The Boolean Hierarchy and the Polynomial Hierarchy: a Closer Connection

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

Jim Kadin†
Computer Science Department
University of Maine
Orono, ME 04469

July 7, 1993

Abstract

We show that if the Boolean hierarchy collapses to level k, then the polynomial hierarchy collapses to $\mathrm{BH}_3(k)$, where $\mathrm{BH}_3(k)$ is the k^{th} level of the Boolean hierarchy over Σ_2^{P}. This is an improvement over the known results [3], which show that the polynomial hierarchy would collapse to $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[O(\log n)]}$. This result is significant in two ways. First, the theorem says that a deeper collapse of the Boolean hierarchy implies a deeper collapse of the polynomial hierarchy. Also, this result points to some previously unexplored connections between the Boolean and query hierarchies of Δ_2^{P} and Δ_3^{P}. Namely,

$$\mathrm{BH}(k) = \text{co-}\mathrm{BH}(k) \Longrightarrow \mathrm{BH}_3(k) = \text{co-}\mathrm{BH}_3(k)$$
$$\mathrm{P}^{\mathrm{NP}\|[k]} = \mathrm{P}^{\mathrm{NP}\|[k+1]} \Longrightarrow \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}\|[k+1]} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}\|[k+2]}.$$

Key words: polynomial time hierarchy, Boolean hierarchy, polynomial time Turing reductions, oracle access, nonuniform algorithms, sparse sets

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

1 Introduction

The Boolean hierarchy (BH) was defined as the closure of NP under Boolean operations and is identical to the difference hierarchy of NP sets [1, 2, 7]. Kadin [3] showed that if the BH collapses at any level, the polynomial time hierarchy (PH) collapses to $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[O(\log n)]}$, the class of languages in $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ that are recognized by deterministic polynomial time machines that make $O(\log n)$ queries

*Supported in part by NSF research grants DCR-8520597 and CCR-88-23053. Current address: Department of Computer Science, University of Maryland Baltimore County, Baltimore, MD 21228.

†This work was carried out while the author was a visiting scientist at Cornell University.

to an $\mathrm{NP}^{\mathrm{NP}}$ oracle. Since the BH is contained in P^{NP}, this result showed that the structure of classes above NP but within P^{NP} is related to the structure of the PH as a whole.

In this paper we extend Kadin's result by showing that if the BH collapses to its k^{th} level (if $\mathrm{BH}(k) = \text{co-}\mathrm{BH}(k)$), then the PH collapses to the k^{th} level of the Boolean hierarchy within Δ_3^{P} (the difference hierarchy of $\mathrm{NP}^{\mathrm{NP}}$ languages). That is

$$\text{if } \mathrm{BH}(k) = \text{co-}\mathrm{BH}(k), \text{ then } \mathrm{PH} \subseteq \mathrm{BH}_3(k)$$

(see Section 2 for precise definitions). The k^{th} level of the Δ_3^{P} Boolean hierarchy ($\mathrm{BH}_3(k)$) is contained within $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[\log k+1]}$, the class of languages in $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ recognized by machines that make at most $\lceil \log k+1 \rceil$ queries for all inputs. Therefore the collapse of the BH implies that the languages within the PH can be recognized by deterministic polynomial time machines that make a constant number of queries to an $\mathrm{NP}^{\mathrm{NP}}$ oracle, and the deeper the collapse of the BH, the smaller this constant is.

This result also yields two unexpected corollaries:

1. If $\mathrm{BH}(k) = \text{co-}\mathrm{BH}(k)$, then $\mathrm{BH}_3(k) = \text{co-}\mathrm{BH}_3(k)$.
2. If $\mathrm{P}^{\mathrm{NP}||[k]} = \mathrm{P}^{\mathrm{NP}||[k+1]}$, then $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}||[k+1]} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}||[k+2]}$.

The first corollary says that the collapse of the Boolean hierarchy in Δ_2^{P} implies an identical collapse of the Boolean hierarchy in Δ_3^{P}. The second corollary says that the bounded query hierarchies within Δ_2^{P} and Δ_3^{P} are linked. $\mathrm{P}^{\mathrm{NP}||[k]}$ is the class of languages recognizable by deterministic polynomial time machines that are allowed to ask k questions in parallel to an oracle from NP (all k queries must be asked at once, so no query can depend on the answers to other queries). $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}||[k]}$ is the class of languages recognizable by deterministic polynomial time machines that are allowed to ask k questions in parallel to an oracle from $\mathrm{NP}^{\mathrm{NP}}$. If the query hierarchy in Δ_2^{P} collapses, then the query hierarchy in Δ_3^{P} collapses to almost the same level. At first glance, one would think that these corollaries could be proven directly by a straightforward oracle replacement proof. However, the only proof that we know uses a refined version of Kadin's "hard/easy" formulas argument.

The "hard/easy" formulas argument [3, 5], which showed that the collapse of the BH implies the collapse of the PH, went as follows:

- If the BH collapses to its k^{th} level, then the unsatisfiable Boolean formulas of each length n can be partitioned into "easy" and "hard" formulas. The easy formulas can be recognized as unsatisfiable by a particular NP algorithm, and the hard formulas cannot be recognized by this algorithm.

- The hard formulas are key strings, because sequences of at most $k-1$ hard formulas of length n contain enough information to allow an NP machine to recognize all the unsatisfiable formulas of length n.

- A sparse set S was constructed by taking one sequence of hard formulas for each length.

- Since co-NP $\subseteq \mathrm{NP}^S$, the results of Yap [8] imply that PH $\subseteq \Sigma_3^{\mathrm{P}}$.
- By arguing further that $S \in \mathrm{NP}^{\mathrm{NP}}$, it was shown that PH $\subseteq \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[O(\log n)]}$. [1]

In this paper we present a deeper analysis of the hard sequences. We show that it is not necessary to choose a particular hard sequence to put into S. In fact a smaller amount of information, contained in a sparse set T, is enough to allow a Σ_2^{P} machine to recognize Σ_3^{P} languages, i.e.

$$\mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}} \subseteq \mathrm{NP}^{T \oplus \mathrm{SAT}}.$$

Since $T \in \mathrm{NP}^{\mathrm{NP}}$ and is almost a tally set (T is a subset of a P-printable set), we are able to show that $\mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}}$ is contained in the k^{th} level of the Δ_3^{P} Boolean hierarchy.

2 Definitions and Notation

We assume that the reader is familiar with the classes NP, co-NP, the polynomial time hierarchy (PH), and the NP-complete set SAT.

Notation For any language L, $L^{\leq n}$ is the set of strings in L of length less than or equal to n. $L^{=n}$ is the set of strings in L of length n.

Notation 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,

$$\pi_j(\langle x_1, \ldots, x_k \rangle) = x_j$$

$$\pi_{(1,j)}(\langle x_1, \ldots, x_k \rangle) = \langle x_1, \ldots, x_j \rangle.$$

Notation We will assume a canonical encoding of the polynomial time nondeterministic oracle Turing Machines, $N_1, N_2, N_3, \ldots$, with effective composition, etc. Also, we will assume that all polynomials and running times used in this paper are at least $O(n)$ and are monotone increasing.

Definition We write BH(k) and co-BH(k) for the k^{th} levels of the Boolean hierarchy, defined as follows:

$$\begin{aligned} \mathrm{BH}(1) &\stackrel{\mathrm{def}}{=} \mathrm{NP}, \\ \mathrm{BH}(k+1) &\stackrel{\mathrm{def}}{=} \{L_1 - L_2 \mid L_1 \in \mathrm{NP} \text{ and } L_2 \in \mathrm{BH}(k)\}, \\ \text{co-BH}(k) &\stackrel{\mathrm{def}}{=} \{L \mid \overline{L} \in \mathrm{BH}(k)\}. \end{aligned}$$

Definition We write $\mathrm{BH}_3(k)$ and co-$\mathrm{BH}_3(k)$ for the k^{th} levels of the Boolean hierarchy in Δ_3^{P}, defined as follows:

$$\mathrm{BH}_3(1) \stackrel{\mathrm{def}}{=} \mathrm{NP}^{\mathrm{NP}},$$

[1] Mahaney [6] has found an error in the proof presented in [3] and [5] that the set S is in $\mathrm{NP}^{\mathrm{NP}}$. See [4]. The argument presented in this paper does not use the erroneous reasoning and actually proves a stronger result.

