On Computing Boolean Connectives of Characteristic Functions

Richard Chang*
Jim Kadin

TR 90-1118
May 1990

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

*Supported in part by NSF Research Grant DCR-8520597 and CCR-88-23053, and by an IBM Graduate Fellowship.

On Computing Boolean Connectives of Characteristic Functions

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

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

May 9, 1990

Abstract

We study the existence of polynomial time Boolean connective functions for languages. A language L has an AND function if there is a polynomial time f such that $f(x,y) \in L \iff x \in L$ and $y \in L$. L has an OR function if there is a polynomial time g such that $g(x,y) \in L \iff x \in L$ or $y \in L$. While all NP-complete sets have these functions, we show that Graph Isomorphism, which is probably not complete, also has them. We characterize the complete sets for the classes $\mathrm{D^P}$ and $\mathrm{P^{NP[O(\log n)]}}$ in terms of AND and OR, and we relate these functions to the structure of the Boolean hierarchy and the query hierarchies. We show that the sets that are complete for levels above the second level of the Boolean hierarchy do not have AND and OR unless the polynomial hierarchy collapses. We show that most of the structural properties of the Boolean hierarchy and query hierarchies depend only on the existence of AND and OR functions for NP-complete sets.

1 Introduction

In this paper we consider the existence of polynomial time Boolean combining functions for languages. We say that a language L has a binary AND function, i.e. an $\mathrm{AND_2}$ function, if there is a polynomial time function f such that for all strings x and y, $f(x,y) \in L$ if and only if $x \in L$ and $y \in L$. Similarly, we say that a language L has a binary OR function, an $\mathrm{OR_2}$ function, if there is a polynomial time function g such that for all strings x and y, $g(x,y) \in L$ if and only if $x \in L$ or $y \in L$. In addition, a language may have "any-ary" Boolean functions ($\mathrm{AND_\omega}$ and $\mathrm{OR_\omega}$), polynomial time functions f and g such that for all n and strings $x_1, \cdots, x_n$, $f(x_1, \cdots, x_n) \in L$ if and only if $x_1, \cdots, x_n$ are all in L, and $g(x_1, \cdots, x_n) \in L$ if and only if at least one of $x_1, \cdots, x_n$ is in L.

The existence of these functions is intimately tied to questions about polynomial time reducibilities and structural properties of languages and complexity classes. Our initial motivation for considering these functions was the observation that all NP-complete languages have $\mathrm{AND_\omega}$ and $\mathrm{OR_\omega}$ functions. In fact any language that is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for even relativized versions of complexity classes such as P, NP, PSPACE have any-ary Boolean functions by virtue of the fact that such robust complexity classes are represented by machine models that can be run on the different strings in the input tuple. We will show that languages that are $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{D^P}$ [PY84] have $\mathrm{AND_\omega}$ but do not have $\mathrm{OR_2}$ unless the polynomial time hierarchy collapses. Complete languages for the higher levels of the Boolean hierarchy [CGH+88] do not have either $\mathrm{AND_2}$ or $\mathrm{OR_2}$ unless the polynomial time hierarchy collapses.

*Supported in part by NSF research grants DCR-8520597 and CCR-88-23053, and by an IBM Graduate Fellowship.

These Boolean functions are related to polynomial time conjunctive and disjunctive reducibilities (defined by Ladner, Lynch, and Selman [LLS75]) and to closure of complexity classes under union and intersection. Let m-1(L) be the class of languages $\leq^{\mathrm{P}}_{\mathrm{m}}$- reducible to a language L. It is easy to show that:

$$\begin{aligned} L \text{ has } \mathrm{AND}_2 &\iff \text{m-1}(L) \text{ is closed under intersection} \\ &\iff \text{m-1}(L) \text{ is closed under } \leq^{\mathrm{P}}_{2-\mathrm{c}}, \end{aligned}$$

$$L \text{ has } \mathrm{AND}_\omega \iff \text{m-1}(L) \text{ is closed under } \leq^{\mathrm{P}}_{\mathrm{c}}$$

($\leq^{\mathrm{P}}_{2-\mathrm{c}}$ and $\leq^{\mathrm{P}}_{\mathrm{c}}$ are defined in [LLS75]). Similarly, OR is related to disjunctive reducibilities and union. Hence by looking at these concepts in terms of Boolean functions for languages, we are simply thinking of them more as structural properties of languages than as structural properties of complexity classes. An advantage of this approach is that it becomes convenient to study interesting languages such as Graph Isomorphism and USAT (the set of Boolean formulas that have exactly one satisfying assignment) that are not known to be $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for any standard classes.

The paper is organized as follows. Section 2 presents definitions and preliminary concepts. In Section 3 we discuss some languages that have AND and OR functions. Most notably, we show that Graph Isomorphism does have any-ary AND and OR functions even though it is not known to be NP-complete. In Section 4 we characterize the $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete languages of D^{P} and $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ in terms of AND and OR functions. In Section 5 we use the above characterizations to show that the complete languages of the different levels of the Boolean hierarchy and the related query hierarchies do not have AND and OR functions unless the Boolean hierarchy collapses (which implies that the polynomial time hierarchy collapses [Kad88]).

Finally, in Section 6, we observe that the existence of AND and OR functions for languages is a condition that makes many proof techniques work. For instance the mind-change technique used by Beigel to show that $\mathrm{P}^{\mathrm{SAT}[[2^k-1]]} = \mathrm{P}^{\mathrm{SAT}[k]}$ works for any set A that has binary AND and OR functions. Similarly, most of the theorems concerning the basic structure and normal forms of the Boolean hierarchy and the intertwining of the Boolean and query hierarchies depend only on the fact that SAT has AND_2 and OR_2. The results of this section have been proven independently by Bertoni, Bruschi, Joseph, Sitharam, and Young [BBJ+89].

2 Definitions, Notations and Facts

We assume the reader is familiar with the classes P, NP, the polynomial time hierarchy (PH), the NP-complete language SAT, the graph isomorphism problem, and polynomial time many-one reducibility ($\leq^{\mathrm{P}}_{\mathrm{m}}$).

