**On the Structure of Bounded Queries
to Arbitrary NP Sets**

Richard Chang*

88-944
November 1988

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Supported in part by NSF research grant DCR-8520597

On the Structure of Bounded Queries to Arbitrary NP Sets

Richard Chang*
Computer Science Department
Cornell University
Ithaca, NY 14853

November 3, 1988

Abstract

In [Kad87b], Kadin showed that if the Polynomial Hierarchy (PH) has infinitely many levels, then for all k ,

$$\text{P}^{\text{SAT}}[k] \subsetneq \text{P}^{\text{SAT}}[k+1].$$

In this paper, we extend Kadin's technique to show that a proper query hierarchy is not an exclusive property of SAT. In fact, for any $A \in \text{NP} - \widehat{\text{low}}_3$, if PH is infinite, then

$$\text{P}^A[k] \subsetneq \text{P}^A[k+1].$$

Moreover, for the case of parallel queries, we show that $\text{P}^{A||[k+1]}$ is not contained in $\text{P}^{\text{SAT}}[k]$. We claim that having a proper query hierarchy is a consequence of the oracle access mechanism and not a result of the "hardness" of a set. To support this claim, we show that assuming PH is infinite, one can construct an intermediate set $B \in \text{NP}$ so that

$$\text{P}^{B[k+1]} \not\subseteq \text{P}^{\text{SAT}}[k].$$

That is, the query hierarchy for B grows as "tall" as the query hierarchy for SAT. In addition, B is intermediate, so it is not "hard" in any sense (e.g., not NP hard under many-one, Turing, or strong nondeterministic reductions). Using these same techniques, we explore some other questions about query hierarchies. For example, we show that if there exists any A such that $\text{P}^{A[2]} = \text{P}^{\text{SAT}[1]}$ then PH collapses to Δ_3^{P} .

*Supported in part by NSF research grant DCR-8520597

1 Introduction

In [Kad87b], Kadin made the Boolean and Query Hierarchies respectable by showing that if the Polynomial Hierarchy (PH) has infinitely many levels then the Boolean Hierarchy (BH) and the Query Hierarchy (QH) are proper hierarchies. BH and QH are well studied complexity classes in Δ_2^P . These hierarchies are essentially built from SAT and $\overline{\text{SAT}}$. BH is made by combining SAT and $\overline{\text{SAT}}$ with logical “and” and “or”. QH is made by allowing any polynomial time computable combination of SAT and $\overline{\text{SAT}}$. Kadin uncovered some intricate structural properties of these hierarchies and showed that the collapse of BH implies the collapse of the PH.

In this paper, we look at hierarchies built with an arbitrary set 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 built from 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 hard sets in NP. 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” in any sense (e.g., not NP hard under many-one, Turing, or strong nondeterministic reductions).

We assume that the reader is familiar with P, NP, PH, sparse sets 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 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

$$x \in A \text{ and } w \text{ is a padded prefix of } x \implies w \in A.$$

If a set is sparse and closed under padded prefixes, it is self-p-printable.

2 The Boolean and Query Hierarchies

We now summarize what is already known about the Boolean and Query Hierarchies. This is neither a complete nor a chronological summary. For completeness, see [Kad87a, Kad87b, Kad87c, Bei, CGH⁺88]

We begin with the Parallel Query Hierarchy ($\text{QH}_{\parallel}$). This hierarchy is made 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. In other words, the queries are made in parallel and the oracle answers with a bit vector. Parallel queries are sometimes called non-adaptive. We write $\text{P}^{\text{SAT}||[k]}$ for the class of languages accepted by a polynomial time Turing Machine that makes at most k parallel queries to SAT on input of any length. Of course,

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

This nested sequence collapses nicely, in the sense that if $\text{P}^{\text{SAT}||[k]} = \text{P}^{\text{SAT}||[k+1]}$ then the entire Parallel Query Hierarchy ($\text{QH}_{\parallel} = \bigcup_{j=1}^{\infty} \text{P}^{\text{SAT}||[j]}$) falls down to $\text{P}^{\text{SAT}||[k]}$. If PH is infinite, then $\text{QH}_{\parallel}$ is a proper hierarchy.