$$
\begin{aligned}
\mathrm{BH}_3(k+1) &\stackrel{\mathrm{def}}{=} \{L_1 - L_2 \mid L_1 \in \mathrm{NP}^{\mathrm{NP}} \text{ and } L_2 \in \mathrm{BH}_3(k)\},\\
\text{co-}\mathrm{BH}_3(k) &\stackrel{\mathrm{def}}{=} \{L \mid \overline{L} \in \mathrm{BH}_3(k)\}.
\end{aligned}
$$

An equivalent way to define the BH is as follows [1]:

$$
\begin{aligned}
\mathrm{BH}(1) &\stackrel{\mathrm{def}}{=} \mathrm{NP},\\
\mathrm{BH}(2k) &\stackrel{\mathrm{def}}{=} \{L \mid L = L' \cap \overline{L_2} \text{ where } L' \in \mathrm{BH}(2k-1) \text{ and } L_2 \in \mathrm{NP}\},\\
\mathrm{BH}(2k+1) &\stackrel{\mathrm{def}}{=} \{L \mid L = L' \cup L_2 \text{ where } L' \in \mathrm{BH}(2k) \text{ and } L_2 \in \mathrm{NP}\},\\
\text{co-}\mathrm{BH}(k) &\stackrel{\mathrm{def}}{=} \{L \mid \overline{L} \in \mathrm{BH}(k)\}.
\end{aligned}
$$

From this definition, it is not hard to prove that the following languages are complete for the respective levels of the Boolean hierarchy under polynomial time many-one reductions [1]:

Definition We write $\mathrm{L}_{\mathrm{BH}(k)}$ for the canonical complete language for $\mathrm{BH}(k)$ and $\mathrm{L}_{\text{co-}\mathrm{BH}(k)}$ for the complete language for co-$\mathrm{BH}(k)$:

$$
\begin{aligned}
\mathrm{L}_{\mathrm{BH}(1)} &\stackrel{\mathrm{def}}{=} \mathrm{SAT},\\
\mathrm{L}_{\mathrm{BH}(2k)} &\stackrel{\mathrm{def}}{=} \{\langle x_1,\ldots,x_{2k}\rangle \mid \langle x_1,\ldots,x_{2k-1}\rangle \in \mathrm{L}_{\mathrm{BH}(2k-1)} \text{ and } x_{2k} \in \overline{\mathrm{SAT}}\},\\
\mathrm{L}_{\mathrm{BH}(2k+1)} &\stackrel{\mathrm{def}}{=} \{\langle x_1,\ldots,x_{2k+1}\rangle \mid \langle x_1,\ldots,x_{2k}\rangle \in \mathrm{L}_{\mathrm{BH}(2k)} \text{ or } x_{2k+1} \in \mathrm{SAT}\},\\
\mathrm{L}_{\text{co-}\mathrm{BH}(1)} &\stackrel{\mathrm{def}}{=} \overline{\mathrm{SAT}},\\
\mathrm{L}_{\text{co-}\mathrm{BH}(2k)} &\stackrel{\mathrm{def}}{=} \{\langle x_1,\ldots,x_{2k}\rangle \mid \langle x_1,\ldots,x_{2k-1}\rangle \in \mathrm{L}_{\text{co-}\mathrm{BH}(2k-1)} \text{ or } x_{2k} \in \mathrm{SAT}\},\\
\mathrm{L}_{\text{co-}\mathrm{BH}(2k+1)} &\stackrel{\mathrm{def}}{=} \{\langle x_1,\ldots,x_{2k+1}\rangle \mid \langle x_1,\ldots,x_{2k}\rangle \in \mathrm{L}_{\text{co-}\mathrm{BH}(2k)} \text{ and } x_{2k+1} \in \overline{\mathrm{SAT}}\}.
\end{aligned}
$$

Definition L_{u_2} and L_{u_3} are the canonical $\leq_m^{\mathrm{P}}$-complete languages for Σ_2^{P} and Σ_3^{P} respectively:

$$L_{u_2} \stackrel{\mathrm{def}}{=} \{(N_j, x, 1^i) \mid N_j^{\mathrm{SAT}}(x) \text{ accepts in } \leq |x| + i \text{ steps}\},$$

$$L_{u_3} \stackrel{\mathrm{def}}{=} \{(N_j, x, 1^i) \mid N_j^{L_{u_2}}(x) \text{ accepts in } \leq |x| + i \text{ steps}\}.$$

3 An Example for $\mathrm{BH}(2) = \text{co-}\mathrm{BH}(2)$

In this section, we outline the proof of the main theorem for the case $k = 2$. We want to show that if $\mathrm{BH}(2) = \text{co-}\mathrm{BH}(2)$ then $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{BH}_3(2)$. This case contains the spirit of the proof of the main theorem and allows us to illustrate the proof without worrying about even and odd cases or messy indices.

If $\mathrm{BH}(2) = \text{co-}\mathrm{BH}(2)$ then there is a reduction from $\mathrm{L}_{\mathrm{BH}(2)}$ to $\mathrm{L}_{\text{co-}\mathrm{BH}(2)}$ via some polynomial time function h. So, if $h(F_1, F_2) = (G_1, G_2)$, then

$$F_1 \in \mathrm{SAT} \text{ and } F_2 \in \overline{\mathrm{SAT}} \iff G_1 \in \overline{\mathrm{SAT}} \text{ or } G_2 \in \mathrm{SAT}.$$

The key is that h maps a conjunction to a disjunction. Both conditions of the conjunction are met if just one of the disjuncts is met. In the easy case, if G_2 is satisfiable, then F_1 is satisfiable and F_2 *is not satisfiable.* This gives rise to an NP algorithm for recognizing some of $\overline{\mathrm{SAT}}$: Given any formula F_2, guess a formula F_1 with $|F_1| \leq |F_2|$ and accept if $\pi_2 \circ h(F_1, F_2) \in \mathrm{SAT}$.

Formulas that can be recognized as unsatisfiable by this NP algorithm are said to be *easy.* Formally, a Boolean formula F is easy if $\exists F_1$ with $|F_1| \leq |F|$ and $\pi_2 \circ h(F_1, F) \in \mathrm{SAT}$. If all unsatisfiable formulas are easy, then co-NP = NP. So it is likely that there are *hard* unsatisfiable formulas. We say a formula F is *hard* if

1. $F \in \overline{\mathrm{SAT}}$.

2. $\forall F_1$ with $|F_1| \leq |F|$, $\pi_2 \circ h(F_1, F) \in \overline{\mathrm{SAT}}$.

While the set of hard strings is probably not in NP (note that it is in co-NP), an individual hard formula of length m encodes enough information to allow an NP machine to recognize all of $\overline{\mathrm{SAT}}^{\leq m}$. Let F be a hard formula of length m. Suppose F_1 is any formula of length $\leq m$ and $h(F_1, F) = (G_1, G)$. Since F is hard, we know that $F \in \overline{\mathrm{SAT}}$ and $G \in \overline{\mathrm{SAT}}$. Recall that

$$F_1 \in \mathrm{SAT} \text{ and } F \in \overline{\mathrm{SAT}} \iff G_1 \in \overline{\mathrm{SAT}} \text{ or } G \in \mathrm{SAT}.$$

Replacing $F \in \overline{\mathrm{SAT}}$ with "true" and $G \in \mathrm{SAT}$ with "false", we get

$$F_1 \in \mathrm{SAT} \iff G_1 \in \overline{\mathrm{SAT}},$$

or (by negating both sides of the iff)

$$F_1 \in \overline{\mathrm{SAT}} \iff G_1 \in \mathrm{SAT}.$$

So, given the hard string, F, an NP machine can recognize if $F_1 \in \overline{\mathrm{SAT}}^{\leq m}$ by computing $G_1 = \pi_1 \circ h(F_1, F)$ and verifying that $G_1 \in \mathrm{SAT}$. In other words, a hard formula of length m and the reduction from BH(2) to co-BH(2) induce a reduction from $\overline{\mathrm{SAT}}^{\leq m}$ to SAT.

The approach taken by Kadin [3] was to encode enough information into a sparse set S so that an NP^S machine could get hold of a hard string of a given length or determine that there was none. Then the NP^S machine could recognize $\overline{\mathrm{SAT}}$, implying that co-NP $\subseteq \mathrm{NP}^S$ and that the PH collapses [8].

