Bounded Query Functions with Limited Output Bits

Richard Chang[†‡] chang@umbc.edu

Jon S. Squire[†] squire@umbc.edu

University of Maryland Baltimore County

Abstract

This paper explores the difference between parallel and serial queries to an NP*-complete oracle,* SAT*, from the perspective of functions with a limited number of output bits. For polynomial-time bounded query language classes, which can be considered as functions with 1-bit output, previous work has shown that 2 serial queries to* SAT *is equivalent to 3 parallel queries to* SAT*. In contrast, for function classes with no limit on the number of output bits, previous work has shown that there exists a function that can be computed in polynomial time using 3 parallel queries to* SAT*, but cannot be computed using 2 serial queries to* SAT*, unless* P = NP*. The results in this paper show that there exists a function with 2-bit output that can be computed using 3 parallel queries to* SAT*, but cannot be computed using 2 serial queries to* SAT*, unless the Polynomial Hierarchy collapses.*

1. Introduction

An important topic in the study of bounded query classes is the difference between parallel and serial oracle access mechanisms. When an oracle Turing machine uses the parallel oracle access mechanism, all of the queries to the oracle are asked simultaneously. Then, the oracle's replies are given as a bit-vector. In contrast, when a Turing machine makes serial queries, the queries can depend on the replies to the previous queries. The difference between parallel and serial queries also highlights the difference between bounded query language and function classes. For example, for *language* classes, for all constants k, there is a tight equivalence between parallel and serial queries to an NP-complete language (e.g., SAT) [5]:

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

This equality is proven using the mind change technique. On the other hand, for bounded query *function* classes, while it is trivial to show that

$$\mathrm{PF}^{\mathrm{SAT}[k]} \subseteq \mathrm{PF}^{\mathrm{SAT}\|[2^k-1]},$$

we also know that [7]

$$\mathrm{PF}^{\mathrm{SAT}[k]} = \mathrm{PF}^{\mathrm{SAT}\|[2^k-1]} \Longrightarrow \mathrm{P} = \mathrm{NP}. \tag{2}$$

The proof in this case is based upon the following enumerability argument. Consider the function $\chi^{\mathrm{SAT}}_{2^k-1}$ which, on input $F_1, \ldots, F_{2^k-1}$, outputs a string with 2^k-1 bits where the ith bit is 1 if and only if $F_i \in \mathrm{SAT}$. Clearly, $\chi^{\mathrm{SAT}}_{2^k-1}$ can be computed by a $\mathrm{PF}^{\mathrm{SAT}\|[2^k-1]}$ machine. However, a $\mathrm{PF}^{\mathrm{SAT}[k]}$ machine has only 2^k possible outputs which cannot include all of the 2^{2^k-1} possible values of $\chi^{\mathrm{SAT}}_{2^k-1}$. The enumerability argument exploits the fact that some possible values of $\chi^{\mathrm{SAT}}_{2^k-1}$ are missing from the list of possible outputs of a $\mathrm{PF}^{\mathrm{SAT}[k]}$ machine to collapse NP down to P [7].

The results stated in (1) and (2) leave us in a bit of confusion because we cannot say outright that $2^k - 1$ parallel queries are stronger than k serial queries. This is only true for functions and not languages. And even in the case of function classes it is not clear if the reason that $\mathrm{PF}^{\mathrm{SAT}[k]} = \mathrm{PF}^{\mathrm{SAT}\|[2^k-1]} \Longrightarrow \mathrm{P} = \mathrm{NP}$ is due to enumerability or the hardness of SAT. In particular, the enumerability argument described above can be used to show that for any oracle X (even nonrecursive ones), $\chi^{\mathrm{SAT}}_{2^k-1} \in \mathrm{PF}^{X[k]} \Longrightarrow \mathrm{P} = \mathrm{NP}$.

In this paper, we revisit the issue of parallel versus serial queries by considering functions that have a limited number of output bits. Let $\mathrm{PF}_r^{\mathrm{SAT}[k]}$ denote the

[†]Address: Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA.

[‡]Supported in part by the National Science Foundation under research grant CCR-9610457 and by the University of Maryland Institute for Advanced Computer Studies.

class of functions computable by $\mathrm{PF}^{\mathrm{SAT}[k]}$ machines that output at most r bits and let $\mathrm{PF}_r^{\mathrm{SAT}\|[k]}$ be the analogous class for parallel queries. Since $\mathrm{PF}_1^{\mathrm{SAT}[k]}$ and $\mathrm{PF}_1^{\mathrm{SAT}\|[k]}$ machines are essentially language recognizers, by (1):

$$\mathrm{PF}_1^{\mathrm{SAT}[k]} = \mathrm{PF}_1^{\mathrm{SAT}\|[2^k-1]}.$$

Also, since $\chi_{2^k-1}^{\mathrm{SAT}}$ has 2^k-1 bits of output, by (2):

$$\mathrm{PF}_{2^k-1}^{\mathrm{SAT}[k]} = \mathrm{PF}_{2^k-1}^{\mathrm{SAT}\|[2^k-1]} \Longrightarrow \mathrm{P} = \mathrm{NP}.$$

So, what about $\mathrm{PF}_j^{\mathrm{SAT}[k]}$ versus $\mathrm{PF}_j^{\mathrm{SAT}\|[2^k-1]}$ for values of j greater than 1 and less than 2^k-1? Can the two classes be equal? Would equality imply that P = NP?

The main result in this paper shows that,

$$\mathrm{PF}_2^{\mathrm{SAT}[2]} = \mathrm{PF}_2^{\mathrm{SAT}\|[3]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}.$$

In contrast, for machines limited to 1 bit of output we already know that $\mathrm{PF}_1^{\mathrm{SAT}[2]} = \mathrm{PF}_1^{\mathrm{SAT}\|[3]}$ [5] and for machines limited to 3 bits of output $\mathrm{PF}_3^{\mathrm{SAT}[2]} = \mathrm{PF}_3^{\mathrm{SAT}\|[3]} \Longrightarrow \mathrm{P} = \mathrm{NP}$ [7].

