscript means binary.)

SAT is a special set because $\text{OR}_2(\text{SAT}) \leq_m^P \text{SAT}$ and $\text{AND}_2(\text{SAT}) \leq_m^P \text{SAT}$. By going over Beigel’s mind change proof [1, 4], one can show that if a set $C \in \text{NP}$ has the property that $\text{OR}_2(C) \leq_m^P C$ and $\text{AND}_2(C) \leq_m^P C$, then $P^{C\|[2^k-1]} = P^{C\|k}$.

Now we modify A slightly, so it has the desired properties. Let B be the set of tuples of the form $\langle F, x_1, \dots, x_n \rangle$ such that F is a boolean formula over n variables y_1, \dots, y_n without negation, and when y_i is evaluated as “ $x_i \in A$ ”, F evaluates to 1. For example, if $a_1, a_2 \in A$ and $a_3, a_4 \notin A$ then:

$$\begin{aligned} \langle y_1 \wedge y_2, a_1, a_2 \rangle &\in B, & \langle y_1 \wedge y_2, a_1, a_3 \rangle &\notin B, \\ \langle y_1 \vee y_2, a_1, a_3 \rangle &\in B, & \langle y_1 \vee y_2, a_4, a_3 \rangle &\notin B. \end{aligned}$$

Clearly, B is in NP and $A \leq_m^P B$. Also, $B \in \text{NP} - \widehat{\text{low}}_3$, because $P^B = P^A$. Moreover, $\text{OR}_2(B) \leq_m^P B$ and $\text{AND}_2(B) \leq_m^P B$, so $P^{B\|[2^k-1]} = P^{B\|k}$. Now if $P^{B\|k+1} \subseteq P^{\text{SAT}\|k}$, we know that

$$\text{BL}_{2^k}(B) \in P^{B\|[2^k]} \subseteq P^{B\|[2^{k+1}-1]} = P^{B\|k+1} \subseteq P^{\text{SAT}\|k} = P^{\text{SAT}\|[2^k-1]} \subseteq \text{co-BH}_{2^k}.$$

Thus, $\text{BL}_{2^k}(B) \leq_m^P \text{co-BL}_{2^k}(\text{SAT})$ which contradicts the assumption that $B \notin \widehat{\text{low}}_3$. \square

We can say more about the existence of incomplete sets whose serial query hierarchies rise in lock step with QH. (Note that it may be the case that none of the sets in $\text{NP} - \widehat{\text{low}}_3$ are incomplete. They may all be Turing complete for NP or complete for NP in some other sense. However, in this case PH is finite.)

COROLLARY 4.11. *If the Polynomial Hierarchy is infinite, then there exists a set $B \in \text{NP}$ such that for all k , $P^{B\|k+1} \not\subseteq P^{\text{SAT}\|k}$, but B is not high.*

Proof. If PH is infinite, then by a Ladner-like delayed diagonalization [8, 9] we can construct a set $I \in \text{NP}$ that is neither high nor low (I stands for intermediate). In particular, $I \notin \widehat{\text{low}}_3$, so using Corollary 4.9 we can obtain a set B such that for all k ,

$P^{B[k+1]} \not\subseteq P^{\text{SAT}[k]}$. Since $P^B = P^I$, B is intermediate if and only if I is intermediate. So, B is not high. \square

Note that if B is not high, then B is not NP-hard under many-one, Turing, strong nondeterministic or other assorted reductions. In particular B is not Turing complete for NP so $\text{SAT} \notin P^B$. So, Corollary 4.11 says that the serial query hierarchy over B rises in lock step with QH but never captures SAT.

5. Proof of Lemma 4.3.

LEMMA 4.3 *If $B \in \text{NP}$ and $\text{BL}_k(A) \leq_m^P \text{co-BL}_k(B)$, then there exists a self-printable set $S \in \Delta_3^P$ such that $\bar{A} \in \text{NP}^S$.*

Proof. First, note that we do not assume $A \in \text{NP}$. However, the assumption that $\text{BL}_k(A) \leq_m^P \text{co-BL}_k(B)$ implies that $A \in P^{\text{NP}}$, since

$$A \leq_m^P \text{BL}_k(A), \quad \text{BL}_k(A) \leq_m^P \text{co-BL}_k(B) \text{ and } \text{co-BL}_k(B) \in P^{\text{NP}}.$$

So, by the closure of P^{NP} under \leq_m^P reductions, $A \in P^{\text{NP}}$.