Although our intuitive notion of AND and OR consists of polynomial time functions that operate on strings, it is convenient to define AND_2, AND_ω, OR_2, OR_ω as sets:

Definition For any set A, we define the sets

$$\begin{aligned} \mathrm{AND}_2(A) &= \{\langle x, y\rangle \mid x \in A \text{ and } y \in A\} \\ \mathrm{OR}_2(A) &= \{\langle x, y\rangle \mid x \in A \text{ or } y \in A\} \\ \mathrm{AND}_k(A) &= \{\langle x_1, \ldots, x_k\rangle \mid \forall i,\ 1 \leq i \leq k, x_i \in A\} \\ \mathrm{OR}_k(A) &= \{\langle x_1, \ldots, x_k\rangle \mid \exists i,\ 1 \leq i \leq k, x_i \in A\} \\ \mathrm{AND}_\omega(A) &= \bigcup_{i=1}^{\infty} \mathrm{AND}_i(A) \end{aligned}$$

$$\mathrm{OR}_\omega(A) = \bigcup_{i=1}^{\infty} \mathrm{OR}_i(A).$$

If $\mathrm{AND}_2(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$ or $\mathrm{AND}_\omega(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$, then we say that A *has* AND_2 or AND_ω respectively. If $\mathrm{OR}_2(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$ or $\mathrm{OR}_\omega(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$, then we say that A *has* OR_2 or OR_ω respectively. Some obvious facts about $\mathrm{AND}_2(A)$ and $\mathrm{OR}_2(A)$ are:

1. $\mathrm{AND}_2(A) \leq_{\mathrm{m}}^{\mathrm{P}} A \iff \mathrm{OR}_2(\overline{A}) \leq_{\mathrm{m}}^{\mathrm{P}} \overline{A}$.
2. if $A \equiv_{\mathrm{m}}^{\mathrm{P}} B$ then $\mathrm{AND}_2(A) \leq_{\mathrm{m}}^{\mathrm{P}} A \iff \mathrm{AND}_2(B) \leq_{\mathrm{m}}^{\mathrm{P}} B$.
3. if $A \equiv_{\mathrm{m}}^{\mathrm{P}} B$ then $\mathrm{OR}_2(A) \leq_{\mathrm{m}}^{\mathrm{P}} A \iff \mathrm{OR}_2(B) \leq_{\mathrm{m}}^{\mathrm{P}} B$.

These facts hold for $\mathrm{AND}_\omega(A)$ and $\mathrm{OR}_\omega(A)$ as well.

Note that if $\mathrm{AND}_2(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$, then for all k, $\mathrm{AND}_k(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$. It is possible that $\mathrm{AND}_2(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$, but $\mathrm{AND}_\omega(A) \not\leq_{\mathrm{m}}^{\mathrm{P}} A$. However if $\mathrm{AND}_2(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$ by a *linear time* reduction, then $\mathrm{AND}_\omega(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$.

Lemma 1 If $\mathrm{AND}_2(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$ by a linear time reduction, then $\mathrm{AND}_\omega(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$. Similarly, if $\mathrm{OR}_2(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$ by a linear time reduction, then $\mathrm{OR}_\omega(A) \leq_{\mathrm{m}}^{\mathrm{P}} A$.

Proof: Let f be a linear time reduction from $\mathrm{AND}_2(A)$ to A. For any r, we can take a tuple $\langle x_1, \ldots, x_r \rangle$ and apply f pairwise to $f(x_1, x_2)\ f(x_3, x_4) \cdots f(x_{r-1}, x_r)$. Then we can apply f pairwise to the outputs of the first applications of f. Repeating this process until we have a single string gives us a tree of applications of f. The height of the tree is $\log r$. If n is the total length of the tuple, $r \leq n$, and so the total running time is bounded by $c^{\log n} n$ for some constant c. This is polynomial in n. □

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

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

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

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

Alternatively, $\mathrm{BH}(2)$ is called $\mathrm{D}^{\mathrm{P}} = \{L_1 - L_2 \mid L_1, L_2 \in \mathrm{NP}\}$. We will use the two terms interchangeably. Also, we will refer to $\mathrm{L}_{\mathrm{BH}(2)}$ as $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ the traditional $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete language for D^{P}.

Definition For any set A, we write $\mathrm{P}^{A[k]}$ for the set of languages recognized by polynomial time Turing machines that ask at most k queries to the oracle A on any input. The *query hierarchy* relative to A is $\mathrm{QH}_A \stackrel{\text{def}}{=} \bigcup_{k=1}^{\infty} \mathrm{P}^{A[k]}$. Similarly, $\mathrm{P}^{A[O(\log n)]}$ is the set of languages recognized by polynomial time Turing machines that ask $O(\log n)$ queries to A. Also, we write $\mathrm{P}^{A[\|[k]}$ for the set of languages recognized by polynomial time Turing machines which ask at most k queries in *parallel* on any input. Finally, we define the *parallel query hierarchy* relative to A as $\mathrm{QH}_{\|A} \stackrel{\text{def}}{=} \bigcup_{k=1}^{\infty} \mathrm{P}^{A[\|[k]}$.

Fact The Boolean, parallel query and the query hierarchies (relative to SAT) are intertwined [Bei87,Hem87]

1. $\mathrm{P}^{\mathrm{SAT}[\|[k]} \subseteq \mathrm{BH}(k+1) \cap \text{co-BH}(k+1)$.
2. $\mathrm{BH}(k) \cup \text{co-BH}(k) \subseteq \mathrm{P}^{\mathrm{SAT}[\|[k]}$.
3. $\mathrm{P}^{\mathrm{SAT}[\|[k]} = \mathrm{P}^{\mathrm{SAT}[2^k-1]}$.
4. $\mathrm{P}^{\mathrm{NP}[O(\log n)]} = \mathrm{P}^{\mathrm{SAT}[\|}$.

3 Languages Which Do

In this section, we present some familiar languages which are known to have AND_ω and OR_ω.