The main result is interesting for two reasons. First of all, a $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machine can enumerate all 4 possible outputs of a two bit function (namely, 00, 01, 10 and 11). Thus, we cannot directly use any of the proof techniques based upon enumerability [1, 2, 5, 7, 11]. In particular, there does exist an oracle X such that $\mathrm{PF}_2^{\mathrm{SAT}\|[3]} \subseteq \mathrm{PF}_2^{X[2]}$. Recall that for functions with unlimited output bits, $\mathrm{PF}^{\mathrm{SAT}\|[3]} \subseteq \mathrm{PF}^{X[2]}$ for any oracle X would imply that P = NP. Secondly, the function $\mathcal{Q}_{32}$, which we use to spite $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machines, is not known to be complete for $\mathrm{PF}_2^{\mathrm{SAT}\|[3]}$. In fact, we would conjecture that $\mathcal{Q}_{32}$ is not complete for $\mathrm{PF}_2^{\mathrm{SAT}\|[3]}$. Furthermore, we spite $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machines not by giving it hard instances of $\mathcal{Q}_{32}$. Instead, we use instances of $\mathcal{Q}_{32}$ for which a $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machine can produce the correct answer. By forcing the machine to behave properly on these easy instances, we gain enough leverage on the machine's behavior on harder instances and thus are able to show that PH collapses if $\mathcal{Q}_{32} \in \mathrm{PF}_2^{\mathrm{SAT}[2]}$.

Our proof techniques can be generalized to show that for all j, k and ℓ such that $1 < j \leq k < \ell \leq 2^k-1$, if $2^k-(\ell+1) < 2^j-2$ and $2^k-(\ell+1) < \ell-1$, then

$$\mathrm{PF}_j^{\mathrm{SAT}\|[\ell]} \subseteq \mathrm{PF}_j^{\mathrm{SAT}[k]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}.$$

In the general case, we consider $\mathrm{PF}_j^{\mathrm{SAT}\|[\ell]} \subseteq \mathrm{PF}_j^{\mathrm{SAT}[k]}$ rather than $\mathrm{PF}_j^{\mathrm{SAT}\|[\ell]} = \mathrm{PF}_j^{\mathrm{SAT}[k]}$ because for $\ell < 2^k-1$, it is already known that $\mathrm{PF}_j^{\mathrm{SAT}[k]} \subseteq \mathrm{PF}_j^{\mathrm{SAT}\|[\ell]}$ implies a collapse of the Boolean Hierarchy which in turn collapses PH.

2. Preliminaries

In this section we discuss the definitions and notation used in this paper as well as prior results needed to prove the main theorem. We assume the reader is familiar with the standard definitions in computational complexity theory (q.v. [3, 4, 13]). We begin with the notation for various bounded query classes.

Definition 1

- Let $\mathrm{P}^{A[k]}$ denote the class of *languages* recognized by polynomial-time oracle Turing machines which ask at most k *serial* queries to the oracle A.
- Let $\mathrm{P}^{A\|[k]}$ denote the class of *languages* recognized by polynomial-time oracle Turing machines which ask at most k *parallel* queries to the oracle A.
- Let $\mathrm{PF}^{A[k]}$ denote the class of *functions* computed by polynomial-time oracle Turing machines which ask at most k *serial* queries to the oracle A.
- Let $\mathrm{PF}^{A\|[k]}$ denote the class of *functions* computed by polynomial-time oracle Turing machines which ask at most k *parallel* queries to the oracle A.
- Let $\mathrm{PF}_j^{A[k]}$ be the subset of $\mathrm{PF}^{A[k]}$ consisting of those functions that output at most j bits.
- Let $\mathrm{PF}_j^{A\|[k]}$ be the subset of $\mathrm{PF}^{A\|[k]}$ consisting of those functions that output at most j bits.

Figure 1 shows the possible computation paths of a $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machine M on input x. This diagram is called the *oracle query tree* for the computation of $M(x)$. We establish the convention that the computation path in an oracle query tree proceeds to the *left* when the oracle replies no. This allows us to define the *sequence of possible outputs* for a $\mathrm{PF}^{A[k]}$ computation $M(x)$ to be the left-to-right ordering of the outputs at each leaf of the oracle query tree. We will also use $\textsc{Out}(M,x)$ to denote this sequence of possible outputs. We define the *true path* in an oracle query tree to be the path taken using the correct replies from the oracle. Given a specific path in the oracle query tree, we say that a query string q_i is a *positive query relative to the path* if the path assumes that $q_i \in \mathrm{SAT}$ — i.e., the path proceeds to the right after the query node q_i. When the number of serial queries is bounded by $O(\log n)$ we can compute the sequence of possible outputs in polynomial time using no oracle queries simply by trying all possible replies from the oracle.

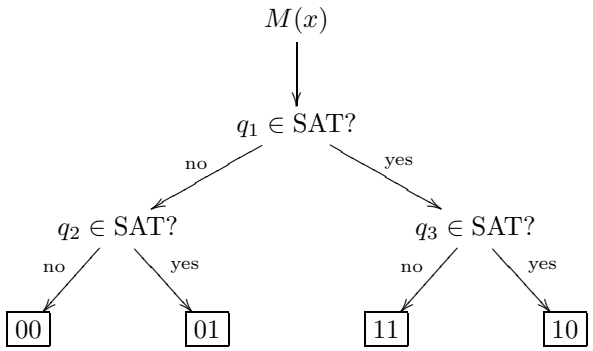

Figure 1: The oracle query tree of a $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ computation. The sequence of possible outputs, $\textsc{Out}(M, x)$, is defined as $\langle 00, 01, 11, 10\rangle$.