Now, we prove this lemma by producing a Δ_3^P program which on input 1^n generates a finite set T_n with $\leq kn$ elements. Furthermore, every string in T_n will have length n . The set produced, $S = \bigcup_{n \geq 1} T_n$, will have the specified properties. The main part of the program is a loop which is iterated for values of i from 0 up to $k-1$. Each iteration produces either the desired T_n or a ‘‘reduction’’ from $\text{BL}_{k-i}(A)$ to $\text{co-BL}_{k-i}(B)$ for strings of length n .

In the following discussion, let g be the polynomial time function that reduces $\text{BL}_k(A)$ to $\text{co-BL}_k(B)$ and let $j = k - i$. We also use the following notational devices:

DEFINITION 5.1. *Let $\langle x_1, \dots, x_k \rangle$ be any k -tuple. When the individual strings in the tuple are not significant, we will substitute \vec{x} for $\langle x_1, \dots, x_k \rangle$. Also, we write \vec{x}^R for $\langle x_k, \dots, x_1 \rangle$, the reversal of the tuple. Finally, we will use $\{0, 1\}^{m \times k}$ to denote the set of k -tuples of strings of length m .*

DEFINITION 5.2. *We will write π_j for the j^{th} projection function, and $\pi_{(i,j)}$ for the function that selects the i^{th} through j^{th} elements of a k -tuple. For example,*

$$\pi_j(x_1, \dots, x_k) = x_j, \text{ and } \pi_{(i,j)}(x_1, \dots, x_k) = \langle x_i, \dots, x_j \rangle.$$

We maintain the following loop invariant to assist our proof. Before each iteration, we have $\vec{z} = \langle z_1, \dots, z_i \rangle \in \{0, 1\}^{n \times i}$ such that for all $\vec{x} = \langle x_1, \dots, x_j \rangle \in \{0, 1\}^{n \times j}$,

$$\vec{x} \in \text{BL}_j(A) \iff \pi_{(1,j)} \circ g(\vec{x}, \vec{z}^R) \in \text{co-BL}_j(B).$$

This loop invariant holds trivially for $i = 0$, since g is a reduction from $\text{BL}_k(A)$ to $\text{co-BL}_k(B)$. The body of the loop is given in Figure 2. Note that the loop terminates either at step 1 or at step 5.

Now we show how to construct T_n . There are two cases. If $\bar{A}^n = \emptyset$, then we put $@^n$ in T_n . Note that we can easily check if $\bar{A}^n = \emptyset$ with a Π_2^P question, because $A \in P^{\text{NP}}$. If $\bar{A}^n \neq \emptyset$, then we start the loop described in Figure 2 with $i = 0$. When the loop terminates, we put z_1, \dots, z_i and all their padded prefixes in T_n . We still have to prove two claims. First, we must show that the loop invariant holds from iteration to iteration. Also, we need to show that $\bar{A}^n \in \text{NP}^{T_n}$.

CLAIM 1: Suppose that in some iteration i we reach step 5. From the loop invariant of the i^{th} loop iteration, we know that $\forall \vec{x} = \langle x_1, \dots, x_j \rangle \in \{0, 1\}^{n \times j}$,

$$\vec{x} \in \text{BL}_j(A) \iff \pi_{(1,j)} \circ g(\vec{x}, \vec{z}^R) \in \text{co-BL}_j(B).$$

1. if $i = k - 1$, then write down $\vec{z} = \langle z_1, \dots, z_i \rangle$ and exit the loop.
2. Compile a function $h : \{0, 1\}^{n \times j} \rightarrow \{0, 1\}^*$ defined by $h(\vec{v}) = \pi_j \circ g(\vec{v}, \vec{z}^R)$.
3. Ask the NP^{NP} oracle:

$$\text{“}\forall x \in \overline{A}^n, \exists \vec{x}' = \langle x_1, \dots, x_{j-1} \rangle \in \{0, 1\}^{n \times (j-1)}, h(\vec{x}', x) \in B\text{”}$$

(It is important to note here that the question above can be answered by an NP^{NP} oracle because $A \in \text{P}^{\text{NP}}$.)