$$\text{P}^{\text{SAT}||[1]} \subsetneq \text{P}^{\text{SAT}||[2]} \subsetneq \dots \subsetneq \text{P}^{\text{SAT}||[k]} \subsetneq \text{P}^{\text{SAT}||[k+1]} \subsetneq \dots$$

If we remove the parallel query restriction from $\text{P}^{\text{SAT}||[k]}$ 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 a polynomial time Turing Machine which asks 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 [Bei87, Bei] showed by a clever binary search routine (dubbed the “mind change” trick) that

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

So, we can think of the levels of QH simply as levels of $\text{QH}_{\parallel}$ exponentially far apart. In complexity theory, serial queries are generally considered more natural and relevant than parallel queries. However, $\text{QH}_{\parallel}$ has a finer structure than QH which makes it more amenable to analysis.

Next we examine the Boolean Hierarchy [CGH⁺88]. Like PH, each level of BH is composed of two complementary language classes $\text{BH}(k)$ and $\text{co-BH}(k)$, such that

$$\text{co-BH}(k) = \{L \mid \bar{L} \in \text{BH}(k)\}.$$

$\text{BH}(k)$ is defined inductively, starting with NP and building up with unions and intersections, as follows,

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

The levels of BH are interlaced, much like PH:

$$\begin{aligned}
&\dots \text{BH}(k) \subseteq \text{BH}(k+1) \cap \text{co-BH}(k+1) \subseteq \text{BH}(k+1) \subseteq \dots \\
&\dots \text{co-BH}(k) \subseteq \text{BH}(k+1) \cap \text{co-BH}(k+1) \subseteq \text{co-BH}(k+1) \subseteq \dots
\end{aligned}$$

BH also collapses nicely. If $\text{BH}(k) = \text{co-BH}(k)$ then the entire BH collapses to $\text{BH}(k) \cap \text{co-BH}(k)$. Many results in this area depend on the fact that BH and $\text{QH}_{\parallel}$ are intertwined

$$\dots \subseteq \text{BH}(k) \cup \text{co-BH}(k) \subseteq \text{P}^{\text{SAT} \parallel [k]} \subseteq \text{BH}(k+1) \cap \text{co-BH}(k+1) \subseteq \text{P}^{\text{SAT} \parallel [k+1]} \subseteq \dots$$

Clearly BH is a proper hierarchy iff $\text{QH}_{\parallel}$ 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:

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

3 High and Low Sets

In [Sch83], Schöning defined the high and low hierarchies for NP to classify sets between P and NP. (High and low sets were originally defined for recursive function theory.) Very roughly, the high-low classification measures the amount of information an NP language (acting as an oracle) can give to a base machine in PH. 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 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 doesn't tell the Σ_k^{P} base machine anything that it couldn't compute itself. In the previous case we call A a high set, in the latter a low set. More formally, we define:

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

Clearly,

$$\text{low}_0 \subseteq \text{low}_1 \subseteq \text{low}_2 \subseteq \dots$$

$$\text{high}_0 \subseteq \text{high}_1 \subseteq \text{high}_2 \subseteq \dots$$

Although, high and low sets are called hierarchies, they are not known to have the familiar collapsing behavior. For example, it is not known whether $\text{low}_2 = \text{low}_3$ implies $\text{low}_2 = \text{low}_4$. Nevertheless, the low hierarchy is quite interesting, because it uniformly classifies many well studied classes.

Schöning also defined a refinement of the low hierarchy [Sch86]

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

These classes lie between the levels of the low hierarchy,

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

and allow a finer classification of languages classes in the low hierarchy.

High and low sets have many interesting properties. We will briefly mention some here. See [Sch82,Sch83,KS85,Sch86,Sch87] for proofs.