In this paper we take a slightly different approach to show that the collapse of the BH implies a deeper collapse of the PH. Rather than constructing a sparse oracle that allows an NP machine to recognize $\overline{\mathrm{SAT}}$, we show that there is a smaller amount of information, essentially a tally set, that allows an $\mathrm{NP}^{\mathrm{NP}}$ machine to recognize the complete language for Σ_3^{P}. For the case where BH(2) = co-BH(2), this information is the tally set

$$T \stackrel{\text{def}}{=} \{1^m \mid \exists \text{ a hard formula of length } m\}.$$

First, we show that $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{NP}^{T \oplus \mathrm{NP}}$. Since the set of hard formulas is in co-NP, if we tell an $\mathrm{NP}^{\mathrm{NP}}$ machine that there is a hard formula of length m, it can guess a hard formula and verify with one query that it is hard. With that formula, the $\mathrm{NP}^{\mathrm{NP}}$ machine can produce an NP algorithm that recognizes $\overline{\mathrm{SAT}}^{=m}$. If we tell an $\mathrm{NP}^{\mathrm{NP}}$ machine that there are no hard formulas of length m, then it knows that the "easy" NP algorithm recognizes all of $\overline{\mathrm{SAT}}^{=m}$. In either case, the $\mathrm{NP}^{\mathrm{NP}}$ machine can use an NP algorithm for $\overline{\mathrm{SAT}}^{=m}$ to remove one level of oracle querying from a Σ_3^{P} machine, and therefore recognize any Σ_3^{P} language.

Now, we show that $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[2]}$. Since an $\mathrm{NP}^{\mathrm{NP}}$ machine can guess and verify hard formulas, $T \in \mathrm{NP}^{\mathrm{NP}}$. This implies that a $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine can tell with one query if there are any hard formulas of a given length. Since this is exactly what an $\mathrm{NP}^{\mathrm{NP}}$ machine needs to recognize a Σ_3^{P} language, the $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine can pass the information in one more $\mathrm{NP}^{\mathrm{NP}}$ query and therefore recognize a Σ_3^{P} language with only two queries. Hence $\mathrm{BH}(2) = \text{co-BH}(2)$ implies $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[2]}$, the class of languages recognizable in deterministic polynomial time with two queries to $\mathrm{NP}^{\mathrm{NP}}$.

With more work, we can show that Σ_3^{P} is actually contained in the second level of the Δ_3^{P} Boolean hierarchy.

4 Main Result

We can generalize the analysis of the previous section to higher levels of the BH by replacing the concept of hard formulas with the concept of *hard sequences* of formulas. Just as an individual hard formula could be used with the reduction from BH(2) to co-BH(2) to induce a reduction from $\overline{\mathrm{SAT}}$ to SAT, a hard sequence is a j-tuple that can be used with a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from $\mathrm{BH}(k)$ to co-$\mathrm{BH}(k)$ to define a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from $\mathrm{BH}(k-j)$ to co-$\mathrm{BH}(k-j)$.

Definition Suppose $\mathrm{L}_{\mathrm{BH}(k)} \leq_{\mathrm{m}}^{\mathrm{P}} \mathrm{L}_{\text{co-BH}(k)}$ via some polynomial time function h. Then, we call $\langle 1^m, x_1, \ldots, x_j\rangle$ a *hard sequence* with respect to h if $j = 0$ or if all of the following hold:

1. $1 \leq j \leq k-1$.
2. $|x_j| \leq m$.
3. $x_j \in \overline{\mathrm{SAT}}$.
4. $\langle 1^m, x_1, \ldots, x_{j-1}\rangle$ is a hard sequence with respect to h.
5. For all $y_1, \ldots, y_\ell \in \Sigma^{\leq m}$ (where $\ell = k - j$)

$$\pi_{\ell+1} \circ h(\langle y_1, \ldots, y_\ell, x_j, \ldots, x_1\rangle) \in \overline{\mathrm{SAT}}.$$

If $\langle 1^m, x_1, \ldots, x_j\rangle$ is a hard sequence, then we refer to j as the *order* of the sequence and say that it is a hard sequence for length m. Also, we will call a hard sequence *maximal* if it cannot be extended to a hard sequence of higher order. We say that j is the maximum order for length m, if there is a hard sequence of order j for length m and there is no hard sequence of order $j+1$ for length m. Finally, when the individual strings $x_1, \ldots, x_j$ are of no importance, we use the shortened notation $\langle 1^m, \vec{x}\rangle$ instead of $\langle 1^m, x_1, \ldots, x_j\rangle$.

Our proof that $\mathrm{BH} \subseteq \mathrm{BH}(k)$ implies $\mathrm{PH} \subseteq \mathrm{BH}_3(k)$ is rather involved. All of our lemmas and theorems start with the assumption that $\mathrm{BH}(k) = \text{co-}\mathrm{BH}(k)$, or in other words, that there exists a function h that is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from $\mathrm{L}_{\mathrm{BH}(k)}$ to $\mathrm{L}_{\text{co-BH}(k)}$. First we show that a hard sequence of order j for length m does indeed induce a reduction from $\mathrm{L}_{\mathrm{BH}(k-j)}$ to $\mathrm{L}_{\text{co-BH}(k-j)}$ (Lemma 1). Then we show that a maximal hard sequence for length m induces a polynomial time reduction from $\overline{\mathrm{SAT}}^{\leq m}$ to SAT (Lemma 2). In Lemma 3 we argue that given a maximal hard sequence, an NP machine can recognize an initial segment of L_{u_2}, the canonical complete language for Σ_2^{P}. Lemma 4 takes this a step further by showing that given the maximum order of hard sequences for a length, an $\mathrm{NP}^{\mathrm{NP}}$ machine can recognize an initial segment of L_{u_3}, the canonical complete language for Σ_3^{P}. We then define the set T which encodes the orders of hard sequences for each length, and we show that $T \in \mathrm{NP}^{\mathrm{NP}}$ (Lemma 5). We put all this analysis together in Theorem 6 and show that $\mathrm{BH}(k) = \text{co-}\mathrm{BH}(k)$ implies $\mathrm{PH} \subseteq \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[k]}$.

Moving toward the Δ_3^{P} Boolean hierarchy, we prove that an NP machine can recognize if there is a hard sequence of order j for length m if it is given a hard sequence for a polynomially longer length (Lemma 7). In Lemma 8 we show that the maximum order of hard sequences for a length is enough information to permit an $\mathrm{NP}^{\mathrm{NP}}$ machine to recognize when a string *is not in* L_{u_3}; that is, the $\mathrm{NP}^{\mathrm{NP}}$ machine can recognize an initial segment of a complete language for Π_3^{P}. Finally, this gives us the machinery to prove our main theorem.

We start by showing that a hard sequence of order j for length m induces a reduction from $\mathrm{L}_{\mathrm{BH}(k-j)}$ to $\mathrm{L}_{\text{co-BH}(k-j)}$ for tuples of strings up to length m.

Lemma 1 Suppose $\mathrm{L}_{\mathrm{BH}(k)} \leq_{\mathrm{m}}^{\mathrm{P}} \mathrm{L}_{\text{co-BH}(k)}$ via some function h and $\langle 1^m, x_1, \ldots, x_j\rangle$ is a hard sequence with respect to h. Then for all $y_1, \ldots, y_\ell \in \Sigma^{\leq m}$ (where $\ell = k - j$)

$$\langle y_1, \ldots, y_\ell\rangle \in \mathrm{L}_{\mathrm{BH}(\ell)} \iff \pi_{(1,\ell)} \circ h(\langle y_1, \ldots, y_\ell, x_j, \ldots, x_1\rangle) \in \mathrm{L}_{\text{co-BH}(\ell)}.$$

Proof (by induction on j)

<u>Induction Hypothesis $P(j)$:</u> For all $y_1, \ldots, y_{k-j} \in \Sigma^{\leq m}$

$$\langle y_1, \ldots, y_{k-j}\rangle \in \mathrm{L}_{\mathrm{BH}(k-j)} \iff \pi_{(1,k-j)} \circ h(\langle y_1, \ldots, y_{k-j}, x_j, \ldots, x_1\rangle) \in \mathrm{L}_{\text{co-BH}(k-j)}.$$

<u>Base Case $P(0)$:</u> By the hypothesis of the lemma, h reduces $\mathrm{L}_{\mathrm{BH}(k)}$ to $\mathrm{L}_{\text{co-BH}(k)}$, so

$$\langle y_1, \ldots, y_k\rangle \in \mathrm{L}_{\mathrm{BH}(k)} \iff h(\langle y_1, \ldots, y_k\rangle) \in \mathrm{L}_{\text{co-BH}(k)}.$$