Definition 2 For constant k, we define the languages BL_k, coBL_k and $\mathrm{ODD}_k^{\mathrm{SAT}}$ as follows:

$$\mathrm{BL}_1 = \mathrm{SAT}$$

$$\mathrm{BL}_{2k} = \{\ \langle x_1, \ldots, x_{2k}\rangle \mid \langle x_1, \ldots, x_{2k-1}\rangle \in \mathrm{BL}_{2k-1} \text{ and } x_{2k} \in \overline{\mathrm{SAT}}\ \}$$

$$\mathrm{BL}_{2k+1} = \{\ \langle x_1, \ldots, x_{2k+1}\rangle \mid \langle x_1, \ldots, x_{2k}\rangle \in \mathrm{BL}_{2k} \text{ or } x_{2k+1} \in \mathrm{SAT}\ \}$$

$$\mathrm{coBL}_k = \{\ \langle x_1, \ldots, x_k\rangle \mid \langle x_1, \ldots, x_k\rangle \notin \mathrm{BL}_k\ \}$$

$$\mathrm{ODD}_k^{\mathrm{SAT}} = \{\ \langle x_1, \ldots, x_k\rangle \mid \|\{i \mid (1 \le i \le k) \wedge (x_i \in \mathrm{SAT})\}\| \text{ is odd}\ \}$$

Definition 3 We say that a sequence of Boolean formulas $\langle F_1, \ldots, F_k\rangle$ is *nested* if for all i, $2 \le i \le k$,

$$F_i \in \mathrm{SAT} \Longrightarrow F_{i-1} \in \mathrm{SAT}.$$

The language BL_k is $\le_{\mathrm{m}}^{\mathrm{P}}$-complete for the kth level of the Boolean Hierarchy [8, 9]. In our proof of the main theorem, we will work with BL_3 and coBL_3 which can also be defined directly as:

$$\mathrm{BL}_3 = (\mathrm{SAT} \wedge \overline{\mathrm{SAT}}) \vee \mathrm{SAT} = \{\langle F_1, F_2, F_3\rangle \mid (F_1 \in \mathrm{SAT} \wedge F_2 \in \overline{\mathrm{SAT}}) \vee F_3 \in \mathrm{SAT}\}$$

$$\mathrm{coBL}_3 = (\overline{\mathrm{SAT}} \vee \mathrm{SAT}) \wedge \overline{\mathrm{SAT}} = \{\langle F_1, F_2, F_3\rangle \mid (F_1 \in \overline{\mathrm{SAT}} \vee F_2 \in \mathrm{SAT}) \wedge F_3 \in \overline{\mathrm{SAT}}\}.$$

Note that $\langle F_1, \ldots, F_k\rangle \in \mathrm{BL}_k$ if and only if the largest i such that $F_i \in \mathrm{SAT}$ is odd. Thus, if a sequence $\langle F_1, \ldots, F_k\rangle$ is nested,

$$\langle F_1, \ldots, F_k\rangle \in \mathrm{BL}_k \iff \langle F_1, \ldots, F_k\rangle \in \mathrm{ODD}_k^{\mathrm{SAT}}.$$

Given any sequence of formulas $\langle F_1, \ldots, F_k\rangle$, we can construct a nested sequence $\langle F_1', \ldots, F_k'\rangle$ in polynomial time such that

$$\langle F_1, \ldots, F_k\rangle \in \mathrm{BL}_k \iff \langle F_1', \ldots, F_k'\rangle \in \mathrm{BL}_k.$$

Simply let $F_j' = F_j \vee \cdots \vee F_k$. For the rest of the paper, we use the convention that $\langle F_1', \ldots, F_k'\rangle$ denotes the nested version of $\langle F_1, \ldots, F_k\rangle$. Thus, for all $\langle F_1, \ldots, F_k\rangle$ (even non-nested sequences),

$$\langle F_1, \ldots, F_k\rangle \in \mathrm{BL}_k \iff \langle F_1', \ldots, F_k'\rangle \in \mathrm{ODD}_k^{\mathrm{SAT}}.$$

Definition 4 The function $\mathcal{Q}_{32}$ takes as input three Boolean formulas $\langle F_1, F_2, F_3\rangle$ and produces two bits of output ab. The first bit a is 1 if and only if $\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3$. The second bit b is 1 if and only if $\langle F_1, F_2, F_3\rangle \in \mathrm{ODD}_3^{\mathrm{SAT}}$.

We will use the function $\mathcal{Q}_{32}$ to spite $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machines. Note that $\mathcal{Q}_{32}$ is easily computable by a $\mathrm{PF}_2^{\mathrm{SAT}\|[3]}$ machine. We simply ask the SAT oracle in parallel if each of F_1, F_2 and F_3 is a member of SAT and determine whether $\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3$ and $\langle F_1, F_2, F_3\rangle \in \mathrm{ODD}_3^{\mathrm{SAT}}$ using the reply from the oracle. On the other hand, there is no obvious way for a $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machine to compute $\mathcal{Q}_{32}$. A $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machine can compute each bit of $\mathcal{Q}_{32}$, but there is no obvious way to compute both bits using 2 serial queries to SAT. One difference between $\mathcal{Q}_{32}(F_1, F_2, F_3)$ and $\chi_3^{\mathrm{SAT}}(F_1, F_2, F_3)$ is that both bits of $\mathcal{Q}_{32}$ depend on all three formulas whereas the ith bit of χ_3^{SAT} depends only on F_i. We will show in the next section that the assumption that $\mathcal{Q}_{32} \in \mathrm{PF}_2^{\mathrm{SAT}[2]}$ is enough to collapse the Polynomial Hierarchy (PH).