1. $\text{low}_k = \text{high}_k \iff \text{PH} \subseteq \Sigma_k^P$.
2. if PH is infinite then $\exists I \in \text{NP} \forall k, I \notin \text{high}_k$ 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 , $A \in P^S$ then $A \in \text{low}_3$.
5. $\text{R} \subseteq \text{BPP} \subseteq \text{low}_2$.
6. $\text{high}_0 = \{A \mid A \text{ is } \leq_T^P\text{-complete for NP}\}$.
7. $\text{low}_0 = \text{P}$.
8. $\text{low}_1 = \text{NP} \cap \text{co-NP}$.
9. $\text{Graph Isomorphism} \in \text{low}_2$.

4 Main Theorem

The three hierarchies QH, $\text{QH}_{||}$, and BH are essentially built from SAT and $\overline{\text{SAT}}$. We now consider analogous query hierarchies built from an arbitrary set in NP. For any set $A \in \text{NP}$ we can consider the classes $\text{P}^{A||[k]}$ and $\text{P}^{A[k]}$. We immediately have

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

but in general we do not know how to repeat Beigel's "mind change" trick. So we can relate parallel and serial queries only loosely

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

Query hierarchies built from an arbitrary set collapse nicely, too. That is

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

We will 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 $\text{BH}(k)$ and $\text{co-BH}(k)$.

Definition For each set $A \in \text{NP}$ we define a sequence of languages

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

Note that

$$\text{BL}_{\text{SAT}}(2) = \text{SAT} \wedge \overline{\text{SAT}}, \quad \text{BL}_{\text{SAT}}(3) = (\text{SAT} \wedge \overline{\text{SAT}}) \vee \text{SAT}$$

and in general $\text{BL}_{\text{SAT}}(k)$ is $\leq_m^{\text{P}}$ -complete for $\text{BH}(k)$ and $\text{co-BL}_{\text{SAT}}(k)$ is $\leq_m^{\text{P}}$ -complete for $\text{co-BH}(k)$. Clearly, $\text{BL}_A(k) \in \text{P}^{A||[k]}$ and $\text{co-BL}_A(k) \in \text{P}^{A||[k]}$. So, there is some interaction between these "boolean languages" built from A and the query hierarchies built from A .

With all these definitions in place we can pose some questions originally asked of SAT. For example, we would like to know what happens if $P^{A||[k]} = P^{A||[k+1]}$? if $P^{A[k]} = P^{A[k+1]}$? or if $BL_A(k) \leq_m^P \text{co-}BL_A(k)$? Also, we would like to know the relationship between queries to SAT and queries to A . That is, we know $P^{A||[k]} \subseteq P^{\text{SAT}||[k]}$, but where is $P^{A||[k]}$ in relation to $P^{\text{SAT}||[k-1]}$? Instead of asking “Is one question to SAT as powerful as two?” we ask “Is one question to SAT as powerful as two questions to some other oracle?”

The following theorem answers some of these questions:

Theorem If $A, B \in \text{NP}$ and $BL_A(k) \leq_m^P \text{co-}BL_B(k)$ then $A \in \widehat{\text{low}}_3$.

We prove this theorem in two parts. Lemma 1 is a rather technical lemma, and follows the same lines as Kadin’s original proof. We relegate the proof of Lemma 1 to the next section.

Lemma 1 If $A, B \in \text{NP}$ and $BL_A(k) \leq_m^P \text{co-}BL_B(k)$ then there exists a self-p-printable set $S \in \Delta_3^P$ such that $\bar{A} \in \text{NP}^S$.

Lemma 2 If $A \in \text{NP}$ and there exists a self-p-printable set $S \in \Delta_3^P$ such that $\bar{A} \in \text{NP}^S$, then $A \in \widehat{\text{low}}_3$.