However, $\pi_{(1,k)} \circ h(\langle y_1, \ldots, y_k\rangle) = h(\langle y_1, \ldots, y_k\rangle)$, so

$$\langle y_1, \ldots, y_k\rangle \in \mathrm{L}_{\mathrm{BH}(k)} \iff \pi_{(1,k)} \circ h(\langle y_1, \ldots, y_k\rangle) \in \mathrm{L}_{\text{co-BH}(k)}.$$

Induction Case $P(j+1)$: Suppose $P(j)$ holds. Let $\ell = k-j$. Let $\langle 1^m, x_1, \ldots, x_{j+1}\rangle$ be a hard sequence. By the induction hypothesis, for all $y_1, \ldots, y_{\ell-1} \in \Sigma^{\leq m}$

$$\langle y_1, \ldots, y_{\ell-1}, x_{j+1}\rangle \in \mathrm{L}_{\mathrm{BH}(\ell)} \iff \pi_{(1,\ell)}\circ h(\langle y_1, \ldots, y_{\ell-1}, x_{j+1}, \ldots, x_1\rangle) \in \mathrm{L}_{\text{co-BH}(\ell)}.$$

If ℓ is even, then by the definitions of $\mathrm{L}_{\mathrm{BH}(\ell)}$ and $\mathrm{L}_{\text{co-BH}(\ell)}$

$$\begin{array}{lcl} \langle y_1, \ldots, y_{\ell-1}\rangle \in \mathrm{L}_{\mathrm{BH}(\ell-1)} & & \pi_{(1,\ell-1)}\circ h(\langle y_1, \ldots, y_{\ell-1}, x_{j+1}, \ldots, x_1\rangle) \in \mathrm{L}_{\text{co-BH}(\ell-1)} \\ \text{and } \ x_{j+1} \in \overline{\mathrm{SAT}} & \iff & \text{or } \ \pi_\ell\circ h(\langle y_1, \ldots, y_{\ell-1}, x_{j+1}, \ldots, x_1\rangle) \in \mathrm{SAT}. \end{array} \tag{1}$$

If ℓ is odd, then by the definitions of $\mathrm{L}_{\mathrm{BH}(\ell)}$ and $\mathrm{L}_{\text{co-BH}(\ell)}$

$$\begin{array}{lcl} \langle y_1, \ldots, y_{\ell-1}\rangle \in \mathrm{L}_{\mathrm{BH}(\ell-1)} & & \pi_{(1,\ell-1)}\circ h(\langle y_1, \ldots, y_{\ell-1}, x_{j+1}, \ldots, x_1\rangle) \in \mathrm{L}_{\text{co-BH}(\ell-1)} \\ \text{or } \ x_{j+1} \in \mathrm{SAT} & \iff & \text{and } \ \pi_\ell\circ h(\langle y_1, \ldots, y_{\ell-1}, x_{j+1}, \ldots, x_1\rangle) \in \overline{\mathrm{SAT}}, \end{array} \tag{2}$$

or (by negating both sides of the iff),

$$\begin{array}{lcl} \langle y_1, \ldots, y_{\ell-1}\rangle \notin \mathrm{L}_{\mathrm{BH}(\ell-1)} & & \pi_{(1,\ell-1)}\circ h(\langle y_1, \ldots, y_{\ell-1}, x_{j+1}, \ldots, x_1\rangle) \notin \mathrm{L}_{\text{co-BH}(\ell-1)} \\ \text{and } \ x_{j+1} \in \overline{\mathrm{SAT}} & \iff & \text{or } \ \pi_\ell\circ h(\langle y_1, \ldots, y_{\ell-1}, x_{j+1}, \ldots, x_1\rangle) \in \mathrm{SAT}. \end{array} \tag{3}$$

Since $\langle 1^m, x_1, \ldots, x_{j+1}\rangle$ is a hard sequence, from parts 3 and 5 of the definition of a hard sequence we know that $x_{j+1} \in \overline{\mathrm{SAT}}$ and

$$\pi_\ell\circ h(\langle y_1, \ldots, y_{\ell-1}, x_{j+1}, \ldots, x_1\rangle) \in \overline{\mathrm{SAT}}.$$

Therefore in equations (1) and (3), the second conjunct on the left side is true and the second disjunct on the right side is false. Hence

$$\langle y_1, \ldots, y_{\ell-1}\rangle \in \mathrm{L}_{\mathrm{BH}(\ell-1)} \iff \pi_{(1,\ell-1)}\circ h(\langle y_1, \ldots, y_{\ell-1}, x_{j+1}, \ldots, x_1\rangle) \in \mathrm{L}_{\text{co-BH}(\ell-1)}.$$

Then replacing $k-j$ for ℓ, we have

$$\langle y_1, \ldots, y_{k-(j+1)}\rangle \in \mathrm{L}_{\mathrm{BH}(k-(j+1))}$$
$$\iff$$
$$\pi_{(1,k-(j+1))}\circ h(\langle y_1, \ldots, y_{k-(j+1)}, x_{j+1}, \ldots, x_1\rangle) \in \mathrm{L}_{\text{co-BH}(k-(j+1))}.$$

So, we have established the induction hypothesis $P(j+1)$. □

Lemma 2 shows that a maximal hard sequence for length m induces a polynomial time reduction from $\overline{\mathrm{SAT}}^{\leq m}$ to SAT, or in other words, given a maximal hard sequence for length m, an NP machine can recognize $\overline{\mathrm{SAT}}^{\leq m}$.

Lemma 2 Suppose $\mathrm{L}_{\mathrm{BH}(k)} \leq^{\mathrm{P}}_{\mathrm{m}} \mathrm{L}_{\text{co-BH}(k)}$ via some function h and $\langle 1^m, x_1, \ldots, x_j\rangle$ is a maximal hard sequence with respect to h. Then for all $y \in \Sigma^{\leq m}$

$$y \in \overline{\mathrm{SAT}}$$
$$\iff$$
$$\exists\, y_1, \ldots, y_{\ell-1} \in \Sigma^{\leq m} \quad \pi_\ell\circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \in \mathrm{SAT}$$

(where $\ell = k-j$).

Proof

If $j = k-1$ ($\langle y_1, \ldots, y_{\ell-1}\rangle$ is the empty sequence), then, by Lemma 1, for all $y \in \Sigma^{\leq m}$

$$y \in \mathrm{BH}(1) \iff \pi_1 \circ h(\langle y, x_j, \ldots, x_1\rangle) \in \text{co-BH}(1).$$

However, $\mathrm{BH}(1) = \mathrm{SAT}$ and $\text{co-BH}(1) = \overline{\mathrm{SAT}}$. So, we have

$$y \in \mathrm{SAT} \iff \pi_1 \circ h(\langle y, x_j, \ldots, x_1\rangle) \in \overline{\mathrm{SAT}}.$$

or (by negating both sides of the iff)

$$y \in \overline{\mathrm{SAT}} \iff \pi_1 \circ h(\langle y, x_j, \ldots, x_1\rangle) \in \mathrm{SAT}.$$

Thus, the lemma holds when $j = k-1$ (i.e. when $y_1, \ldots, y_{\ell-1}$ is the empty sequence).

Consider the case when $j < k-1$.