Lemma 2

1. SAT has AND_ω and OR_ω.
2. $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ has AND_ω.

Proof:

1. Given n formulas $\langle f_1, \ldots, f_n \rangle$,

$$\langle f_1, \ldots, f_n \rangle \in \mathrm{AND}_\omega(\mathrm{SAT}) \iff f_1 \wedge \ldots \wedge f_n \in \mathrm{SAT}$$
$$\langle f_1, \ldots, f_n \rangle \in \mathrm{OR}_\omega(\mathrm{SAT}) \iff f_1 \vee \ldots \vee f_n \in \mathrm{SAT}.$$

2. Given n tuples $\langle (f_1, g_1), \ldots, (f_n, g_n) \rangle$,

$$\langle (f_1, g_1), \ldots, (f_n, g_n) \rangle \in \mathrm{AND}_\omega(\mathrm{SAT} \wedge \overline{\mathrm{SAT}})$$
$$\iff (f_1 \wedge \ldots \wedge f_n, g_1 \vee \ldots \vee g_n) \in \mathrm{SAT} \wedge \overline{\mathrm{SAT}}.$$

□

Lemma 2 also implies that all NP-complete languages have AND_ω and OR_ω. In fact any language that is $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for any relativized version of NP, P, or PSPACE has AND_ω and OR_ω. In addition, all languages in P also have AND_ω and OR_ω. One question is whether any of the incomplete languages in $\mathrm{NP} - \mathrm{P}$ have these Boolean functions or not. In our next theorem, we show that Graph Isomorphism, a natural language that is probably not $\leq_{\mathrm{m}}^{\mathrm{P}}$-complete for NP [Sch88], does have AND_ω and OR_ω. One open question is whether Primes has these Boolean functions.

Definition $\mathrm{GI} \stackrel{\text{def}}{=} \{\langle G, H \rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$.

The Labelled Graph Isomorphism problem is the problem of recognizing whether two graphs with labelled vertices are isomorphic by an isomorphism that preserves the labels.

Definition

$$\text{LGI} \stackrel{\text{def}}{=} \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs with labelled nodes, and the isomorphism preserves labels }\}.$$

We will show that LGI has AND_ω and OR_ω functions. The existence of AND_ω and OR_ω functions for GI follows from the fact that $\text{LGI} \equiv_m^P \text{GI}$ [Hof79].

Lemma 3 LGI has AND_ω and OR_ω.

Proof: Without loss of generality we can assume that graphs are represented as adjacency matrices paired with a table mapping vertices to integer labels.

We define an AND_ω function for LGI as follows. Given r pairs of graphs $\langle (G_1, H_1), \ldots, (G_r, H_r)\rangle$, we preprocess each G_i and H_i by:

1. for each G_i and H_i, add a new vertex and make it adjacent to every old vertex of the original graph.

2. define r new labels. For each i, label the new vertex of G_i and H_i with new label i.

Then let G be the disjoint union of all the altered G_i's (i.e. put them all together in one graph with no extra edges), and let H be the disjoint union of all the altered H_i's.

If for all i, the original G_i is isomorphic to the original H_i by a label preserving mapping, then G is isomorphic to H by mapping each G_i to H_i and the new vertex of G_i to the new vertex of H_i. Clearly, this is an isomorphism from G to H that preserves the labelling. If G is isomorphic (label preserving) to H, then the isomorphism must map the unique vertex in G with new label i (the new vertex added to G_i) to the unique vertex in H with new label i (the new vertex added to H_i). This induces an isomorphism between G_i and H_i (for all i).

To show that LGI has OR_ω, we first show that LGI has OR_2. Then, we note that the reduction can be done in linear time, which implies that LGI has OR_ω.

Given 2 pairs of labelled graphs, (G_1, H_1) and (G_2, H_2), we preprocess the graphs as described above (adding 2 new labels). Define a new labelled graph G containing all four graphs G_1, H_1, G_2, and H_2 as subgraphs with 2 new edges added connecting the new vertices of G_1 and G_2 and the new vertices of H_1 and H_2. H is produced similarly except the new edges connect G_1 with H_2 and H_1 with G_2 (see Figure 1).

Suppose G_1 and H_1 are isomorphic by a label preserving mapping. Then, G and H are isomorphic by mapping G_1 in G to H_1 in H, H_1 to G_1, G_2 to G_2 and H_2 to H_2. Symmetrically, if G_2 and H_2 are isomorphic, then G and H are also isomorphic.

If G and H are isomorphic and G_1 is not isomorphic to H_1, then the new vertex of G_1 in G must be mapped to the new vertex of G_1 in H. This means that the new vertex of G_2 in G must be mapped to the new vertex of H_2 in H. This induces an isomorphism between G_2 and H_2.

To see that the reduction from OR_2(LGI) to LGI is linear time, note that we only doubled the size of the input and added only 2 new labels. Lemma 1 then implies that LGI has OR_ω. □

Corollary 4 GI has AND_ω and OR_ω.

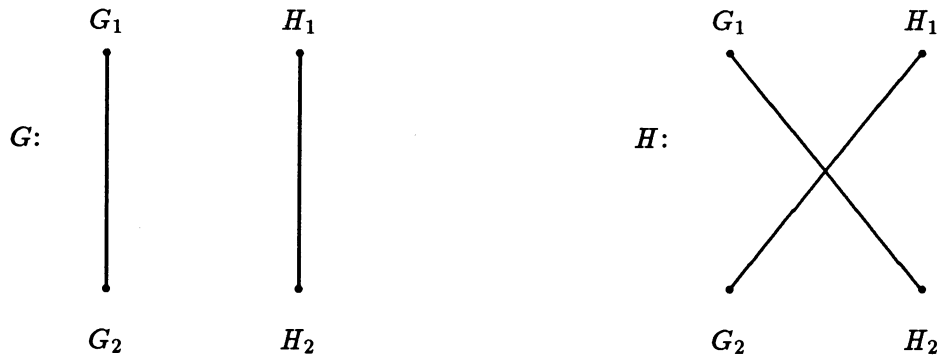

Figure 1: New labelled graphs G and H.

4 Characterizations of Complete Languages

In this section, we show that the complete languages for D^{P} and $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ can be characterized using AND_2 and OR_ω. We show that the two characterizations are very similar. The only difference is that $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete sets have OR_ω and D^{P} complete sets do not. Since $\mathrm{D}^{\mathrm{P}} \neq \mathrm{P}^{\mathrm{NP}[O(\log n)]}$ unless PH collapses, D^{P} complete sets probably do not have OR_ω.