Proof

If $\bar{A} \in \text{NP}^S$ 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 $\bar{A}$ in parallel. One of the algorithms will terminate.) So, we have

$$P^{\text{NP}^{\text{NP}^A}} \subseteq P^{\text{NP}^{\text{NP}^S}} \text{ (i.e., } \Delta_3^{P,A} \subseteq \Delta_3^{P,S} \text{)}$$

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 can write down this initial segment and send it along with the oracle queries to NP^{NP} . Now 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, $A \in \widehat{\text{low}}_3$. □

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 1 If $A \in \text{NP}$ and 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

1. If $P^{A[2]} \subseteq P^{\text{SAT}[1]}$, then $\text{BL}_A(2) \in P^{A||[2]} \subseteq P^{A[2]} \subseteq P^{\text{SAT}[1]} \subseteq \text{co-BH}(2)$. However, $\text{co-BL}_{\text{SAT}}(2)$ is $\leq_m^P$ -complete for $\text{co-BH}(2)$, so $\text{BL}_A(2) \leq_m^P \text{co-BL}_{\text{SAT}}(2)$. Then, by the theorem, $A \in \widehat{\text{low}}_3$.
2. If $P^{A||[k+1]} \subseteq P^{\text{SAT}||[k]}$, then $\text{BL}_A(k+1) \in P^{A||[k+1]} \subseteq P^{\text{SAT}||[k]} \subseteq \text{co-BH}(k+1)$. Again, $\text{co-BL}_{\text{SAT}}(k+1)$ is $\leq_m^P$ -complete for $\text{co-BH}(k+1)$, so we have a reduction from $\text{BL}_A(k+1)$ to $\text{co-BL}_{\text{SAT}}(k+1)$. Then, by the theorem, $A \in \widehat{\text{low}}_3$.
3. $P^{A||[k+1]} = P^{A||[k]}$ implies $P^{A||[k+1]} \subseteq P^{\text{SAT}||[k]}$, so $A \in \widehat{\text{low}}_3$ follows from part 2.
4. $P^{A[k+1]} = P^{A[k]}$ implies $P^{A[2^k]} \subseteq P^{A[k]}$, because the whole query hierarchy based on A collapses. Thus, $P^{A||[2^k]} \subseteq P^{A[2^k]} \subseteq P^{A[k]} \subseteq P^{A||[2^k-1]}$. Then, by part 3, $A \in \widehat{\text{low}}_3$.

□

Parts 3 and 4 of Corollary 1 state that if the parallel or serial query hierarchies built from 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 built from A proper, it also rises in lock step with $\text{QH}_{||}$ (see figure 1). That is, with $k+1$ parallel queries to A , we rise above the k^{th} level of $\text{QH}_{||}$. For serial queries, we only know that two queries to A is above one query to SAT.

For hard sets in NP, we can relate the theorem with the collapse of PH.

Corollary 2 If $A \in \text{high}_j$ for some j and one of the following holds, then PH is finite.

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

Let $i = \max(3, j)$. By Corollary 1, $A \in \widehat{\text{low}}_3 \subseteq \text{low}_i$, 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 $\Sigma_i^P = \Sigma_i^{P,A} = \Sigma_{i+1}^P$. $\square$

In particular, if the query hierarchy built from a Turing complete set collapses then PH collapses to Δ_3^P . This 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 3 If there exists A such that $P^{A[2]} = P^{\text{SAT}[1]}$ then $\text{PH} \subseteq \Delta_3^P$.

Proof

If $P^{A[2]} = P^{\text{SAT}[1]}$ then $A \in P^{A[2]} = P^{\text{SAT}[1]}$. So, $A \leq_m^P \text{SAT} \oplus \overline{\text{SAT}}$ via some polynomial time function g . $\text{SAT} \oplus \overline{\text{SAT}}$ is defined by

$$\text{SAT} \oplus \overline{\text{SAT}} = \{0F \mid F \in \text{SAT}\} \cup \{1F \mid F \in \overline{\text{SAT}}\}.$$

We split A into two sets,

$$A_0 = \{x \mid x \in A \text{ and } g(x) = 0F \text{ for some } F\}$$

$$A_1 = \{x \mid x \in A \text{ and } g(x) = 1F \text{ for some } F\}.$$

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

$$P^{A[k]} = P^{C[k]}, \text{ for all } k.$$

So, $P^{C[2]} = P^{A[2]} = P^{\text{SAT}[1]}$. However, $\text{SAT} \in P^{C[2]}$ implies C is Turing complete for NP. So, C is a high_0 set. Then, by the proof of Corollary 2, part 1, $P^{C[2]} \subseteq P^{\text{SAT}[1]}$ implies PH collapses to Δ_3^P . $\square$

Corollary 4 explores some technical conditions that allow us to strengthen some results about the serial query hierarchies.