Suppose that $\langle F_1, F_2, F_3\rangle$ is a nested sequence. Then the output of $\mathcal{Q}_{32}(F_1, F_2, F_3)$ can only be 00 or 11. This is due to the observation we made above that for nested sequences:

$$\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3 \iff \langle F_1, F_2, F_3\rangle \in \mathrm{ODD}_3^{\mathrm{SAT}}.$$

Thus, a $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machine can compute the value of $\mathcal{Q}_{32}(F_1, F_2, F_3)$ when $\langle F_1, F_2, F_3\rangle$ is nested, since the

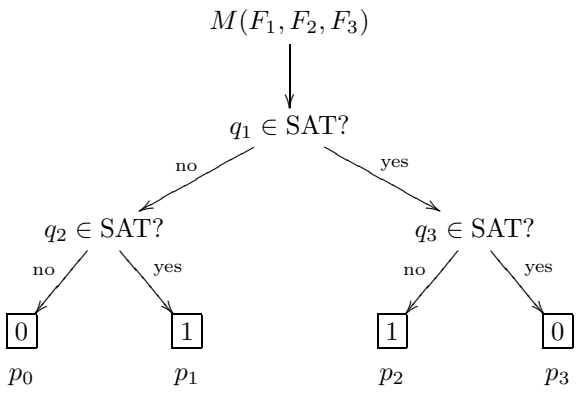

Figure 2: A $\mathrm{PF}_1^{\mathrm{SAT}[2]}$ computation with possible output sequence $\textsc{Out}(M, \langle F_1, F_2, F_3\rangle) = \langle 0, 1, 1, 0\rangle$.

first and second bits are equal. One would think that a proof that $\mathcal{Q}_{32} \in \mathrm{PF}_2^{\mathrm{SAT}[2]} \Longrightarrow$ PH collapses should avoid nested sequence entirely. However, the proof of our main theorem does the exact opposite, as we shall see in the next section.

To prove our main theorem, we will also need the following lemmas:

Lemma 5 If $\mathrm{BL}_3 \leq_{\mathrm{m}}^{\mathrm{P/poly}} \mathrm{coBL}_3$, then $\mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}$.

Proof Sketch: We can use any of several proofs that a collapse of the Boolean Hierarchy implies the collapse of PH [6, 10, 12, 15]. Using any of these proofs, we would get

$$\mathrm{BL}_3 \leq_{\mathrm{m}}^{\mathrm{P}} \mathrm{coBL}_3 \Longrightarrow \overline{\mathrm{SAT}} \in \mathrm{NP/poly}.$$

In our application, the reduction from BL_3 to coBL_3 is by a polynomial-time function which has polynomial advice. Since the advice for the reduction can also be given to the NP machine computing $\overline{\mathrm{SAT}}$. We still get

$$\mathrm{BL}_3 \leq_{\mathrm{m}}^{\mathrm{P/poly}} \mathrm{coBL}_3 \Longrightarrow \overline{\mathrm{SAT}} \in \mathrm{NP/poly}.$$

Then, PH collapses to Σ_3^{P} by Yap's theorem [17]. □

Lemma 6 Let M be a $\mathrm{PF}_1^{\mathrm{SAT}[2]}$ machine. Suppose that for some $\langle F_1, F_2, F_3\rangle$ we know that

$$M(F_1, F_2, F_3) = 1 \iff \langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3.$$

Furthermore, suppose that the sequence of possible outputs, $\textsc{Out}(M, \langle F_1, F_2, F_3\rangle)$, is not $\langle 0, 1, 0, 1\rangle$. Then, we can construct $\langle G_1, G_2, G_3\rangle$ in time polynomial in the running time of $M(F_1, F_2, F_3)$ such that

$$\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3 \iff \langle G_1, G_2, G_3\rangle \in \mathrm{coBL}_3.$$

Proof Sketch: The proof of this lemma is implicit in the original work of Wagner and Wechsung on the Boolean Hierarchy [16]. From their results, one can show that any $\mathrm{PF}_1^{\mathrm{SAT}[2]}$ computation can be reduced to one of BL_1, coBL_1, BL_2, coBL_2, BL_3 and coBL_3 depending on the number and type of mind changes made in the sequence of possible outputs of the oracle query tree. The only sequence that corresponds to BL_3 is $\langle 0, 1, 0, 1\rangle$. In the remaining cases when the possible output sequence is not $\langle 0, 1, 0, 1\rangle$, the computation of $M(F_1, F_2, F_3)$ can be reduced to coBL_3, since BL_1, coBL_1, BL_2, coBL_2 all reduce to coBL_3. We provide the proof of one of these cases and leave the rest to the reader.

Consider the $\mathrm{PF}_1^{\mathrm{SAT}[2]}$ computation in Figure 2. Using the queries q_1, q_2 and q_3 in the oracle query tree, we define the following sequence of Boolean formulas:

$$p_0 = \mathrm{TRUE}, \qquad p_1 = q_2$$
$$p_2 = q_1, \qquad p_3 = q_1 \wedge q_3.$$

The general pattern here is that p_i is the conjunction of the positive queries relative to path i in the oracle query tree, where the paths are numbered from left to right. In general, path i is the true path if and only if p_i is satisfiable and for all $j > i$, p_j is unsatisfiable. Next we turn $\langle p_0, p_1, p_2, p_3\rangle$ into a nested sequence $\langle p'_0, p'_1, p'_2, p'_3\rangle$:

$$p'_0 = \mathrm{TRUE}, \qquad p'_1 = q_2 \vee q_1 \vee (q_1 \wedge q_3)$$
$$p'_2 = q_1 \vee (q_1 \wedge q_3), \qquad p'_3 = q_1 \wedge q_3.$$

By hypothesis, $\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3$ if and only if $M(F_1, F_2, F_3) = 1$. Since $M(F_1, F_2, F_3)$ outputs 1 only on path 1 and path 2, $\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3$ if and only if the index of the true path is in the half-open interval $[1, 3)$. This happens exactly when $p'_1 \in \mathrm{SAT}$ and $p'_3 \in \overline{\mathrm{SAT}}$. Thus, the computation of $M(F_1, F_2, F_3)$ reduces

to BL_2. More explicitly, we can define $\langle G_1, G_2, G_3\rangle$ by $G_1 = \mathrm{TRUE}$, $G_2 = p'_1$, and $G_3 = p'_3$. Then, $\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3$ if and only if $\langle G_1, G_2, G_3\rangle \in \mathrm{coBL}_3$. □

3. Main Theorem

Theorem 7 (Main Theorem)
$\mathrm{PF}_2^{\mathrm{SAT}[2]} = \mathrm{PF}_2^{\mathrm{SAT}\|[3]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}$.