($\Rightarrow$) Suppose $y \in \overline{\mathrm{SAT}}$. Since $\langle 1^m, x_1, \ldots, x_j\rangle$ is maximal, $\langle 1^m, x_1, \ldots, x_j, y\rangle$ is not a hard sequence. However, $j+1 \leq k-1$, $|y| \leq m$, $y \in \overline{\mathrm{SAT}}$, and $\langle 1^m, x_1, \ldots, x_j\rangle$ is a hard sequence. So, $\langle 1^m, x_1, \ldots, x_j, y\rangle$ must fail to be a hard sequence by failing to satisfy condition 5 of the definition of hard sequences. Thus,

$$\exists\ y_1, \ldots, y_{\ell-1} \in \Sigma^{\leq m} \quad \pi_\ell \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \in \mathrm{SAT}.$$

($\Leftarrow$) Suppose that for some $y_1, \ldots, y_{\ell-1} \in \Sigma^{\leq m}$

$$\pi_\ell \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \in \mathrm{SAT}.$$

Since $\langle x_1, \ldots, x_j\rangle$ is a hard sequence for length m, by Lemma 1

$$\langle y_1, \ldots, y_{\ell-1}, y\rangle \in \mathrm{L}_{\mathrm{BH}(\ell)} \iff \pi_{(1,\ell)} \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \in \mathrm{L}_{\text{co-BH}(\ell)}.$$

If ℓ is even, then by the definitions of $\mathrm{L}_{\mathrm{BH}(\ell)}$ and $\mathrm{L}_{\text{co-BH}(\ell)}$

$$\begin{array}{l} \langle y_1, \ldots, y_{\ell-1}\rangle \in \mathrm{L}_{\mathrm{BH}(\ell-1)} \\ \text{and} \quad y \in \overline{\mathrm{SAT}} \end{array} \iff \begin{array}{l} \pi_{(1,\ell-1)} \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \in \mathrm{L}_{\text{co-BH}(\ell-1)} \\ \text{or} \quad \pi_\ell \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \in \mathrm{SAT}. \end{array} \tag{4}$$

If ℓ is odd, then by the definitions of $\mathrm{L}_{\mathrm{BH}(\ell)}$ and $\mathrm{L}_{\text{co-BH}(\ell)}$

$$\begin{array}{l} \langle y_1, \ldots, y_{\ell-1}\rangle \in \mathrm{L}_{\mathrm{BH}(\ell-1)} \\ \text{or} \quad y \in \mathrm{SAT} \end{array} \iff \begin{array}{l} \pi_{(1,\ell-1)} \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \in \mathrm{L}_{\text{co-BH}(\ell-1)} \\ \text{and} \quad \pi_\ell \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \in \overline{\mathrm{SAT}}, \end{array} \tag{5}$$

or (by negating both sides of the iff)

$$\begin{array}{l} \langle y_1, \ldots, y_{\ell-1}\rangle \notin \mathrm{L}_{\mathrm{BH}(\ell-1)} \\ \text{and} \quad y \in \overline{\mathrm{SAT}} \end{array} \iff \begin{array}{l} \pi_{(1,\ell-1)} \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \notin \mathrm{L}_{\text{co-BH}(\ell-1)} \\ \text{or} \quad \pi_\ell \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \in \mathrm{SAT}. \end{array} \tag{6}$$

In either case, we already know by hypothesis that

$$\pi_\ell \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1\rangle) \in \mathrm{SAT},$$

so the right sides of the iff in equations (4) and (6) are satisfied. Therefore, the left sides of equations (4) and (6) must also be satisfied, and we have $y \in \overline{\mathrm{SAT}}$. □

Lemma 2 essentially states that a maximal hard sequence produces a way to witness that a formula is unsatisfiable. Hence, given a maximal hard sequence, an NP machine can guess these witnesses and verify that formulas up to a certain length are unsatisfiable. But if an NP machine can verify that formulas are unsatisfiable, it can simulate an $\mathrm{NP}^{\mathrm{NP}}$ computation by guessing the answer to each NP query and verifying that its answer is correct. We use this idea to prove Lemma 3 which states that given a maximal hard sequence, an NP machine can recognize an initial segment of L_{u_2}, the canonical complete language for Σ_2^{P}.

Lemma 3 Suppose h is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from $\mathrm{L}_{\mathrm{BH}(k)}$ to $\mathrm{L}_{\text{co-BH}(k)}$. Then there exists an NP machine $N_{\sigma 2}$ and a polynomial $p_{\sigma 2}$ such that if $m \geq p_{\sigma 2}(|w|)$ and $\langle 1^m, x_1, \ldots, x_j \rangle$ is a maximal hard sequence w.r.t. h, then

$$w \in L_{u_2} \iff N_{\sigma 2}(w, \langle 1^m, x_1, \ldots, x_j \rangle) \text{ accepts.}$$

Proof

Let $L_{u_2} = L(N_{u_2}^{\mathrm{SAT}})$. Define $p_{\sigma 2}(n)$ to be the upper bound on the running time of N_{u_2} on inputs of length n. Obviously, $N_{u_2}(w)$ queries only strings of length $\leq m$, since $m \geq p_{\sigma 2}(|w|)$. On input $(w, \langle 1^m, x_1, \ldots, x_j \rangle)$, $N_{\sigma 2}$ does the following:

1. Simulate N_{u_2} step by step until N_{u_2} makes an oracle query to SAT.

2. When N_{u_2} queries "$y \in \mathrm{SAT}$?", branch into two computations. One guesses that $y \in \mathrm{SAT}$, the other guesses that $y \in \overline{\mathrm{SAT}}$.

3. The branch that guesses $y \in \mathrm{SAT}$, will guess a satisfying assignment for y. If none are found, all computations along this branch terminate. If a satisfying assignment is found, then the guess that $y \in \mathrm{SAT}$ is correct and the simulation continues.

4. The branch that guesses $y \in \overline{\mathrm{SAT}}$ will use the maximal hard sequence to find a witness for $y \in \overline{\mathrm{SAT}}$. I.e., it guesses $\ell - 1$ strings $y_1, \ldots, y_{\ell-1} \in \Sigma^{\leq m}$, computes

   $$F = \pi_\ell \circ h(\langle y_1, \ldots, y_{\ell-1}, y, x_j, \ldots, x_1 \rangle),$$

   and guesses a satisfying assignment for F. If none are found, all computations along this branch terminate. Otherwise, the guess that $y \in \overline{\mathrm{SAT}}$ is correct and the simulation continues.

By Lemma 2, if $\langle 1^m, x_1, \ldots, x_j \rangle$ is a maximal hard sequence, then for each query y, $y \in \overline{\mathrm{SAT}}$ if and only if some computation in step 4 finds a satisfiable F. So, in the simulation of the oracle query, all the computations along one branch will terminate and some computations in the other branch will continue. Thus, the simulation continues if and only if the guesses for the oracle answers (either $y \in \text{SAT}$ or $y \in \overline{\text{SAT}}$) are verified, and hence

$$w \in L(N_{u_2}^{\text{SAT}}) \iff N_{\sigma_2}(w, \langle 1^m, x_1, \ldots, x_j \rangle) \text{ accepts.}$$

□

Taking the spirit of Lemma 3 one step further, we show that, with the help of a maximal hard sequence, an NP^{NP} machine can simulate a Σ_3^{P} machine. In addition, an NP^{NP} machine can guess and verify hard sequences, so it does not need to be given a maximal hard sequence, all it really needs is the *maximum order*. Therefore there exists an NP^{NP} machine which, given the maximum order of hard sequences for a length, can recognize initial segments of L_{u_3}, the complete language for Σ_3^{P}.

Lemma 4 Suppose h is a $\leq_{\text{m}}^{\text{P}}$-reduction from $\text{L}_{\text{BH}(k)}$ to $\text{L}_{\text{co-BH}(k)}$. There exists an NP^{SAT} machine N_{σ_3} and a polynomial p_{σ_3} such that for any $m \geq p_{\sigma_3}(|w|)$, if j is the maximum order for length m w.r.t. h, then

$$w \in L_{u_3} \iff N_{\sigma_3}^{\text{SAT}}(w, j, 1^m) \text{ accepts.}$$

Furthermore, if j is greater than the maximum order for length m w.r.t. h,

$$\forall w \quad N_{\sigma_3}^{\text{SAT}}(w, j, 1^m) \text{ rejects.}$$

Proof

Let $L_{u_3} = L(N_{u_3}^{L_{u_2}})$, where $L_{u_2} = L(N_{u_2}^{\text{SAT}})$ is the canonical complete language for Σ_2^{P}. Let $r(n)$ be a polynomial upper bound on the running time of N_{u_3} on inputs of length n. Clearly, $N_{u_3}(w)$ will only query strings of length $\leq r(n)$, where $n = |w|$. Apply Lemma 3 to obtain N_{σ_2} and the polynomial p_{σ_2}. Let $p_{\sigma_3}(n) \stackrel{\text{def}}{=} p_{\sigma_2}(r(n))$.