Corollary 4 If PH 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 (Sketch)

If PH is infinite, then by a Ladner-like delayed diagonalization [Lad75, Sch82] we can construct a set $I \in \text{NP}$ that is neither high nor low [Sch82] (I stands for intermediate). In particular, $I \notin \widehat{\text{low}}_3$, so none of the conditions in Corollary 1 hold. Part 2 of Corollary 1

tells us that each additional parallel query to I allows the oracle machine to recognize a set not recognized with one less parallel query to SAT. To extend this to serial queries, we need to mimic Beigel's "mind change" trick. SAT has some special properties that make the trick work.

Definition

$$\begin{aligned}\text{OR}_A^2 &= \{\langle x, y \rangle \mid x \in A \text{ or } y \in A\} \\ \text{AND}_A^2 &= \{\langle x, y \rangle \mid x \in A \text{ and } y \in A\}\end{aligned}$$

(The "2" in the superscript means binary.)

SAT is special because $\text{OR}_{\text{SAT}}^2 \leq_m^P \text{SAT}$ and $\text{AND}_{\text{SAT}}^2 \leq_m^P \text{SAT}$. By going over Beigel's "mind change" proof, one can show that if a set $A \in \text{NP}$ has the property that $\text{OR}_A^2 \leq_m^P A$ and $\text{AND}_A^2 \leq_m^P A$, then

$$\text{P}^{A||[2^k-1]} = \text{P}^{A[k]}$$

Now we modify I slightly, so it has the desired properties but remains an intermediate set. Consider, the set

$$B = \{\langle F, x_1, \dots, x_n \rangle \mid F \text{ is a boolean formula over } n \text{ variables } y_1, \dots, y_n \text{ without negation, and when } y_i \text{ is evaluated as } x_i \in I, F \text{ evaluates to 1.}\}$$

Clearly, B is in NP. Also, $\text{P}^B = \text{P}^I$, so B is intermediate iff I is intermediate. Moreover, $\text{OR}_B^2 \leq_m^P B$ and $\text{AND}_B^2 \leq_m^P B$, so

$$\text{P}^{B||[2^k-1]} = \text{P}^{B[k]}.$$

Now if $\text{P}^{B[k+1]} \subseteq \text{P}^{\text{SAT}[k]}$, we know

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

which tells us $\text{BL}_B(k+1) \leq_m^P \text{co-BL}_{\text{SAT}}(k+1)$ and contradicts the assumption that B is not in $\widehat{\text{low}}_3$. $\square$

An interesting consequence of Corollary 4 is that $\text{SAT} \notin \text{P}^B$. That is, the query hierarchy based on B rises in lock step with the query hierarchy based on SAT, but never captures SAT. We view this as evidence that having a proper query hierarchy is a result of the oracle access mechanism and not a result of the hardness or content of the oracle.

5 Proof of Lemma 1

Lemma 1 If $A, B \in \text{NP}$ and $\text{BL}_A(k) \leq_m^P \text{co-BL}_B(k)$ then there exists a self-p-printable set $S \in \Delta_3^P$ such that $\overline{A} \in \text{NP}^S$.

Proof

We prove this 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 to $k - 1$. Each iteration produces either the desired T_n or a “reduction” from $\text{BL}_A(k - i)$ to $\text{co-BL}_B(k - i)$ for strings of length n .

In the following discussion, let g be the polynomial time function that reduces $\text{BL}_A(k)$ to $\text{co-BL}_B(k)$ and let $j = k - i$. We maintain the following loop invariant to assist our proof. Before each iteration, we have $z_1, \dots, z_i \in \{0, 1\}^n$ such that $\forall x_1, \dots, x_j \in \{0, 1\}^n$,

$$\begin{aligned} g(\langle x_1, \dots, x_j, z_i, \dots, z_1 \rangle) &= \langle u_1, \dots, u_j, v_i, \dots, v_1 \rangle \\ \implies [\langle x_1, \dots, x_j \rangle \in \text{BL}_A(j) &\iff \langle u_1, \dots, u_j \rangle \in \text{co-BL}_B(j)] \end{aligned}$$

This loop invariant holds trivially for $i = 0$, since g is a reduction from $\text{BL}_A(k)$ to $\text{co-BL}_B(k)$. In each iteration of the loop we do the following.