Proof: Since $\mathrm{PF}_2^{\mathrm{SAT}[2]} \subseteq \mathrm{PF}_2^{\mathrm{SAT}\|[3]}$ trivially and since $\mathcal{Q}_{32} \in \mathrm{PF}_2^{\mathrm{SAT}\|[3]}$, it suffices to show that $\mathcal{Q}_{32} \in \mathrm{PF}_2^{\mathrm{SAT}[2]}$ implies $\mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}$. Let M be a $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machine that computes $\mathcal{Q}_{32}$. Our goal is to construct a P/poly reduction h from BL_3 to coBL_3. Then by Lemma 5, PH collapses to Σ_3^{P} as desired. Let $\langle F_1, F_2, F_3\rangle$ be an instance of the input to h. Our first step is construct the nested version of $\langle F_1, F_2, F_3\rangle$, namely:

$$F'_1 = F_1 \vee F_2 \vee F_3, \quad F'_2 = F_2 \vee F_3, \quad \text{and} \quad F'_3 = F_3.$$

Recall that

$$\begin{aligned}&\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3\\ &\quad\Longleftrightarrow\ \langle F'_1, F'_2, F'_3\rangle \in \mathrm{BL}_3\\ &\quad\Longleftrightarrow\ \langle F'_1, F'_2, F'_3\rangle \in \mathrm{ODD}_3^{\mathrm{SAT}}.\end{aligned}$$

Now consider $\textsc{Out}(M, \langle F'_1, F'_2, F'_3\rangle)$ the sequence of possible outputs. Since M is a $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machine and $\langle F'_1, F'_2, F'_3\rangle$ is nested, M can compute the value of $\mathcal{Q}_{32}(F'_1, F'_2, F'_3)$ which is either 00 or 11. The most obvious way of doing so is to ask the SAT oracle whether F'_2 is satisfiable. If so, check F'_3; otherwise, F'_1. The obvious algorithm for computing $\mathcal{Q}_{32}(F'_1, F'_2, F'_3)$ gives us an oracle query tree with possible output sequence $\langle 00, 11, 00, 11\rangle$. Since we have no control over M's strategy, it is possible that $\textsc{Out}(M, \langle F'_1, F'_2, F'_3\rangle) \neq \langle 00, 11, 00, 11\rangle$. However, in that case, the sequence of either the first or the second bits of $\textsc{Out}(\mathrm{M}, \langle F'_1, F'_2, F'_3\rangle)$ is not $\langle 0, 1, 0, 1\rangle$. Then by Lemma 6, we can immediately construct $\langle G_1, G_2, G_3\rangle$ such that

$$\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3 \iff \langle G_1, G_2, G_3\rangle \in \mathrm{coBL}_3,$$

and we would be done. Thus, we can assume that $\textsc{Out}(M, \langle F'_1, F'_2, F'_3\rangle) = \langle 00, 11, 00, 11\rangle$ — that is, M must behave nicely on the easy instances of $\mathcal{Q}_{32}$. For the harder instances, we need the help of an advice function to construct $\langle G_1, G_2, G_3\rangle$.

Before we define the advice function, let us first see how the advice might help us. Suppose that we are given, among other things, a formula H by the advice function and are also told whether $H \in \mathrm{SAT}$. Now, consider the behavior of M on input $\langle F'_1 \wedge H, F'_2, F'_3\rangle$. There are three possible cases to consider.

Case 1: If $H \in \mathrm{SAT}$, then $\langle F'_1 \wedge H, F'_2, F'_3\rangle$ is still a nested sequence. In fact, we have

$$\begin{aligned}&\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3\\ &\quad\Longleftrightarrow\ \langle F'_1 \wedge H, F'_2, F'_3\rangle \in \mathrm{BL}_3\\ &\quad\Longleftrightarrow\ \langle F'_1 \wedge H, F'_2, F'_3\rangle \in \mathrm{ODD}_3^{\mathrm{SAT}}.\end{aligned}$$

Thus, $\textsc{Out}(M, F'_1 \wedge H, F'_2, F'_3)$ would still have to be $\langle 00, 11, 00, 11\rangle$ or we win by Lemma 6 as before.

Case 2: Now, suppose that $H \notin \mathrm{SAT}$ and that $\textsc{Out}(M, \langle F'_1 \wedge H, F'_2, F'_3\rangle)$ *does* equal $\langle 00, 11, 00, 11\rangle$. Then we can construct $\langle G_1, G_2, G_3\rangle$ as follows. Suppose that $F_3 \in \mathrm{SAT}$, then $\mathcal{Q}_{32}(F'_1 \wedge H, F'_2, F'_3) = 10$ since $F'_1 \wedge H \notin \mathrm{SAT}$, $F'_2 \in \mathrm{SAT}$ and $F'_3 \in \mathrm{SAT}$. Since 10 is not in $\textsc{Out}(M, \langle F'_1 \wedge H, F'_2, F'_3\rangle)$ and M computes $\mathcal{Q}_{32}$, it follows that $F_3 \notin \mathrm{SAT}$. Thus,

$$\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3 \iff F_1 \in \mathrm{SAT} \text{ and } F_2 \notin \mathrm{SAT}.$$

Therefore, we can define $\langle G_1, G_2, G_3\rangle$ as

$$G_1 = \mathrm{TRUE}, \quad G_2 = F_1, \quad \text{and} \quad G_3 = F_2$$

and be guaranteed that $\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3$ if and only if $\langle G_1, G_2, G_3\rangle \in \mathrm{coBL}_3$.