Lemma 5 If $\mathrm{SAT}, \overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} A$ and A has AND_2, then A is D^{P} hard under $\leq^{\mathrm{P}}_{\mathrm{m}}$ reductions.

Proof: $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for D^{P}. Clearly, $\mathrm{SAT} \wedge \overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} A$. So, A must be D^{P} hard. □

Theorem 6 A set A is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for D^{P} if and only if

1. $A \in \mathrm{D}^{\mathrm{P}}$.
2. $\mathrm{SAT}, \overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} A$.
3. A has AND_2.

Now, we show that if a D^{P}-hard set has OR_2 or OR_ω, then it is also hard for higher levels of the parallel query hierarchy.

Theorem 7 Let A be any set that is $\leq^{\mathrm{P}}_{\mathrm{m}}$-hard for D^{P}.

1. If A has OR_2, then for all k, A is $\leq^{\mathrm{P}}_{\mathrm{m}}$-hard for $\mathrm{P}^{\mathrm{SAT}[k]}$.
2. If A has OR_ω, then A is $\leq^{\mathrm{P}}_{\mathrm{m}}$-hard for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

Proof: Let C be any set in $\mathrm{P}^{\mathrm{SAT}[k]}$. To determine if $x \in C$, consider the query tree of the $\mathrm{P}^{\mathrm{SAT}[k]}$ computation. (The query tree is a full binary tree where the internal nodes are labelled by the oracle queries. The two subtrees below the node represent the computations that follow oracle reply. One branch assumes the oracle replied "yes", the other "no".) The query tree has height k and 2^k leaves. Only one path in the tree (from root to leaf) is the correct path, and $x \in C$ if and only if this path ends in an accepting configuration.

Now we show that a D^P computation can determine if a given path is the correct path. Let $p_1, \ldots, p_i$ be the queries on the path assumed to be answered "yes", and $q_1, \ldots, q_j$ be the queries assumed to be answered "no". Then, the path is correct if and only if $p_1 \wedge \ldots \wedge p_i \in \text{SAT}$ and $q_1 \vee \ldots \vee q_j \in \overline{\text{SAT}}$. This is exactly a D^P computation.

Since the query tree is of constant depth, it is possible to generate the entire tree in polynomial time and write down all the paths that end in an accepting configuration. Note that $x \in C$ if and only if one of these accepting paths is the correct path. Let r be the number of accepting paths. We can use r separate D^P computations to check if any of the accepting paths is the correct path; and since we are assuming that A is D^P-hard and has OR_2, we can use one A computation instead of the r D^P computations.

The proof of the second case is similar. The only difference is that the query tree is polynomial in size instead of constant. □

Corollary 8 If SAT, $\overline{\text{SAT}} \leq_m^P A$ and A has AND_2 and OR_ω, then A is $P^{NP[O(\log n)]}$ hard under $\leq_m^P$ reductions.

Proof: By Lemma 5 A is a D^P-hard set. By the preceding theorem, A is also hard for $P^{NP[O(\log n)]}$. □

Theorem 9 A set A is $\leq_m^P$-complete for $P^{NP[O(\log n)]}$ if and only if

1. $A \in P^{NP[O(\log n)]}$.
2. $\text{SAT}, \overline{\text{SAT}} \leq_m^P A$.
3. A has AND_2.
4. A has OR_ω.

Notice that the only difference in the characterizations of D^P complete sets and $P^{NP[O(\log n)]}$ complete sets (Theorems 6 and 9) is that $P^{NP[O(\log n)]}$ complete sets have OR_ω. So, we have the following corollary.

Corollary 10 If $\text{SAT} \wedge \overline{\text{SAT}}$ has OR_ω, then $P^{NP[O(\log n)]} \subseteq D^P$.

Since $P^{NP[O(\log n)]} \subseteq D^P$ implies that PH collapses to just above Σ_2^P [CK89], Corollary 10 is evidence that $\text{SAT} \wedge \overline{\text{SAT}}$ does not have OR_ω. In fact, it probably does not have OR_2, either.

Corollary 11 If $\text{SAT} \wedge \overline{\text{SAT}}$ has OR_2, then $P^{NP[O(\log n)]} \subseteq D^P$.

Proof: If $\text{SAT} \wedge \overline{\text{SAT}}$ has OR_2, then by Theorem 7, $\text{SAT} \wedge \overline{\text{SAT}}$ is hard for $P^{SAT[2]}$. However, the complement of $\text{SAT} \wedge \overline{\text{SAT}}$ is $\overline{\text{SAT}} \vee \text{SAT}$ which is in $P^{SAT[2]}$. So, $\text{SAT} \wedge \overline{\text{SAT}} \equiv_m^P \overline{\text{SAT}} \vee \text{SAT}$. Since $\text{SAT} \wedge \overline{\text{SAT}}$ has AND_ω (Lemma 2), $\overline{\text{SAT}} \vee \text{SAT}$ has OR_ω by DeMorgan's Law. Thus, $\text{SAT} \wedge \overline{\text{SAT}}$ must also have OR_ω since the two sets are $\leq_m^P$-equivalent. Then, by Corollary 10 $P^{NP[O(\log n)]} \subseteq D^P$. □

Corollary 12 If $D^P = \text{co-}D^P$ then PH collapses and $P^{NP[O(\log n)]} \subseteq D^P$.

Proof: If D^P= co-D^P, then $\overline{SAT} \vee SAT$ is $\leq_m^P$-complete for D^P. So, $\overline{SAT} \vee SAT$ has AND_2 by Theorem 6. Then, by DeMorgan's Law, $SAT \wedge \overline{SAT}$ has OR_2 which implies $P^{NP[O(\log n)]} \subseteq D^P$ by Theorem 11. The polynomial hierarchy collapses by Kadin's proof [Kad88]. □

It was observed by Kadin [Kad87] that the collapse of the Boolean hierarchy at level 2 (D^P = co-D^P) immediately implies $P^{NP[O(\log n)]} \subseteq D^P$. However, we cannot push the same theorem through for the collapse of the Boolean hierarchy at levels 3 or higher. We explain this phenomenon using AND_2 and OR_2. Observe that in the proof above we relied on the fact that $SAT \wedge \overline{SAT}$ has AND_2. We will show in the next section that the complete languages for the levels of the Boolean hierarchy cannot have AND_2 or OR_2, unless PH collapses.