1. if $i = k - 1$ then write down $z_1, \dots, z_i$ and exit the loop.
2. Compile a function $h : \{0, 1\}^{n \times j} \rightarrow \{0, 1\}^*$ defined by

$$h(\langle x_1, \dots, x_j \rangle) = \pi_j \circ g(\langle x_1, \dots, x_j, z_i, \dots, z_1 \rangle),$$

where π_j is the projection onto the j^{th} element.

3. Ask the NP^{NP} oracle: “ $\forall x \in \overline{A}^n \exists x_1, \dots, x_{j-1} \in \{0, 1\}^n, h(\langle x_1, \dots, x_{j-1}, x \rangle) \in B$?”
4. If the oracle answers “yes”, then write down $z_1, \dots, z_i$ and exit the loop.
5. If the oracle answers “no”, we know there exists an $x \in \overline{A}^n$ such that $\forall x_1, \dots, x_{j-1} \in \{0, 1\}^n, h(\langle x_1, \dots, x_{j-1}, x \rangle) \notin B$. Using binary search and the NP^{NP} oracle, we can find the lexically smallest such x and write it down. Let $z_{i+1} = x$ and advance to the next iteration.

Now we show how to construct T_n . There are two cases. If $\overline{A}^n = \emptyset$ then we put $@$ in T_n . Note that we can easily check if $\overline{A}^n = \emptyset$ with a Π_1^P question. If $\overline{A}^n \neq \emptyset$ then we

start the loop described above 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 $\overline{A}^{\text{=n}} \in \text{NP}^{T_n}$.

Claim 1: Suppose that in some iteration i we reach step 5. From the loop invariant of the current loop iteration, we know that $\forall x_1, \dots, x_j \in \{0, 1\}^n$,

$$\begin{aligned} g(\langle x_1, \dots, x_j, z_i, \dots, z_1 \rangle) &= \langle u_1, \dots, u_j, v_i, \dots, v_1 \rangle \\ \implies [\langle x_1, \dots, x_j \rangle \in \text{BL}_A(j) &\iff \langle u_1, \dots, u_j \rangle \in \text{co-BL}_B(j)]. \end{aligned}$$

If j is even, then by the definition of $\text{BL}_A(j)$ and $\text{co-BL}_B(j)$ we have

$$\begin{aligned} \langle x_1, \dots, x_{j-1} \rangle \in \text{BL}_A(j-1) &\iff \langle u_1, \dots, u_{j-1} \rangle \in \text{co-BL}_B(j-1) \\ \text{and } x_j \in \overline{A} &\iff \text{or } u_j \in B. \end{aligned}$$

Now fix x_j to be the x found in step 5. We know that $x \in \overline{A}$ and $u_j = h(\langle x_1, \dots, x_{j-1}, x \rangle) \notin B$, so we are left with

$$\langle x_1, \dots, x_{j-1} \rangle \in \text{BL}_A(j-1) \iff \langle u_1, \dots, u_{j-1} \rangle \in \text{co-BL}_B(j-1).$$

This is exactly the the loop invariant for the $(i+1)^{\text{th}}$ iteration, so we can maintain the loop invariant by setting z_{i+1} to be x . If j is odd, we get

$$\begin{aligned} \langle x_1, \dots, x_{j-1} \rangle \in \text{BL}_A(j-1) &\iff \langle u_1, \dots, u_{j-1} \rangle \in \text{co-BL}_B(j-1) \\ \text{or } x_j \in A &\iff \text{and } u_j \in \overline{B}. \end{aligned}$$

or

$$\begin{aligned} \langle x_1, \dots, x_{j-1} \rangle \notin \text{BL}_A(j-1) &\iff \langle u_1, \dots, u_{j-1} \rangle \notin \text{co-BL}_B(j-1) \\ \text{and } x_j \in \overline{A} &\iff \text{or } u_j \in B. \end{aligned}$$