The critical observation to make here is that the set of hard sequences is in co-NP. (This is obvious from the definition of hard sequences.) So, given j, the maximum order for length $m \geq p_{\sigma_3}(n)$, an NP^{NP} machine can guess j strings $x_1, \ldots, x_j \in \Sigma^{\leq m}$ and ask the NP oracle if $\langle 1^m, x_1, \ldots, x_j \rangle$ forms a hard sequence. If $\langle 1^m, x_1, \ldots, x_j \rangle$ does form a hard sequence, then it must also be a maximal sequence since it is of maximum order. Now, $N_{\sigma_3}^{\text{SAT}}(w, j, 1^m)$ can simulate $N_{u_3}^{L_{u_2}}(w)$ step by step, and when N_{u_3} queries "$y \in L_{u_2}$?", N_{σ_3} will ask "$(y, \langle 1^m, x_1, \ldots, x_j \rangle) \in L(N_{\sigma_2})$". By Lemma 3, the two queries will return with the same answers, so

$$w \in L(N_{u_3}^{L_{u_2}}) \iff N_{\sigma_3}^{\text{SAT}}(w, j, 1^m) \text{ accepts.}$$

Note that when N_{σ_3} guesses the hard sequence $\langle 1^m, x_1, \ldots, x_j \rangle$, several computation paths of the NP machine may survive, because there may be many hard sequences of maximum order. However, uniqueness is not important here because any maximal hard sequence will work for N_{σ_2}. So, all the computation branches that manage to guess a hard sequence of maximum order will have the same acceptance behavior. Furthermore, if j is greater than the maximum order, then none of the computation paths survive because there are no hard sequences of order j for length m. Thus, in this case, $N_{\sigma_3}(w, j, 1^m)$ will reject. $\square$

We have shown that maximal hard sequences and maximum orders expand the computational power of nondeterministic machines. We define the set T to be the set of strings encoding the orders of hard sequences for each length.

Definition Suppose h is a $\leq_m^P$-reduction from $L_{BH(k)}$ to $L_{co\text{-}BH(k)}$. We define an associated set T by

$$T \stackrel{\text{def}}{=} \{(1^m, j) \mid \exists x_1, \ldots, x_j \in \Sigma^{\leq m}, \text{s.t. } \langle 1^m, x_1, \ldots, x_j \rangle \text{ is a hard sequence.}\}$$

Note that since the set of hard sequences is in co-NP, T itself is in $\mathrm{NP}^{\mathrm{NP}}$. This gives us the following lemma.

Lemma 5 Suppose h is a $\leq_m^P$-reduction from $L_{BH(k)}$ to $L_{co\text{-}BH(k)}$, then the set T defined above is in $\mathrm{NP}^{\mathrm{NP}}$. $\square$

Since $T \in \mathrm{NP}^{\mathrm{NP}}$, a $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine can compute the order of the maximum hard sequence for length m with $k-1$ queries. The $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine can then pass this number to the $\mathrm{NP}^{\mathrm{NP}}$ machine $N_{\sigma_3}^{\mathrm{SAT}}$ of Lemma 4 to recognize L_{u_3}, the complete language for Σ_3^P. Therefore if the BH collapses to its k^{th} level, Lemmas 4 and 5 imply that the PH collapses to $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[k]}$. This collapse of the polynomial time hierarchy implied by the collapse of the BH is lower than previously known.

Theorem 6 Suppose h is a $\leq_m^P$-reduction from $L_{BH(k)}$ to $L_{co\text{-}BH(k)}$. Then there exists a $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine which accepts L_{u_3} with only k queries to the $\mathrm{NP}^{\mathrm{NP}}$ oracle. That is, the polynomial hierarchy collapses to $\mathrm{P}^{(\mathrm{NP}^{\mathrm{NP}})[k]}$.

Proof

By Lemma 4, there exists N_{σ_3} and p_{σ_3} such that if j is the maximum order for length m and $m \geq p_{\sigma_3}(|w|)$, then

$$w \in L_{u_3} \iff N_{\sigma_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts.}$$

Using the fact that T is in $\mathrm{NP}^{\mathrm{NP}}$ (Lemma 5), a $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine can determine if $(1^m, \ell)$ is in T by asking the oracle. Doing this for all values of ℓ between 1 and $k-1$, it can determine the maximum ℓ such that $(1^m, \ell)$ is in T. This maximum ℓ, call it j, is of course the maximum order for length m. Then with one final query, the $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ machine asks if

$$N_{\sigma_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts.}$$

If the oracle answers "yes", the machine accepts. Otherwise, it rejects. $\square$

Note that we could make Theorem 6 stronger by using binary search instead of linear search to find the maximum order. However, we will push the collapse even further in Theorem 9, so our inquiry will follow a new direction.

The following lemma states that an NP machine can recognize if there is a hard sequence of order j for length m if it is given a maximal hard sequence for a longer length.

Lemma 7 Suppose h is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from $\mathrm{L}_{\mathrm{BH}(k)}$ to $\mathrm{L}_{\text{co-BH}(k)}$. There exists an NP machine N_t and a polynomial p_t such that if $\langle 1^{m_2}, \vec{x}\,\rangle$ is a maximal hard sequence w.r.t. h and $m_2 \geq p_t(m_1+k)$, then

$$(1^{m_1}, j_1) \in T \iff N_t((1^{m_1}, j_1), \langle 1^{m_2}, \vec{x}\,\rangle) \text{ accepts.}$$

Proof

Use Lemmas 3 and 5. □

In Lemma 4, we showed that, with the help of the maximum order, an $\mathrm{NP}^{\mathrm{NP}}$ machine can recognize a complete language for Σ_3^{P}. In the next lemma we show that with the help of the maximum order, an $\mathrm{NP}^{\mathrm{NP}}$ machine can also recognize a complete language for Π_3^{P} (i.e. recognize when a string is *not* in L_{u_3}).

Lemma 8 Suppose h is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from $\mathrm{L}_{\mathrm{BH}(k)}$ to $\mathrm{L}_{\text{co-BH}(k)}$. Let L_{u_3} be the canonical complete language for Σ_3^{P}. There exists an $\mathrm{NP}^{\mathrm{SAT}}$ machine N_{π_3} and a polynomial p_{π_3} such that for any $m \geq p_{\pi_3}(|w|)$, if j is the maximum order for length m w.r.t. h, then

$$w \in \overline{L_{u_3}} \iff N_{\pi_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts.}$$

Furthermore, if j is greater than the maximum order for length m w.r.t. h,

$$\forall w \quad N_{\pi_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ rejects.}$$

Proof

Let $L_{u_3} = L(N_{u_3}^{L_{u_2}})$ where L_{u_2} is the canonical complete language for Σ_2^{P}. By Lemma 4, there exist N_{σ_3} and p_{σ_3} such that if j_1 is the maximum order for length m_1, and $m_1 \geq p_{\sigma_3}(|w|)$, then

$$w \in L_{u_3} \iff N_{\sigma_3}^{\mathrm{SAT}}(w, j_1, 1^{m_1}) \text{ accepts.}$$

The language accepted by $N_{\sigma_3}^{\mathrm{SAT}}$ is in Σ_2^{P}, so we can reduce it to L_{u_2} via some polynomial time function g. Let $r(n)$ be an upper bound on the running time of g. Using Lemma 3, there exist N_{σ_2} and p_{σ_2}, such that if $\langle 1^{m_2}, \vec{y}\,\rangle$ is a maximal hard sequence and $m_2 \geq p_{\sigma_2}(r(|w|+k+m_1))$, then

$$N_{\sigma_3}^{\mathrm{SAT}}(w, j_1, 1^{m_1}) \text{ accepts} \iff N_{\sigma_2}(g(w, j_1, 1^{m_1}), \langle 1^{m_2}, \vec{y}\,\rangle) \text{ accepts.}$$

Let N_s be the NP machine that runs the reduction g and then simulates N_{σ_2}, i.e.,

$$N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\,\rangle) \text{ accepts.} \iff N_{\sigma_2}(g(w, j_1, 1^{m_1}), \langle 1^{m_2}, \vec{y}\,\rangle) \text{ accepts.}$$

Let $p_s \stackrel{\text{def}}{=} p_{\sigma_2} \circ r$. Now, if $m_1 \geq p_{\sigma_3}(|w|)$, $m_2 \geq p_s(|w| + k + m_1)$, j_1 is the maximum order for length m_1 and $\langle 1^{m_2}, \vec{y}\,\rangle$ is a maximal hard sequence, then

$$w \in L_{u_3} \iff N_{\sigma_3}^{\text{SAT}}(w, j_1, 1^{m_1}) \text{ accepts} \iff N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\,\rangle) \text{ accepts.} \tag{7}$$

We are trying to prove that there exists a machine $N_{\pi_3}^{\text{SAT}}$ that accepts $(w, j, 1^m)$ if $w \notin L_{u_3}$ when m is big enough in relation to $|w|$ and j is the maximum order of the hard sequences for length m. The $N_{\pi_3}^{\text{SAT}}$ that we have in mind will map

$$(w, j, 1^m) \longrightarrow (w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\,\rangle)$$

and accept iff $N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\,\rangle)$ rejects (iff $w \notin L_{u_3}$). $N_{\pi_3}^{\text{SAT}}$ can tell if $N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\,\rangle)$ rejects with one query to SAT.