4. If the oracle answers yes, then write down $\vec{z} = \langle z_1, \dots, z_i \rangle$ and exit the loop.
5. If the oracle answers no, then there exists an $x \in \overline{A}^n$ such that

$$\forall \vec{x}' \in \{0, 1\}^{n \times (j-1)}, h(\vec{x}', x) \notin B.$$

Using binary search and the NP^{NP} oracle, find the lexically smallest such x and write it down. Finally, let $z_{i+1} := x$, $\vec{z} := \langle z_1, \dots, z_{i+1} \rangle$, $i := i + 1$ and advance to the next iteration.

FIG. 2. *The body of the loop.*

Let $\vec{u} = \langle u_1, \dots, u_j \rangle = \pi_{(1,j)} \circ g(\vec{x}, \vec{z}^R)$. Then,

$$\vec{x} \in \text{BL}_j(A) \iff \vec{u} \in \text{co-BL}_j(B).$$

Now, let $\vec{x}' = \langle x_1, \dots, x_{j-1} \rangle$ and $\vec{u}' = \langle u_1, \dots, u_{j-1} \rangle$. If j is even, then by the definition of $\text{BL}_j(A)$ and $\text{co-BL}_j(B)$ we have

$$\vec{x}' \in \text{BL}_{j-1}(A) \text{ and } x_j \in \overline{A} \iff \vec{u}' \in \text{co-BL}_{j-1}(B) \text{ or } u_j \in B.$$

Now fix x_j to be the lexically smallest x found in step 5. We know that $x \in \overline{A}$ and $u_j = h(\vec{x}', x) \notin B$, so we are left with

$$\vec{x}' \in \text{BL}_{j-1}(A) \iff \vec{u}' \in \text{co-BL}_{j-1}(B).$$

Since $\vec{u}' = \pi_{(1,j-1)} \circ g(\vec{x}', x, z_i, \dots, z_1)$, this is exactly the the loop invariant for the $(i + 1)^{\text{th}}$ iteration when we define z_{i+1} to be x .

In the other case, if j is odd, we have

$$\vec{x}' \in \text{BL}_{j-1}(A) \text{ or } x_j \in A \iff \vec{u}' \in \text{co-BL}_{j-1}(B) \text{ and } u_j \in \overline{B},$$

or (by negating both sides of the iff)

$$\vec{x}' \notin \text{BL}_{j-1}(A) \text{ and } x_j \in \overline{A} \iff \vec{u}' \notin \text{co-BL}_{j-1}(B) \text{ or } u_j \in B.$$

Fixing x_j to be the minimal x found in step 5, we have

$$\vec{x}' \notin \text{BL}_{j-1}(A) \iff \vec{u}' \notin \text{co-BL}_{j-1}(B),$$

or (by negating both sides of the iff)

$$\vec{x}' \in \text{BL}_{j-1}(A) \iff \vec{u}' \in \text{co-BL}_{j-1}(B),$$

Again, this is the loop invariant for the next iteration. Thus, in both cases we manage to maintain the loop invariant.

CLAIM 2: We need to show that $\overline{A}^{\text{=n}} \in \text{NP}^{T_n}$. There are two cases to consider.

CASE 1: If the loop terminated with $i = k - 1$, then $\vec{z} = \langle z_1, \dots, z_{k-1} \rangle$ and from the loop invariant we know that for all $x \in \{0, 1\}^n$,

$$x \in \text{BL}_1(A) \iff \pi_{(1,1)} \circ g(x, \vec{z}^R) \in \text{co-BL}_1(B).$$

However, $\text{BL}_1(A) = A$ and $\text{co-BL}_1(B) = \overline{B}$, so we really have

$$x \in A \iff \pi_1 \circ g(x, \vec{z}^R) \in \overline{B}$$

or (by negating both sides of the iff)

$$x \in \overline{A} \iff \pi_1 \circ g(x, \vec{z}^R) \in B.$$

To check if $x \in \overline{A}^{\text{=n}}$ an NP^{T_n} machine simply queries T_n to find \vec{z} , computes $u = \pi_1 \circ g(x, \vec{z})$ and accepts iff $u \in B$.