5 Languages Which Don't

In this section, we show that the complete languages for the levels of the Boolean hierarchy and query hierarchies probably do not have AND_2 or OR_2. In the following theorems, keep in mind that $BH(k) \subseteq BH \subseteq P^{NP[O(\log n)]}$. Also recall that if $BH \subseteq BH(k)$, $QH_{||} \subseteq P^{SAT||[k]}$, $QH \subseteq P^{SAT[k]}$, or $P^{NP[O(\log n)]} \subseteq BH(k)$, then the polynomial hierarchy collapses. Of course Theorems 13 and 14 apply to any $\leq_m^P$-complete sets for the various levels of the BH.

Theorem 13 For $k \geq 2$,

1. $L_{BH(k)}$ has $OR_2 \iff BH \subseteq BH(k)$.
2. $L_{BH(k)}$ has $OR_\omega \iff P^{NP[O(\log n)]} \subseteq BH(k)$.

Proof:

1. ($\Rightarrow$) For $k \geq 2$, $L_{BH(k)}$ is D^P-hard, so by Theorem 7, $L_{BH(k)}$ has OR_2 implies that it is hard for $P^{SAT[k]}$. However, $L_{\text{co-}BH(k)} \in P^{SAT[k]}$, so $L_{\text{co-}BH(k)} \leq_m^P L_{BH(k)}$. This implies that $BH(k)$ is closed under complementation, which implies $BH \subseteq BH(k)$.

   ($\Leftarrow$) If $BH \subseteq BH(k)$ then $P^{SAT||[2k]} \subseteq BH(k)$, since $P^{SAT||[2k]} \subseteq BH(2k+1)$. Clearly, $OR_2(L_{BH(k)}) \in P^{SAT||[2k]}$, so $L_{BH(k)}$ has OR_2.

2. ($\Rightarrow$) As above, $L_{BH(k)}$ is D^P-hard. By Theorem 9, $L_{BH(k)}$ has OR_ω implies that $L_{BH(k)}$ is $\leq_m^P$-complete for $P^{NP[O(\log n)]}$. So, $P^{NP[O(\log n)]} \subseteq BH(k)$.

   ($\Leftarrow$) $OR_\omega(L_{BH(k)}) \in P^{SAT||} = P^{NP[O(\log n)]}$. So, if $P^{NP[O(\log n)]} \subseteq BH(k)$ then $L_{BH(k)}$ has OR_ω. □

Theorem 14 For $k \geq 3$,

1. $L_{BH(k)}$ has $AND_2 \iff BH \subseteq BH(k)$.
2. $L_{BH(k)}$ has $AND_\omega \iff P^{NP[O(\log n)]} \subseteq BH(k)$.

Proof: DeMorgan's Law implies that $L_{\text{co-}BH(k)}$ has OR_2 or OR_ω. Then, the proof proceeds as above, because for $k \geq 3$, $L_{\text{co-}BH(k)}$ is D^P-hard. □

If we believe that the polynomial hierarchy does not collapse, then the preceding theorems tell us that as we go up the Boolean hierarchy, the complete languages lose the Boolean connective

functions. At the first level, $L_{BH(1)} = SAT$, so $L_{BH(1)}$ has both AND_ω and OR_ω. At the second level, $L_{BH(2)}$ is D^Pcomplete, so $L_{BH(2)}$ has AND_ω but does not have OR_2. From the third level up, $L_{BH(k)}$ has neither AND_2 nor OR_2. In conclusion, we see that when we walk up the Boolean hierarchy from level one, the complete languages lose pieces of "robustness".

The $\leq^P_m$-complete languages for the different levels of query hierarchy and the parallel query hierarchy (relative to SAT) also do not have AND_2 or OR_2 unless the PH collapses.

Theorem 15 For any k, if C is $\leq^P_m$-complete for $P^{SAT|||[k]}$, then

$$C \text{ has } AND_2 \text{ or } OR_2 \Longrightarrow BH \subseteq P^{SAT|||[k]}.$$

Proof: First, note that since $P^{SAT|||[k]}$ is closed under complementation, $\overline{C} \equiv^P_m C$. Therefore C has AND_2 if and only if C has OR_2. We will show that if C has either Boolean function, then $BH(k+1) \subseteq P^{SAT|||[k]}$ which implies the collapse of the BH (and the PH). If k is odd, then the $\leq^P_m$-complete language for $BH(k+1)$ is

$$L_{BH(k+1)} = \{\langle x_1, \ldots, x_{k+1}\rangle \mid \langle x_1, \ldots, x_k\rangle \in L_{BH(k)} \text{ and } x_{k+1} \in \overline{SAT}\}.$$

If C has AND_2, then $L_{BH(k+1)} \leq^P_m C$ since $L_{BH(k)} \leq^P_m C$ and $\overline{SAT} \leq^P_m C$. But this implies $BH(k+1) \subseteq P^{SAT|||[k]}$. If k is even, then the $\leq^P_m$-complete language for $BH(k+1)$ is

$$L_{BH(k+1)} = \{\langle x_1, \ldots, x_{k+1}\rangle \mid \langle x_1, \ldots, x_k\rangle \in L_{BH(k)} \text{ or } x_{k+1} \in SAT\}.$$

If C has OR_2, then $L_{BH(k+1)} \leq^P_m C$ since $L_{BH(k)} \leq^P_m C$ and $SAT \leq^P_m C$. Again this implies $BH(k+1) \subseteq P^{SAT|||[k]}$. □

In related research, we have shown that USAT, the set of Boolean formulas that have exactly one satisfying assignment, does not have OR_ω unless the PH collapses [CK90].