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

$$\langle x_1, \dots, x_{j-1} \rangle \notin \text{BL}_A(j-1) \iff \langle u_1, \dots, u_{j-1} \rangle \notin \text{co-BL}_B(j-1)$$

or

$$\langle x_1, \dots, x_{j-1} \rangle \in \text{BL}_A(j-1) \iff \langle u_1, \dots, u_{j-1} \rangle \in \text{co-BL}_B(j-1).$$

So, 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 from the loop invariant we know that $\forall x \in \{0, 1\}^n$,

$$g(\langle x, z_i, \dots, z_1 \rangle) = \langle u, v_i, \dots, v_1 \rangle \implies [x \in \text{BL}_A(1) \iff u \in \text{co-BL}_B(1)].$$

However, $\text{BL}_A(1) = A$ and $\text{co-BL}_B(1) = \overline{B}$, so we really have

$$x \in A \iff u \in \overline{B}$$

or

$$x \in \overline{A} \iff u \in B.$$

To check if $x \in \overline{A}^{\text{=n}}$ an NP^{T_n} machine simply queries T_n to find $z_1, \dots, z_{k-1}$, computes $u = \pi_1 \circ g(\langle x, z_{k-1}, \dots, z_1 \rangle)$ and accepts iff $u \in B$.

Case 2: If the loop terminated with $i < k - 1$ then in step 2 the NP^{NP} oracle answered “yes” to “ $\forall x \in \overline{A}^{\text{=n}} \exists x_1, \dots, x_{j-1} \in \{0, 1\}^n, h(\langle x_1, \dots, x_{j-1}, x \rangle) \in B$ ”, where $j = k - i$ and $h(\langle x_1, \dots, x_j \rangle) = \pi_j \circ g(\langle x_1, \dots, x_j, z_i, \dots, z_1 \rangle)$. So, $x \in \overline{A}^{\text{=n}} \implies \exists x_1, \dots, x_{j-1} \in \{0, 1\}^n, h(\langle x_1, \dots, x_{j-1}, x \rangle) \in B$. Moreover, we know from the loop invariant that $\forall x_1, \dots, x_j \in \{0, 1\}^n$,

$$\begin{aligned} g(\langle x_1, \dots, x_j, z_i, \dots, z_1 \rangle) &= \langle u_1, \dots, u_j, v_i, \dots, v_1 \rangle \\ \implies [\langle x_1, \dots, x_j \rangle \in \text{BL}_A(j) &\iff \langle u_1, \dots, u_j \rangle \in \text{co-BL}_B(j)]. \end{aligned}$$

If j is even we know from the definition of $\text{BL}_A(j)$ and $\text{co-BL}_B(j)$ that

$$\begin{aligned} \langle x_1, \dots, x_{j-1} \rangle \in \text{BL}_A(j-1) &\iff \langle u_1, \dots, u_{j-1} \rangle \in \text{co-BL}_B(j-1) \\ \text{and } x_j \in \overline{A} &\text{ or } u_j \in B. \end{aligned}$$

If j is odd, we get

$$\begin{aligned} \langle x_1, \dots, x_{j-1} \rangle \in \text{BL}_A(j-1) &\iff \langle u_1, \dots, u_{j-1} \rangle \in \text{co-BL}_B(j-1) \\ \text{or } x_j \in A &\text{ and } u_j \in \overline{B}. \end{aligned}$$

Note that $u_j = h(\langle x, x_1, \dots, x_{j-1} \rangle)$ and that in either case $u_j \in B \implies x_j \in \overline{A}$. So, if there exist $x_1, \dots, x_{j-1} \in \{0, 1\}^n, h(\langle x_1, \dots, x_{j-1}, x \rangle) \in B$ then $x \in \overline{A}$. Combined with the preceding result, we have

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

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 $z_1, \dots, z_i$, guesses $x_1, \dots, x_{j-1}$, and accepts iff $h(\langle x_1, \dots, x_{j-1}, x \rangle) \in B$.