CASE 2: If the loop terminated with $i < k - 1$, then $\vec{z} = \langle z_1, \dots, z_i \rangle$. We want to show that

$$x \in \overline{A}^{\text{=n}} \iff \exists \vec{x}' = \langle x_1, \dots, x_{j-1} \rangle \in \{0, 1\}^{n \times (j-1)}, h(\vec{x}', x) \in B,$$

where $j = k - i$ and $h(\vec{v}) = \pi_j \circ g(\vec{v}, \vec{z}^R)$. Since the loop terminated in step 4, the NP^{NP} oracle must have answered yes to the question:

$$“\forall x \in \overline{A}^{\text{=n}} \exists \vec{x}' = \langle x_1, \dots, x_{j-1} \rangle \in \{0, 1\}^{n \times (j-1)}, h(\vec{x}', x) \in B?”$$

So, we obtain one direction of the iff

$$(1) \quad x \in \overline{A}^{\text{=n}} \implies \exists \vec{x}' = \langle x_1, \dots, x_{j-1} \rangle \in \{0, 1\}^{n \times (j-1)}, h(\vec{x}', x) \in B,$$

Moreover, we know from the loop invariant that $\forall \vec{x}' = \langle x_1, \dots, x_{j-1} \rangle \in \{0, 1\}^{n \times (j-1)}$,

$$\langle \vec{x}', x \rangle \in \text{BL}_j(A) \iff \pi_{(1,j)} \circ g(\vec{x}', x, \vec{z}^R) \in \text{co-BL}_j(B).$$

Again, let $\vec{u} = \langle u_1, \dots, u_j \rangle = \pi_{(1,j)} \circ g(\vec{x}', x, \vec{z}^R)$ and let $\vec{u}' = \langle u_1, \dots, u_{j-1} \rangle$. If j is even, then we know from the definition of $\text{BL}_j(A)$ and $\text{co-BL}_j(B)$ that

$$\vec{x}' \in \text{BL}_{j-1}(A) \text{ and } x \in \overline{A} \iff \vec{u}' \in \text{co-BL}_{j-1}(B) \text{ or } u_j \in B.$$

If j is odd, we get

$$\vec{x}' \in \text{BL}_{j-1}(A) \text{ or } x \in A \iff \vec{u}' \in \text{co-BL}_{j-1}(B) \text{ and } u_j \in \overline{B}.$$

Note that $u_j = h(\vec{x}', x)$ and that in either case $u_j \in B$ implies $x \in \overline{A}$. So, we obtain the other direction of the iff

$$(2) \quad \exists \vec{x}' = \langle x_1, \dots, x_{j-1} \rangle \in \{0, 1\}^{n \times (j-1)}, h(\vec{x}', x) \in B \implies x \in \overline{A}.$$

Combining the implications in (1) and (2), we have

$$x \in \overline{A}^{\text{=n}} \iff \exists \vec{x}' = \langle x_1, \dots, x_{j-1} \rangle \in \{0, 1\}^{n \times (j-1)}, h(\vec{x}', x) \in B,$$

This relationship allows us to compute $\overline{A}^{\text{=n}}$ with an NP^{T_n} machine. To check if $x \in \overline{A}^{\text{=n}}$, an NP^{T_n} machine queries T_n to find $\vec{z} = \langle z_1, \dots, z_i \rangle$, guesses $\vec{x}' = \langle x_1, \dots, x_{j-1} \rangle$ and accepts iff $\pi_j \circ g(\vec{x}', x, \vec{z}^R) \in B$.

In summary, we constructed $S \in \Delta_3^P$ length by length (i.e., $S = \bigcup_{n \geq 1} T_n$). Each T_n has at most kn strings and all the strings in T_n are of length n . Also, each T_n is closed under prefixes, so S is self-p-printable. Finally, $\overline{A} \in \text{NP}^S$ because the following NP^S program determines if $x \in \overline{A}$:

1. Let $|x| = n$.
2. If $@^n \in S$, then $\overline{A}^n = \emptyset$. Reject x .
3. Print out the strings z_1, \dots, z_i in S^n . Let $j = k - i$
4. If $i = k - 1$, compute $u = \pi_1 \circ g(x, \overline{z}^R)$ and accept iff $u \in B$.
5. if $i < k - 1$ accept x iff $\exists \overline{x}' \in \{0, 1\}^{n \times (j-1)}$, $\pi_j \circ g(x_1, \dots, x_i, x, \overline{z}^R) \in B$.