Case 3: It remains possible that for $H \in \mathrm{SAT}$, $\textsc{Out}(M, \langle F'_1 \wedge H, F'_2, F'_3\rangle) = \langle 00, 11, 00, 11\rangle$ and for $H \notin \mathrm{SAT}$, $\textsc{Out}(M, \langle F'_1 \wedge H, F'_2, F'_3\rangle) \neq \langle 00, 11, 00, 11\rangle$. However, if this happens for every H, then we would have a polynomial time algorithm that determines whether $H \in \mathrm{SAT}$ since

$$\begin{aligned}&H \in \mathrm{SAT} \iff\\ &\quad \textsc{Out}(M, \langle F'_1 \wedge H, F'_2, F'_3\rangle) = \langle 00, 11, 00, 11\rangle.\end{aligned}$$

Having a polynomial time algorithm for SAT would certainly help us reduce BL_3 to coBL_3. In fact, we can relax this requirement to the probabilistic case. We can get a BPP algorithm for SAT as long as for most $\langle x_1, x_2, x_3\rangle$, $\textsc{Out}(M, \langle x_1 \wedge H, x_2, x_3\rangle) = \langle 00, 11, 00, 11\rangle$ if and only if $H \in \mathrm{SAT}$.

We are now ready to construct the advice function. For those who are familiar with the proof techniques in bounded query complexity, we will use the "advisees trick" from Amir, Beigel and Gasarch [1, 2]. We will essentially show that one of the three cases described above must always happen.

Fix a length n. We will consider only those triples $\langle x_1, x_2, x_3\rangle$ such that $n = |x_1| = |x_2| = |x_3|$. Let H be a Boolean formula of length n. We say that $\langle x_1, x_2, x_3\rangle$ is an *advisee* of H if either $H \in \mathrm{SAT}$

and $\textsc{Out}(M, \langle x'_1 \wedge H, x'_2, x'_3\rangle) \neq \langle 00, 11, 00, 11\rangle$ or $H \notin$ SAT and $\textsc{Out}(M, \langle x'_1 \wedge H, x'_2, x'_3\rangle) = \langle 00, 11, 00, 11\rangle$. As usual, $\langle x'_1, x'_2, x'_3\rangle$ denotes the nested version of $\langle x_1, x_2, x_3\rangle$. In the construction of the first part of the advice string for length n, we add to the advice string the H's that have the most number of advisees.

Advice Construction, First Part

1. advice := ε
2. $\mathcal{S} := \{\vec{x} \mid \vec{x} = \langle x_1, x_2, x_3\rangle, |x_1| = |x_2| = |x_3| = n\}$
3. $\mathcal{T} := \{\phi \mid \phi$ is a Boolean formula of length $n\}$
4. while $\mathcal{S}$ has at least 16 elements, repeat Steps 4a through 4f
   a. For each $\phi \in \mathcal{T}$, let $\mathcal{A}(\phi)$ be the set of $\vec{x} \in \mathcal{S}$ such that $\vec{x} = \langle x_1, x_2, x_3\rangle$ is an advisee of ϕ.
   b. Let H be a Boolean formula of length n such that $|\mathcal{A}(H)|$ is largest.
   c. If $|\mathcal{A}(H)| < \frac{1}{4}|\mathcal{S}|$, halt construction (abnormally).
   d. $\mathcal{S} := \mathcal{S} - \mathcal{A}(H)$.
   e. $\mathcal{T} := \mathcal{T} - \{H\}$.
   f. If $H \in$ SAT, add $(H, 0)$ to the advice. Otherwise, add $(H, 1)$ to the advice.

End of Construction

The while loop in this construction can iterate at most $O(n)$ times since each time through the loop we remove at least one quarter of the elements from $\mathcal{S}$. The construction of the first part of the advice string can terminate in two ways. When the while loops terminates normally, $\mathcal{S}$ contains fewer than 16 triples. Then, we simply add each triple to the advice string along with 1 bit indicating whether the triple is in BL_3. We call this the second part of the advice string. We also add one more bit to the advice string indicating that the while loop terminated normally.

On the other hand, if the while loop halted abnormally, then $\forall H \in \mathcal{T}$ and $\forall \vec{x} \in \mathcal{S}$, let $E(\vec{x})$ denote the event that $\textsc{Out}(M, \langle x'_1 \wedge H, x'_2, x'_3\rangle) = \langle 00, 11, 00, 11\rangle$, where $\vec{x} = \langle x_1, x_2, x_3\rangle$.

$$H \in \text{SAT} \Longrightarrow \text{Prob}_{\vec{x}\in\mathcal{S}}[\ E(\vec{x})\ ] > 3/4. \tag{3}$$

$$H \notin \text{SAT} \Longrightarrow \text{Prob}_{\vec{x}\in\mathcal{S}}[\ \neg E(\vec{x})\ ] > 3/4. \tag{4}$$

If this were not the case, then for some $H \in \mathcal{T}$, $|\mathcal{A}(H)|$ would be at least $\frac{1}{4}|\mathcal{S}|$ and we would not have halted the loop abnormally. Thus, we almost have a BPP algorithm for SAT for instances in $\mathcal{T}$. (Note that the only H's of length n that are not in $\mathcal{T}$ have already been added to the advice string.) The difficulty in converting the implications in (3) and (4) into a BPP algorithm is that the set $\mathcal{S}$ might be difficult to compute. Hence it is not clear how one can effectively choose an $\vec{x}$ randomly from $\mathcal{S}$. However, this difficulty does not prevent us from amplifying the probabilities by repeated trials and siding with the majority. Then, using Schöning's proof that BPP $\subseteq$ P/poly [14], we obtain a polynomially long sequence of triples $\vec{x}_1, \ldots, \vec{x}_m$ such that $\forall H \in \mathcal{T}$, $H \in$ SAT if and only if for a majority of the $\vec{x}_i = \langle x_{i1}, x_{i2}, x_{i3}\rangle$, $\textsc{Out}(M, \langle x'_{i1} \wedge H, x'_{i2}, x'_{i3}\rangle) = \langle 00, 11, 00, 11\rangle$. (A complete proof for the process of extracting $\vec{x}_1, \ldots, \vec{x}_m$ is given by Amir, Beigel and Gasarch [2, Theorem 10.6].) Thus, when the while loop in the construction of the first part of the advice string halts abnormally, we add the sequence $\vec{x}_1, \ldots, \vec{x}_m$ to the advice string. As before, we call this the second part of the advice string. This ends the construction of the advice string.