6 AND_2 and OR_2 and Hierarchies

Considerable research has focused on understanding the structure of the Boolean hierarchy and the query hierarchies [WW85,Bei87,CGH+88,KSW87]. The basic structural properties of these hierarchies are as follows:

1. *Normal forms for* BH: there are many different but equivalent ways to define the levels of the BH [CGH+88].
2. *Complete languages*:
   (a) $L_{BH(k)}$ is $\leq^P_m$-complete for $BH(k)$.
   (b) $ODD_k(SAT)$ is $\leq^P_m$-complete for $BH(k)$ [Bei87,WW85]. ($ODD_k(SAT)$ is defined below)
   (c) $EVEN_k(SAT) \oplus ODD_k(SAT)$ is $\leq^P_m$-complete for $P^{SAT|||[k]}$ [Bei87,WW85].
3. *Basic containments and intertwining* [Bei87,KSW87]:

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

4. *Intertwining of the query hierarchies* [Bei87]:

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

5. *Upward collapse*:

$$\begin{aligned} BH(k) = \text{co-}BH(k) &\implies BH \subseteq BH(k) \\ P^{SAT||[k]} = P^{SAT||[k+1]} &\implies QH_{||} \subseteq P^{SAT||[k]} \\ P^{SAT[k]} = P^{SAT[k+1]} &\implies QH \subseteq P^{SAT[k]}. \end{aligned}$$

Definition For any set A, we define the following languages:

$$\begin{aligned} ODD_k(A) &\stackrel{\text{def}}{=} \{\langle x_1, \ldots, x_k\rangle \mid \|\{x_1, \ldots, x_k\} \cap A\| \text{ is odd}\} \\ EVEN_k(A) &\stackrel{\text{def}}{=} \{\langle x_1, \ldots, x_k\rangle \mid \|\{x_1, \ldots, x_k\} \cap A\| \text{ is even}\}. \end{aligned}$$

In this section, we show that all the above properties follow simply from the fact that SAT (or any NP-complete set) has AND_2 and OR_2. That is, they do not depend on the fact that SAT is NP-complete or even in NP at all. This observation follows from the fact that given any set A that has AND_2 and OR_2, the Boolean and query hierarchies based on A have all the above properties. The results of this section have been proven independently by Bertoni, Bruschi, Joseph, Sitharam, and Young [BBJ+89].

The QH and $QH_{||}$ relative to a set A were defined in section 2. We define the Boolean hierarchy relative to A as the difference hierarchy of m-1$(A) \stackrel{\text{def}}{=}$ set of all languages $\leq^P_m$-reducible to A.

Definition

$$\begin{aligned} BH_A(1) &= \text{m-1}(A), \\ BH_A(2k) &= \{L \mid L = L' \cap \overline{L_2} \text{ where } L' \in BH_A(2k-1) \text{ and } L_2 \in \text{m-1}(A)\}, \\ BH_A(2k+1) &= \{L \mid L = L' \cup L_2 \text{ where } L' \in BH_A(2k) \text{ and } L_2 \in \text{m-1}(A)\}, \\ \text{co-}BH_A(k) &= \{L \mid \overline{L} \in BH_A(k)\}, \\ BH_A &= \bigcup_{k=1}^{\infty} BH_A(k). \end{aligned}$$

First, we will leave it to the reader to show that if A has AND_2 and OR_2, then the various normal forms defined in [CGH+88] hold for the different levels of BH_A (see [CH85]). For example, a language L is in $BH_A(k)$ iff there exist languages $L_1, \ldots, L_k \in$ m-1(A) such that

$$L = D(L_1, \ldots, L_k) \stackrel{\text{def}}{=} L_1 - (L_2 - (L_3 - (\cdots - L_k))).$$

Theorem 16 If A has AND_2 and OR_2, then $ODD_k(A)$ is $\leq^P_m$-complete for $BH_A(k)$.

Proof: First, we show that $ODD_k(A)$ is $\leq^P_m$-hard for $BH_A(k)$. If $L \in BH_A(k)$, then $L = D(L_1, \ldots, L_k)$ for $L_1, \ldots, L_k \in$ m-1(A). Define $L'_i \stackrel{\text{def}}{=} \bigcap_{j \leq i} L_j$. Then each $L'_i \in$ m-1(A), since m-1(A) is closed under intersection, and $L'_1 \supseteq L'_2 \supseteq \cdots \supseteq L'_k$. One can prove by induction that $D(L_1, \ldots, L_k) = D(L'_1, \ldots, L'_k)$. Then clearly, $x \in L$ iff the number of sets $L'_1, \ldots, L'_k$ that contains x is odd. Since each L'_i is in m-1(A), given a string x, we can reduce x to $\langle q_1, \ldots, q_k\rangle$ such that $x \in L'_i \iff q_i \in A$. Then $x \in L \iff \langle q_1, \ldots, q_k\rangle \in ODD_k(A)$. Therefore $ODD_k(A)$ is $\leq^P_m$-hard for $BH_A(k)$.

To see that $ODD_k(A) \in BH_A(k)$, for each $i \leq k$, consider the set

$L_i \stackrel{\text{def}}{=} \{\langle x_1, \ldots, x_k\rangle \mid \|\{x_1, \ldots, x_k\} \cap A\| \geq i\}$.

$L_i \in$ m-1(A) since given $\langle x_1, \ldots, x_k\rangle$, we can generate all $s = \binom{k}{i}$ subsets of i strings and use the reduction from $\text{OR}_s(\text{AND}_i(\text{A}))$ to A to generate a string that is in A iff all i strings in one of the subsets are in A. Then $\text{ODD}_k(A) = D(L_1, \ldots, L_k)$, implying $\text{ODD}_k(A) \in \text{BH}_A(k)$. □

The most difficult proof in this section is the argument that $\text{EVEN}_k(A) \oplus \text{ODD}_k(A)$ is $\leq^{\text{P}}_{\text{m}}$-complete for $\text{P}^{A|||[k]}$. We prove this by showing that the mind-change technique used by Beigel [Bei87] works provided A has AND_2 and OR_2.

Theorem 17 If A has AND_2 and OR_2, then $\text{EVEN}_k(A) \oplus \text{ODD}_k(A)$ is $\leq^{\text{P}}_{\text{m}}$-complete for $\text{P}^{A|||[k]}$.