□

6. Conclusion. In this paper we have shown that except for the sets in $\widehat{\text{low}}_3$, sets in NP yield proper query hierarchies. In fact, assuming that PH is infinite, there even exist incomplete sets that produce proper query hierarchies.

Many open questions remain. For example, we know that query hierarchies over sets in P always collapse. We also know that for any $A \in \text{NP} \cap \text{co-NP}$ there exists $B \in \text{NP} \cap \text{co-NP}$ such that $A \leq_m^P B$ and the query hierarchy over B collapses. Are there sets in $\text{NP} \cap \text{co-NP}$ that have proper query hierarchies? What about sets between $\widehat{\text{low}}_3$ and $\text{NP} \cap \text{co-NP}$? Are their query hierarchies proper? Many interesting sub-classes of NP are contained in this region, including sparse sets in NP, $\text{NP} \cap \text{P/poly}$, R and $\text{BPP} \cap \text{NP}$. Can we show that any of these sets have proper or collapsing hierarchies? Also, we would like to strengthen the results for serial query hierarchies. We know that $\text{AND}_2(A) \leq_m^P A$ and $\text{OR}_2(A) \leq_m^P A$ is a sufficient condition. We know that this condition holds for all \leq_m^P -complete sets, all sets in P, and Graph Isomorphism. Primes remains the only candidate for a natural language that does not have this property.

7. Acknowledgements. The author would like to thank his advisor Juris Hartmanis for guidance and support, Jim Kadin for invaluable discussions and for suggesting nice open questions, Wei Li and Desh Ranjan for keeping him on his toes, Georges Lauri for proofreading this paper, and Christine Piatko for lending him a red pen during a time of great need.

REFERENCES

- [1] R. BEIGEL, *Bounded queries to SAT and the Boolean hierarchy*, Theoretical Comput. Sci., 84 (1991), pp. 199–223.
- [2] J. CAI, T. GUNDERMANN, J. HARTMANIS, L. HEMACHANDRA, V. SEWELSON, K. WAGNER, AND G. WECHSUNG, *The Boolean hierarchy I: Structural properties*, SIAM J. Comput., 17 (1988), pp. 1232–1252.
- [3] R. CHANG AND J. KADIN, *The Boolean hierarchy and the polynomial hierarchy: a closer connection*, in Proceedings of the 5th Structure in Complexity Theory Conference, July 1990, pp. 169–178.
- [4] ———, *On computing Boolean connectives of characteristic functions*, Tech. Report TR 90-1118, Cornell Department of Computer Science, May 1990. To appear in *Mathematical Systems Theory*.
- [5] J. HARTMANIS, *Collapsing hierarchies*, Bulletin of the European Association for Theoretical Computer Science, 33 (1987), pp. 26–39.
- [6] J. KADIN, *The polynomial time hierarchy collapses if the Boolean hierarchy collapses*, SIAM J. Comput., 17 (1988), pp. 1263–1282.
- [7] K. KO AND U. SCHÖNING, *On circuit size complexity and the low hierarchy for NP*, SIAM J. Comput., 14 (1985), pp. 41–51.
- [8] R. LADNER, *On the structure of polynomial time reducibility*, J. Assoc. Comput. Mach., 22 (1975), pp. 155–171.
- [9] U. SCHÖNING, *A uniform approach to obtain diagonal sets in complexity classes*, Theoretical Comput. Sci., 18 (1982), pp. 95–103.
- [10] ———, *A low and a high hierarchy in NP*, J. Comput. System Sci., 27 (1983), pp. 14–28.

- [11] ———, *Complexity and Structure*, vol. 211 of Lecture Notes in Computer Science, Springer-Verlag, 1985.
- [12] ———, *Graph isomorphism is in the low hierarchy*, in Proceedings of the 4th Symposium on Theoretical Aspects of Computer Science, vol. 247 of Lecture Notes in Computer Science, Springer-Verlag, 1987, pp. 114–124.
- [13] K. WAGNER, *Bounded query computations*, in Proceedings of the 3rd Structure in Complexity Theory Conference, June 1988, pp. 260–277.