Finally, we recap how the P/poly reduction h uses the advice function to reduce BL_3 to coBL_3.

1. Input $\langle F_1, F_2, F_3\rangle$, where $|F_1| = |F_2| = |F_3| = n$.
2. Let $\langle F'_1, F'_2, F'_3\rangle$ be the nested version of $\langle F_1, F_2, F_3\rangle$.
3. If $\textsc{Out}(M, \langle F'_1, F'_2, F'_3\rangle) \neq \langle 00, 11, 00, 11\rangle$, then use Lemma 6 to construct $\langle G_1, G_2, G_3\rangle$.
4. Examine the advice string for length n and determine whether the while loop terminated normally during the construction of the first part of the advice string.
5. If the while loop terminated normally:
   a. Check whether $\langle F_1, F_2, F_3\rangle$ is one of the < 16 triples remaining in $\mathcal{S}$ when the loop terminated. If so, the second part of the advice string states whether $\langle F_1, F_2, F_3\rangle \in \text{BL}_3$. Then output a trivial $\langle G_1, G_2, G_3\rangle$ that is or is not a member of coBL_3 as appropriate.
   b. Look for an $H \in$ SAT in the first part of the advice string such that
      $$\textsc{Out}(M, \langle F'_1 \wedge H, F'_2, F'_3\rangle) \neq \langle 00, 11, 00, 11\rangle.$$
      If such an $H \in$ SAT is found, use Lemma 6 to construct $\langle G_1, G_2, G_3\rangle$.
   c. If Steps 5a and 5b failed, then by construction, there must be an $H \notin$ SAT in the first part of the advice string such that
      $$\textsc{Out}(M, \langle F'_1 \wedge H, F'_2, F'_3\rangle) = \langle 00, 11, 00, 11\rangle.$$
      In this case, the fact that 10 is not in the possible output sequence guarantees that $F_3 \notin$ SAT. We construct $\langle G_1, G_2, G_3\rangle$ as discussed previously.

6. If the while loop terminated abnormally:

   a. For each $F_j \in \{F_1, F_2, F_3\}$, check whether F_j is equal to some H in the first part of the advice string. If so, the advice string also indicated whether $F_j \in \mathrm{SAT}$.

   b. If the membership of F_j in SAT was not determined in the previous step, then $F_j \in \mathcal{T}$ when the while loop terminated. Then, for each $\vec{x}_i \in \{\vec{x}_1, \ldots, \vec{x}_m\}$ from the second part of the advice string, let $\vec{x}_i = \langle x_{i1}, x_{i2}, x_{i3}\rangle$ and compute $\textsc{Out}(M, \langle x'_{i1} \wedge F_j, x'_{i2}, x'_{i3}\rangle)$, the possible output sequence. As argued above, $F_j \in \mathrm{SAT}$ if and only if for a majority of the $\vec{x}_i$'s, the possible output sequence is $\langle 00, 11, 00, 11\rangle$.

   c. Since we have determined the membership of F_1, F_2 and F_3 in SAT, we can determine whether $\langle F_1, F_2, F_3\rangle \in \mathrm{BL}_3$. Hence, we can output a trivial $\langle G_1, G_2, G_3\rangle$ that is or is not a member of coBL_3 as appropriate. □

4. Extensions

The proof of the main theorem can be extended in several ways. We omit the full proofs in this version of the paper. In the first extension, we consider the case of k serial queries for functions with 2 bits of output.

Theorem 8 For all $k \geq 2$,

$$\mathrm{PF}_2^{\mathrm{SAT}[k]} = \mathrm{PF}_2^{\mathrm{SAT}\|[2^k-1]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}.$$

Proof Sketch: The proof proceeds in the same fashion as the proof for Theorem 7. We do need to prove an analog of Lemma 6 for BL_{2^k-1} and $\mathrm{PF}_1^{\mathrm{SAT}[k]}$ machines whose sequence of possible outputs is not $\langle 0, 1, 0, 1, \ldots, 0, 1\rangle$. Also, instead of $\mathcal{Q}_{32}$ we will use a function $\mathcal{Q}$ which on input $\langle F_1, \ldots, F_{2^k-1}\rangle$ outputs the 2-bit value ab, where $a = 1 \iff \langle F_1, \ldots, F_{2^k-1}\rangle \in \mathrm{BL}_{2^k-1}$ and $b = 1 \iff \langle F_1, \ldots, F_{2^k-1}\rangle \in \mathrm{ODD}_{2^k-1}^{\mathrm{SAT}}$. □

Corollary 9 For all $j \geq 2$ and $k \geq 2$,

$$\mathrm{PF}_j^{\mathrm{SAT}[k]} = \mathrm{PF}_j^{\mathrm{SAT}\|[2^k-1]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}.$$

Proof: The corollary follows immediately from Theorem 8 and the observation that for all $j \geq 2$,

$$\mathrm{PF}_j^{\mathrm{SAT}\|[2^k-1]} \subseteq \mathrm{PF}_j^{\mathrm{SAT}[k]} \Longrightarrow \mathrm{PF}_2^{\mathrm{SAT}\|[2^k-1]} \subseteq \mathrm{PF}_2^{\mathrm{SAT}[k]}.$$ □