Proof: Both $\text{EVEN}_k(A)$ and $\text{ODD}_k(A)$ are obviously in $\text{P}^{A|||[k]}$, so we need to show that every set in $\text{P}^{A|||[k]}$ can be reduced to $\text{EVEN}_k(A) \oplus \text{ODD}_k(A)$.

Let L be a language in $\text{P}^{A|||[k]}$ computed by a polynomial time oracle Turing machine M which asks at most k parallel queries. With each $M^A(x)$ computation, we associate *truth tables* that have the following syntax:

q_1	q_2	...	q_k	result
0	0	...	0	acc
0	0	...	1	rej
⋮	⋮		⋮	⋮
1	1	...	1	rej

The idea is that $q_1, \ldots, q_k$ are the queries that $M^A(x)$ makes. Each row of the truth table records whether the computation accepts or rejects assuming that the answer to the queries are as listed in the row. (A "1" in column q_i means we are assuming that $q_i \in A$ and a "0", $q_i \notin A$). The full truth table has 2^k lines, but consider truth tables with fewer lines. In particular, we call a truth table *valid* if

1. The first row of the truth table is all 0's.
2. If a "1" appears in column q_i of row j, then for all rows below row j a "1" appears in column q_i. (Think of each row as the set of q_i's assumed to be in A represented as a bit vector. This condition implies that the rows are monotonic under the set containment relation.)
3. If a "1" appears in column q_i of any row, then q_i is in fact an element of A.

N.B. valid truth tables have between 1 and $k+1$ rows. The following is an example of a valid truth table.

q_1	q_2	q_3	q_4	q_5	q_6	q_7	result
0	0	0	0	0	0	0	rej
0	0	1	0	0	0	0	acc
0	0	1	0	1	0	0	acc
0	0	1	0	1	1	0	rej
0	0	1	0	1	1	0	acc
1	0	1	0	1	1	0	rej

For each valid truth table T, we associate a number m_T—the mind changes of T—which is the number of times the result column changes from accept to reject or from reject to acc. In the example above, m_T is 4. Since valid truth tables have between 1 and $k+1$ rows, $0 \leq m_T \leq k$.

Now we define the set of valid truth tables labelled with the number of mind changes.

$$\mathcal{T} = \{\langle T, x, s\rangle \mid T \text{ is a valid truth table for } M^A(x) \text{ and } m_T = s\}.$$

Claim: $\mathcal{T} \leq_{\mathrm{m}}^{\mathrm{P}} A$. On input $\langle T, x, s\rangle$, a polynomial time machine can simulate $M^A(x)$ to determine which strings, $q_1, \ldots, q_k$, will be queried. Then, the machine can easily check the syntax of T to see if it meets conditions 1 and 2 in the definition of a valid truth table and to see if the number of mind changes is indeed equal to s. If any of these conditions is not met, $\langle T, x, s\rangle$ is reduced to a known string in $\overline{A}$. Otherwise, T is a valid truth table if and only if all the q_i's with a 1 in the last row are actually in A. This last condition can be reduced to A via the reduction from $\mathrm{AND}_k(A)$ to A.

Now define

$$\mathcal{ET}_r = \{x \mid \exists T, s \text{ such that } s \geq r \text{ and } \langle T, x, s\rangle \in \mathcal{T}\}.$$

Claim: $\mathcal{ET}_r \leq_{\mathrm{m}}^{\mathrm{P}} A$. Since k is constant, for fixed queries $q_1, \ldots, q_k$, there is only a constant number of truth tables with k columns. So simply generate all truth tables T and all values of s between 0 and r. Since $x \in \mathcal{ET}_r$ if and only if $\langle T, x, s\rangle \in \mathcal{T}$ for one of the (T, s) pairs generated and since $\mathcal{T} \leq_{\mathrm{m}}^{\mathrm{P}} A$, $\mathcal{ET}_r$ can be reduced to A by using $\mathrm{OR}_2(A)$ to combine all the reductions from $\mathcal{T}$ to A into one reduction.

We use the reduction from $\mathcal{ET}_r$ to A to produce a reduction from L to $\mathrm{EVEN}_k(A) \oplus \mathrm{ODD}_k(A)$. This reduction will simply print out a k-tuple $\langle z_1, \ldots, z_k\rangle$ and an extra bit to indicate if the reduction is to $\mathrm{EVEN}_k(A)$ or $\mathrm{ODD}_k(A)$. Each z_i has the property that $z_i \in A$ iff $x \in \mathcal{ET}_i$ iff there exists a valid truth table that makes i mind changes. Let t be the maximum number of mind changes. Then, $z_1, \ldots, z_t \in A$ and $z_{t+1}, \ldots, z_k \notin A$. So, the parity of the number of z_i's in A is the same as the parity of t.

Now, we claim that the parity of the maximum number of mind changes is enough to determine if $M^A(x)$ accepted. First, note that there must exist a valid truth table with the maximum number of mind changes which has, in its last row, the actual oracle replies. (I.e., there is a 1 in column q_i of the last row if and only if $q_i \in A$.) To see this, simply confirm that adding the row of actual oracle replies to the bottom of a valid truth table results in another valid truth table. Thus, $M^A(x)$ accepts iff the result in the last row of this truth table is "accept".

Second, consider this valid truth table that makes t mind changes. Note that if t is odd (even) then the result in the last row is the opposite of (same as) the result in the first row. Suppose the result in the first row is "accept", then $x \in L$ iff t is even iff $\langle z_1, \ldots, z_k\rangle \in \mathrm{EVEN}_k(A)$. Similarly, if the result in the first row is "reject", then $x \in L$ iff $\langle z_1, \ldots, z_k\rangle \in \mathrm{ODD}_k(A)$. Since the result in the first row can be computed in polynomial time, a polynomial time function can reduce L to $\mathrm{EVEN}_k(A) \oplus \mathrm{ODD}_k(A)$. □

Theorem 18 If A has AND_2 and OR_2, then

$$\mathrm{BH}_A(k) \cup \text{co-}\mathrm{BH}_A(k) \subseteq \mathrm{P}^{A\|[k]} \subseteq \mathrm{BH}_A(k+1) \cap \text{co-}\mathrm{BH}_A(k+1).$$

Proof: The first containment follows straight from the definitions of the various classes.