The difficulty in mapping $(w, j, 1^m) \longrightarrow (w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\,\rangle)$ lies in the fact that $N_{\pi_3}^{\text{SAT}}$ is given j, the maximum order of hard sequences for one length m, and it must compute the maximum orders of two other lengths, m_1 and m_2. We will define p_{π_3} so that if $m \geq p_{\pi_3}(|w|)$, then m will be bigger enough than both m_1 and m_2 so that we can apply Lemma 7 to compute j_1 and j_2.

Let $p_{\pi_3}(n) \stackrel{\text{def}}{=} p_t(p_s(n + k + p_{\sigma_3}(n)) + k)$, i.e. $p_{\pi_3}(n) = p_t(m_2 + k)$ where $m_2 = p_s(n + k + m_1)$ and $m_1 = p_{\sigma_3}(n)$ (recall p_t is the polynomial bound from Lemma 7).

$N_{\pi_3}^{\text{SAT}}(w, j, 1^m)$ will do the following. (We will annotate the program with a description of what $N_{\pi_3}^{\text{SAT}}(w, j, 1^m)$ accomplishes when j is the maximum order.)

1. Reject if $m < p_{\pi_3}(|w|)$.

2. Guess j strings $x_1, \ldots, x_j \in \Sigma^{\leq m}$ and confirm that $\langle 1^m, x_1, \ldots, x_j\rangle = \langle 1^m, \vec{x}\,\rangle$ is a hard sequence by asking the SAT oracle. (Recall that checking if a given tuple forms a hard sequence is a co-NP question.) If j is the maximum order and $\langle 1^m, \vec{x}\,\rangle$ is a hard sequence, then $\langle 1^m, \vec{x}\,\rangle$ is a maximal hard sequence, too.

3. Let $n = |w|$. Compute $m_1 = p_{\sigma_3}(n)$ and $m_2 = p_s(n + k + m_1)$.

4. For $\ell = 0$ to $k - 1$ ask SAT if $N_t((1^{m_1}, \ell), \langle 1^m, \vec{x}\,\rangle)$ accepts. Let j_1 be the maximum ℓ where $N_t((1^{m_1}, \ell), \langle 1^m, \vec{x}\,\rangle)$ does accept. Note that $m = p_t(m_2 + k) \geq p_t(m_1 + k)$ so j_1 is the maximum order for length m_1 (by Lemma 7) if j is the maximum order for length m.

5. For $\ell = 0$ to $k - 1$ ask SAT if $N_t((1^{m_2}, \ell), \langle 1^m, \vec{x}\,\rangle)$ accepts. Let j_2 be the maximum ℓ where $N_t((1^{m_2}, \ell), \langle 1^m, \vec{x}\,\rangle)$ does accept. As in step 4, j_2 is the maximum order for length m_2 if j is the maximum order for length m.

6. Guess j_2 strings $y_1, \ldots, y_{j_2} \in \Sigma^{\leq m_2}$ and confirm that $\langle 1^{m_2}, y_1, \ldots, y_{j_2}\rangle = \langle 1^{m_2}, \vec{y}\,\rangle$ is a hard sequence (with one query to SAT). Note that if $\langle 1^{m_2}, \vec{y}\,\rangle$ is a hard sequence and j_2 is the maximum order, then $\langle 1^{m_2}, \vec{y}\,\rangle$ is also a maximal hard sequence.

7. Ask SAT if $N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\rangle)$ accepts. If SAT returns "no", then $N_{\pi_3}^{\mathrm{SAT}}$ accepts. Note that by the preceding discussion, if j is the maximum order for length m, then j_1 is the maximum order for length m_1, and $\langle 1^{m_2}, \vec{y}\rangle$ is a maximal hard sequence. Also, $m_1 = p_{\sigma_3}(|w|)$ and $m_2 = p_s(|w| + k + m_1)$, so by equation 7

$$w \in L_{u_3} \iff N_s(w, j_1, 1^{m_1}, \langle 1^{m_2}, \vec{y}\rangle) \text{ accepts.}$$

Now, we argue that if j is the maximum order for length m and $m \geq p_{\pi_3}(|w|)$, then

$$w \in \overline{L_{u_3}} \iff N_{\pi_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts.}$$

First of all, $N_{\pi_3}^{\mathrm{SAT}}$ accepts in step 7 only. So, if $w \in L_{u_3}$, all computation paths of $N_{\pi_3}^{\mathrm{SAT}}$ reject—even those that reach step 7, because SAT would answer "yes" in step 7. On the other hand, if $w \in \overline{L_{u_3}}$ then some computation path will reach step 7, get "no" from the SAT oracle and accept.

Finally, we note that if j is greater than the maximum order for length m, then no computation path will survive step 2. Thus, in this case $N_{\pi_3}^{\mathrm{SAT}}(w, j, 1^m)$ rejects. □

Now we are ready to prove our main theorem. This theorem demonstrates a close linkage between the collapse of the Boolean hierarchy and the polynomial time hierarchy.

Theorem 9 Suppose h is a $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduction from $\mathrm{L}_{\mathrm{BH}(k)}$ to $\mathrm{L}_{\text{co-BH}(k)}$. Let L_{u_3} be the canonical complete language for Σ_3^{P}. Then there exist languages $B_1, \ldots, B_k \in \mathrm{NP}^{\mathrm{NP}}$ such that

$$L_{u_3} = B_1 - (B_2 - (B_3 - (\cdots - B_k))).$$

That is, $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{BH}_3(k)$, and therefore $\mathrm{PH} \subseteq \mathrm{BH}_3(k)$.

Proof

First, recall that in Lemmas 4 and 8 it was shown that $N_{\sigma_3}^{\mathrm{SAT}}$ and $N_{\pi_3}^{\mathrm{SAT}}$ accepted L_{u_3} and $\overline{L_{u_3}}$ (respectively) with the help of the maximum order for a large enough length (and they reject if the number given for the maximum order is too large). Let w be any string. Let $m = \max(p_{\sigma_3}(|w|), p_{\pi_3}(|w|))$; then m is large enough so that if j is the maximum order for length m,

$$N_{\sigma_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts} \iff w \in L_{u_3},$$

$$N_{\pi_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ accepts} \iff w \notin L_{u_3}.$$

We will define the $\mathrm{NP}^{\mathrm{NP}}$ languages $B_1, \ldots, B_k$ to be the strings accepted by $\mathrm{NP}^{\mathrm{NP}}$ machines that try to guess j, the maximum order for length m, and then run N_{σ_3} and N_{π_3}. These $\mathrm{NP}^{\mathrm{NP}}$ machines cannot verify when they have guessed the true maximum order, instead they will base their acceptance behavior on whether they can determine that an earlier machine in the sequence may have been fooled by an incorrect guess for j. This successive approximation scheme converges to the language L_{u_3} within k steps.

Definition For $1 \leq \ell \leq k$ the language B_ℓ is the set of strings w with the property that there exist $j_1, \ldots, j_\ell$ such that

1. $0 \leq j_1 < j_2 < \ldots < j_\ell \leq k-1$.
2. for all odd d, $1 \leq d \leq \ell$, $N_{\sigma_3}^{\mathrm{SAT}}(w, j_d, 1^m)$ accepts.
3. for all even d, $1 \leq d \leq \ell$, $N_{\pi_3}^{\mathrm{SAT}}(w, j_d, 1^m)$ accepts.