In summary, we constructed $S \in \Delta_3^{\text{P}}$ length by length (i.e., $S = \bigcup_{n \geq 1}^\infty T_n$). 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 $\bar{A}^n = \emptyset$. 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 $u = \pi_1 \circ g(\langle x, z_i, \dots, z_1 \rangle)$ and accept iff $u \in B$.
5. if $i < k - 1$ accept x iff $\exists x_1, \dots, x_{j-1} \in \{0, 1\}^n$, $\pi_j \circ g(\langle x_1, \dots, x_{j-1}, x, z_i, \dots, z_1 \rangle) \in B$.

□

6 Conclusion

Many open questions remain. For example, we know that query hierarchies built from sets in P always collapse. We also know that for any $A \in NP \cap co-NP$ there exists $B \in NP \cap co-NP$ such that $A \leq_m^P B$ and the query hierarchy built from B collapses. Are there sets in $NP \cap co-NP$ that have proper query hierarchies? What about sets between $\widehat{low}_3$ and $low_1 = NP \cap co-NP$? Are their query hierarchies proper? Many interesting language classes live in this region, including sparse sets in NP , $NP \cap P/poly$, R and BPP . 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

$$AND_A^2 \leq_m^P A, \text{ and } OR_A^2 \leq_m^P A$$

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 set that does not have this property. We would like to see either a construction for an NP set without this property, or a proof that all NP sets have this property.

Now that we know more about query hierarchies, we see that having a proper query hierarchy is not dependent on the “hardness” of a set. In fact, sets in $low_3 - \widehat{low}_3$, if they exist, have proper query hierarchies. Instead we believe that having a proper query hierarchy is a result of the meagerness of the access mechanism. When you are not told very much, every bit of extra information helps. However, we should not consider this an artifact of the definition, because we get a glimpse of the structure of the Polynomial Hierarchy in these “mini” hierarchies within Δ_2^P .

7 Acknowledgements

The author would like to thank his advisor Juris Hartmanis for guidance and support, Jim Kadin for invaluable discussions and for leaving 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

- [Bei] R. Beigel. Bounded queries to SAT and the Boolean Hierarchy. To appear in *Theoretical Computer Science*.
- [Bei87] R. Beigel. Bounded queries to SAT and the Boolean Hierarchy. Technical Report 7, The Johns Hopkins University, Department of Computer Science, 1987.
- [CGH⁺88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The Boolean Hierarchy I: Structural properties. Manuscript, 1988.
- [Kad87a] J. Kadin. Is one NP question as powerful as two? Technical Report TR 87-842, Cornell Department of Computer Science, June 1987.
- [Kad87b] J. Kadin. The Polynomial Time Hierarchy collapses if the Boolean Hierarchy collapses. Technical Report 87-843, Cornell Department of Computer Science, June 1987.
- [Kad87c] J. Kadin. The Polynomial Time Hierarchy collapses if the Boolean Hierarchy collapses. In *Proc. 3rd Structure in Complexity Theory Conference*, June 1987.
- [KS85] K. Ko and U. Schöning. On circuit size complexity and the low hierarchy for NP. *SIAM Journal on Computing*, 14(1):41–51, February 1985.
- [Lad75] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, January 1975.
- [Sch82] U. Schöning. A uniform approach to obtain diagonal sets in complexity classes. *Theoretical Computer Science*, 18:95–103, 1982.
- [Sch83] U. Schöning. A low and a high hierarchy in NP. *Journal of Computer and System Sciences*, 27:14–28, 1983.

- [Sch86] U. Schöning. *Complexity and Structure*. Lecture Notes in Computer Science #211. Springer Verlag, 1986.
- [Sch87] U. Schöning. Graph isomorphism is in the low hierarchy. In *STACS 87*, Lecture Notes in Computer Science #247, pages 114–124. Springer-Verlag, 1987.

Figure 1: If $A \in NP - \widehat{low}_3$, then the query hierarchy for A grows as tall as $QH_{||}$.