The second containment holds because $\mathrm{EVEN}_k(A) \oplus \mathrm{ODD}_k(A) \leq_{\mathrm{m}}^{\mathrm{P}} \mathrm{ODD}_{k+1}(A)$ and the completeness results of the previous two theorems. □

Theorem 19 If a set A has AND_2 and OR_2, then $\mathrm{P}^{A[k]} = \mathrm{P}^{A\|[2^k-1]}$.

Proof: For any A, $\mathrm{P}^{A[k]} \subseteq \mathrm{P}^{A|||[2^k-1]}$, so all we need to show is $\mathrm{P}^{A|||[2^k-1]} \subseteq \mathrm{P}^{A[k]}$. We will do this by showing that $\mathrm{ODD}_{2^k-1}(A) \in \mathrm{P}^{A[k]}$ and then Theorem 17 implies that $\mathrm{P}^{A|||[2^k-1]} \subseteq \mathrm{P}^{A[k]}$.

The main idea of this proof is to use k serial queries to do binary search over $2^k - 1$ strings to determine how many are elements of A. To determine if at least r strings are in A, generate all $s = \binom{2^k-1}{r}$ subsets of the queries with r elements. Then, use the reduction from $\mathrm{OR}_s(\mathrm{AND}_r(\mathrm{A}))$ to A to determine if one of the subsets contains only strings in A. If so, at least r of the query strings are elements of A. Since it takes exactly k steps of binary search to search over $2^k - 1$ elements, $\mathrm{P}^{A[k]} = \mathrm{P}^{A|||[2^k-1]}$. □

In [Cha89], Chang erroneously stated that for all A, if $\mathrm{P}^{A|||[k]} = \mathrm{P}^{A|||[k+1]}$, then the entire query hierarchy based on A collapses to $\mathrm{P}^{A|||[k]}$. This is only known to be true when A has AND_2 and OR_2.

Theorem 20 If A has AND_2 and OR_2 and $\mathrm{P}^{A|||[k]} = \mathrm{P}^{A|||[k+1]}$, then for all $j > k$, $\mathrm{P}^{A|||[k]} = \mathrm{P}^{A|||[j]}$.

Proof: It suffices to show that under these assumptions, $\mathrm{P}^{A|||[k+1]} = \mathrm{P}^{A|||[k+2]}$. Since A has AND_2 and OR_2, by Theorem 17, $\mathrm{EVEN}_{k+1}(A) \oplus \mathrm{ODD}_{k+1}(A)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{P}^{A|||[k+1]}$. However, $\mathrm{P}^{A|||[k]} = \mathrm{P}^{A|||[k+1]}$, so there is some $\mathrm{P}^{A|||[k]}$ computation for $\mathrm{EVEN}_{k+1}(A) \oplus \mathrm{ODD}_{k+1}(A)$. This means that a $\mathrm{P}^{A|||[k+1]}$ computation can accept $\mathrm{EVEN}_{k+2}(A) \oplus \mathrm{ODD}_{k+2}(A)$ by using k queries to compute the parity of the first $k+1$ strings and the last query to determine if the last string is in A. Since $\mathrm{EVEN}_{k+2}(A) \oplus \mathrm{ODD}_{k+2}(A)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\mathrm{P}^{A|||[k+2]}$, $\mathrm{P}^{A|||[k+1]} = \mathrm{P}^{A|||[k+2]}$. □

The upward collapse properties of the BH_A and QH_A follow from Theorem 20.

References

[BBJ+89] A. Bertoni, D. Bruschi, D. Joseph, M. Sitharam, and P. Young. Generalized Boolean hierarchies and the hierarchy over RP. Technical Report 809, Department of Computer Sciences, University of Wisconsin—Madison, 1989.

[Bei87] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. Technical Report 7, Department of Computer Science, The Johns Hopkins University, 1987. To appear in *Theoretical Computer Science.*

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

[CH85] J. Cai and L. A. Hemachandra. The Boolean hierarchy: Hardware over NP. Technical Report TR 85-724, Cornell Department of Computer Science, December 1985.

[Cha89] R. Chang. On the structure of bounded queries to arbitrary NP sets. In *Proceedings of the 4th Structure in Complexity Theory Conference*, pages 250–258, June 1989.

[CK89] R. Chang and J. Kadin. The Boolean hierarchy and the polynomial hierarchy: a closer connection. Technical Report 89-1008, Department of Computer Science, Cornell University, May 1989. To appear in *Proceedings of the 5th Structure in Complexity Theory Conference.*

[CK90] R. Chang and J. Kadin. On the structure of uniquely satisfiable formulas. Technical Report 90-1124, Department of Computer Science, Cornell University, May 1990.

[Hem87] L. A. Hemachandra. The strong exponential hierarchy collapses. In *ACM Symposium on Theory of Computing*, pages 110–122, 1987.

[Hof79] Christoph M. Hoffman. *Group-Theoretic Algorithms and Graph Isomorphism.* Lecture Notes in Computer Science #136. Springer-Verlag, 1979.

[Kad87] J. Kadin. $P^{NP[\log n]}$ and sparse Turing complete sets for NP. In *Proceedings of the 2nd Structure in Complexity Theory Conference*, pages 33–40, Ithaca, New York, June 1987. Revised version to appear in *Journal of Computer and System Sciences.*

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

[KSW87] J. Köbler, Uwe Schöning, and K. Wagner. The difference and truth-table hierarchies for NP. *RAIRO Theoretical Informatics and Applications*, 21:419–435, 1987.

[LLS75] R. E. Ladner, N. A. Lynch, and A. L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–123, 1975.

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

[Sch88] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37:312–323, December 1988.

[WW85] K. Wagner and G. Wechsung. On the Boolean closure of NP. In *Proceedings of the 1985 International Conference on Fundamentals of Computation Theory*, Springer-Verlag *Lecture Notes in Computer Science #199*, pages 485–493, 1985.