Clearly, B_ℓ is in $\mathrm{NP}^{\mathrm{NP}}$, since an $\mathrm{NP}^{\mathrm{NP}}$ machine can guess $j_1, \ldots, j_\ell$, verify the first property, then simulate $N_{\sigma_3}^{\mathrm{SAT}}$ and $N_{\pi_3}^{\mathrm{SAT}}$ for the different values of j_d. Also, observe that the B_ℓ's form a nested sequence

$$B_k \subseteq B_{k-1} \subseteq \ldots \subseteq B_2 \subseteq B_1.$$

Finally, note that if $r = \max\{\ell \mid w \in B_\ell\}$, then

$$w \in B_1 - (B_2 - (B_3 - (\cdots - B_k))) \iff r \text{ is odd.}$$

Example: Here we give an example which demonstrates that

$$w \in L_{u_3} \iff r = \max\{\ell \mid w \in B_\ell\} \text{ is odd.}$$

Let $k = 8$, the maximum order for length m be 5, and $N_{\sigma_3}^{\mathrm{SAT}}$ and $N_{\pi_3}^{\mathrm{SAT}}$ behave as shown below for the different values of s plugged in as the guess for the maximum order.

$s =$	0	1	2	3	4	5	6	7
$N_{\sigma_3}^{\mathrm{SAT}}(w, s, 1^m)$	rej	acc	acc	rej	rej	acc	rej	rej
$N_{\pi_3}^{\mathrm{SAT}}(w, s, 1^m)$	acc	rej	acc	rej	acc	rej	rej	rej

Note that since the maximum order is 5, both $N_{\sigma_3}^{\mathrm{SAT}}$ and $N_{\pi_3}^{\mathrm{SAT}}$ reject for $s = 6, 7$. Also, one of $N_{\sigma_3}^{\mathrm{SAT}}(w, 5, 1^m)$ and $N_{\pi_3}^{\mathrm{SAT}}(w, 5, 1^m)$ must accept and the other reject. For smaller s, both may accept or reject, since their behavior is unpredictable. Finally, $w \in L_{u_3}$, so we want show that r is odd.

To determine if $w \in B_\ell$, look for an alternating (between the top and bottom row) sequence of accepts starting from the top row moving left to right. If there is such a sequence of ℓ accepts, then $w \in B_\ell$. In this example, $r = 3$.

- $w \in B_1$, because j_1 can be 1, 2 or 5.
- $w \in B_2$, because both $N_{\sigma_3}^{\mathrm{SAT}}(w, 1, 1^m)$ and $N_{\pi_3}^{\mathrm{SAT}}(w, 4, 1^m)$ accept.
- $w \in B_3$ with $j_1 = 1, j_2 = 2, j_3 = 5$.

- $w \notin B_4, B_5, \ldots, B_8$, because there is no alternating sequence longer than 3. The sequence $j_1 = 0, j_2 = 1, j_3 = 2, j_4 = 5$ does not count because the sequence must start from the top row.

Claim 1: If $w \in L_{u_3}$, then $r = \max\{\ell \mid w \in B_\ell\}$ is odd.
Proof: Let j be the maximum order for length m. Now suppose r is even and $w \in B_r$. Then, there exist $j_1, \ldots, j_r$ so that properties 1–3 in the definition above hold. Therefore

$$N_{\pi_3}^{\mathrm{SAT}}(w, j_r, 1^m) \text{ accepts}$$

(since r is even and $w \in B_r$). Since $w \in L_{u_3}$, for the true maximum order j,

$$N_{\pi_3}^{\mathrm{SAT}}(w, j, 1^m) \text{ rejects.}$$

Therefore $j_r \neq j$. Observe that j_r cannot be greater than j either since for all $s > j$

$$N_{\pi_3}^{\mathrm{SAT}}(w, s, 1^m) \text{ rejects.}$$

Hence $j_r < j$.

Since we are given that $w \in L_{u_3}$, we know that $N_{\sigma_3}^{\mathrm{SAT}}(w, j, 1^m)$ must accept (Lemma 4). Now consider the sequence $j_1, \ldots, j_{r+1}$ where $j_{r+1} = j$. $N_{\sigma_3}^{\mathrm{SAT}}(w, j_{r+1}, 1^m)$ accepts and $r+1$ is odd which implies that $j_1, \ldots, j_{r+1}$ satisfies conditions 1–3, and therefore $w \in B_{r+1}$. Thus if r is even, $r \neq \max\{\ell \mid w \in B_\ell\}$. Therefore, r must be odd.

Claim 2: If $w \notin L_{u_3}$ then $r = \max\{\ell \mid w \in B_\ell\}$ is even.
Proof: Similar to the proof of Claim 1.

Combining Claims 1 and 2 with the observation that if $r = \max\{\ell \mid w \in B_\ell\}$, then

$$w \in B_1 - (B_2 - (B_3 - (\cdots - B_k))) \iff r \text{ is odd,}$$

we have

$$w \in L_{u_3} \iff w \in B_1 - (B_2 - (B_3 - (\cdots - B_k))).$$

□

Theorem 9 also shows some unexpected connections between the Boolean and query hierarchies within Δ_2^{P} and Δ_3^{P}.

Corollary 10 $\mathrm{BH}(k) = \text{co-}\mathrm{BH}(k) \Longrightarrow \mathrm{BH}_3(k) = \text{co-}\mathrm{BH}_3(k)$.

Corollary 11 $\mathrm{P}^{\mathrm{NP}\|[k]} = \mathrm{P}^{\mathrm{NP}\|[k+1]} \Longrightarrow \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}\|[k+1]} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}\|[k+2]}$.

Proof: We use the fact that the Boolean and query hierarchies are intertwined:

$$\mathrm{P}^{\mathrm{NP}||[k]} \subseteq \mathrm{BH}(k+1) \subseteq \mathrm{P}^{\mathrm{NP}||[k+1]} \quad \text{and} \quad \mathrm{BH}_3(k+1) \subseteq \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}||[k+1]}.$$

If $\mathrm{P}^{\mathrm{NP}||[k]} = \mathrm{P}^{\mathrm{NP}||[k+1]}$, then $\mathrm{BH}(k+1)$ is closed under complementation. Moreover, Corollary 10 implies that $\mathrm{BH}_3(k+1)$ is closed under complementation as well. Thus, the Boolean and query hierarchies in Δ_3^{P} collapse to $\mathrm{BH}_3(k+1)$. These simple containments give a result that is off by one. However, we believe that with more effort we should be able to show that $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}||[k]} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}||[k+1]}$. □

5 Conclusion

We have demonstrated a closer connection between the Boolean hierarchy and the polynomial hierarchy—a deeper collapse of the Boolean hierarchy implies a deeper collapse of the polynomial hierarchy. We would like to think that this relationship is a consequence of some underlying structure connecting $\mathrm{BH}(k)$ and $\mathrm{BH}_3(k)$. However, attempts to simplify the proof along these lines have failed. Is there some straightforward argument which would show that $\mathrm{BH}(k) = \text{co-}\mathrm{BH}(k)$ implies $\mathrm{BH}_3(k)$ is closed under complementation? Could such an argument be extended to show that $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{BH}_3(k)$? Finally, we ask: is this collapse optimal (for $k \geq 2$) or can it be shown that $\mathrm{BH}(k) = \text{co-}\mathrm{BH}(k)$ implies $\mathrm{PH} \subseteq \mathrm{BH}(k)$?

Acknowledgements

We are grateful to Juris Hartmanis for his support and guidance. We would also like to thank Stephen Mahaney for starting us on this endeavor and Georges Lauri for reading a draft of this paper.

References

[1] 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, December 1988.

[2] J. Cai and L. A. Hemachandra. The Boolean Hierarchy: Hardware over NP. In *Structure in Complexity Theory*, pages 105–124, 1986. Springer-Verlag *Lecture Notes in Computer Science # 223*.

[3] J. Kadin. The Polynomial Time Hierarchy Collapses if the Boolean Hierarchy Collapses. *SIAM Journal on Computing*, 17(6):1263–1282, December 1988.

[4] J. Kadin. ERRATUM: The Polynomial Time Hierarchy Collapses if the Boolean Hierarchy Collapses. *SIAM Journal on Computing*, 20(2):404, April 1991.

[5] J. Kadin. *Restricted Turing Reducibilities and the Structure of the Polynomial Time Hierarchy.* PhD thesis, Cornell University, February 1988.

[6] S. Mahaney, 1989. private communication.

[7] K. Wagner and G. Wechsung. On the Boolean Closure of NP. In *Proc. of the 1985 International Conference on Fundamentals of Computation Theory*, pages 485–493. Springer-Verlag *Lecture Notes in Computer Science #199*, 1985.

[8] C. Yap. Some Consequences of Non-uniform Conditions on Uniform Classes. *Theoretical Computer Science*, 26:287–300, 1983.