In the next extension, we compare $\mathrm{PF}_2^{\mathrm{SAT}[k]}$ to $\mathrm{PF}_2^{\mathrm{SAT}\|[\ell]}$ where $\ell < 2^k - 1$. We encounter some complications when we generalize the proof of the main theorem. Consider a $\mathrm{PF}_2^{\mathrm{SAT}[3]}$ machine M that computes $\mathcal{Q}_{62}$, the concatenation of BL_6 and $\mathrm{ODD}_6^{\mathrm{SAT}}$. The possible output sequence of M contains 8 strings. We can generalize Lemma 6 to show that given a nested input $\vec{F} = \langle F_1, \ldots, F_6\rangle$, $\textsc{Out}(M, \vec{F})$ must contain as a subsequence $\langle 00, 11, 00, 11, 00, 11, 00\rangle$ (in this case we say that $M(\vec{F})$ makes 6 *mind changes*), otherwise we can construct $\vec{G}$ such that $\vec{F} \in \mathrm{BL}_6 \iff \vec{G} \in \mathrm{coBL}_6$. However, $M(\vec{F})$ has 8 computation paths in its oracle query tree, so it can make 6 mind changes and output another value on its 1 remaining computation path. Nevertheless, if the machine makes 6 mind changes, it must still be missing either 01 or 10 in its possible output sequence. This is enough for us to extend the proof of the main theorem and show that

$$\mathrm{PF}_2^{\mathrm{SAT}\|[6]} \subseteq \mathrm{PF}_2^{\mathrm{SAT}[3]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}.$$

In general, we can show that for $j \geq 2$ and $k \geq 3$,

$$\mathrm{PF}_j^{\mathrm{SAT}\|[2^k-2]} \subseteq \mathrm{PF}_j^{\mathrm{SAT}[k]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}.$$

Note that we already know from bounded query language classes (i.e., the 1-bit functions) that

$$\mathrm{PF}_j^{\mathrm{SAT}[k]} \subseteq \mathrm{PF}_j^{\mathrm{SAT}\|[2^k-2]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}},$$

because

$$\mathrm{PF}_j^{\mathrm{SAT}[k]} \subseteq \mathrm{PF}_j^{\mathrm{SAT}\|[2^k-2]} \Longrightarrow \mathrm{P}^{\mathrm{SAT}\|[2^k-1]} \subseteq \mathrm{P}^{\mathrm{SAT}\|[2^k-2]}$$

which collapses the Boolean Hierarchy and which in turn collapses PH.

On the other hand, our techniques fail to show any consequences to the assumption that $\mathrm{PF}_2^{\mathrm{SAT}\|[5]} \subseteq \mathrm{PF}_2^{\mathrm{SAT}[3]}$, because a $\mathrm{PF}_2^{\mathrm{SAT}[3]}$ machine can make 5 mind changes and include all four strings 00, 01, 10 and 11 in its possible output sequence. To compare 3 serial queries to 5 parallel queries, we need to consider functions with 3 bits of output. Thus, we need to construct a 3-bit function $\mathcal{Q}_{53}$ that is easily computable by a $\mathrm{PF}_3^{\mathrm{SAT}\|[5]}$ machine, but is difficult for $\mathrm{PF}_3^{\mathrm{SAT}[3]}$ machines. Here, we point out that the only property of $\mathrm{ODD}_3^{\mathrm{SAT}}$ we needed in the proof of the main theorem is that it agreed with BL_3 on nested inputs. In fact, we could have used an artificially constructed set that agrees with BL_3 on nested inputs and tailor the membership of non-nested inputs according to the needs of the proof. Since a $\mathrm{PF}_3^{\mathrm{SAT}[3]}$ machine that makes 5 mind changes must include as a subsequence

$\langle 000, 111, 000, 111, 000, 111\rangle$, it can include at most 4 of the 8 strings in $\{0,1\}^3$ in its possible output sequence. Again, this is enough for us to show that

$$\mathrm{PF}_3^{\mathrm{SAT}[3]} = \mathrm{PF}_3^{\mathrm{SAT}\|[5]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}.$$

In general, we can show that for $j \geq 3$ and $k \geq 3$,

$$\mathrm{PF}_j^{\mathrm{SAT}[k]} = \mathrm{PF}_j^{\mathrm{SAT}\|[2^k-3]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}.$$

When we continue to generalize the proof along these lines and compare $\mathrm{PF}_r^{\mathrm{SAT}[k]}$ to $\mathrm{PF}_r^{\mathrm{SAT}\|[\ell]}$, we reach a limiting case because we do not necessarily win when a $\mathrm{PF}_r^{\mathrm{SAT}[k]}$ machine fails to include all 2^r strings from $\{0,1\}^r$ in its possible output sequence. We only win when this allows us to eliminate a possible value for $\chi_\ell^{\mathrm{SAT}}(F_1', \ldots, F_\ell')$. In general, we can create at most $\ell - 1$ such situations using our current techniques. Thus, our most general theorem is:

Theorem 10 For all j, k and ℓ such that

$$1 < j \leq k < \ell \leq 2^k - 1,$$

if $2^k - (\ell + 1) < 2^j - 2$ and $2^k - (\ell + 1) < \ell - 1$, then

$$\mathrm{PF}_j^{\mathrm{SAT}\|[\ell]} \subseteq \mathrm{PF}_j^{\mathrm{SAT}[k]} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}.$$

5. Conclusion

We have shown that except for the case of functions with 1-bit output (which are essentially languages), $2^k - 1$ parallel queries to SAT are indeed more powerful than k serial queries to SAT for functions with limited output bits unless the Polynomial Hierarchy collapses. These results clarify some of the issues remaining from previous work on bounded query functions with unlimited output bits [1, 2, 7, 11].

There are still many open questions about parallel versus serial queries to SAT for functions with limited output. For example, our techniques do not show any drastic consequences to the assumption that

$$\mathrm{PF}_2^{\mathrm{SAT}\|[5]} \subseteq \mathrm{PF}_2^{\mathrm{SAT}[3]}.$$

Although we would conjecture that some two-bit function computable by $\mathrm{PF}_2^{\mathrm{SAT}\|[5]}$ cannot be computed by a $\mathrm{PF}_2^{\mathrm{SAT}[3]}$ machine. Similarly we do not know how to handle the case where

$$\mathrm{PF}_3^{\mathrm{SAT}\|[4]} \subseteq \mathrm{PF}_3^{\mathrm{SAT}[3]}.$$

New proof techniques are needed. In particular, we need techniques that avoid the use of the enumerability argument even more than we have managed in this paper.

References

[1] A. Amir, R. Beigel, and W. I. Gasarch. Some connections between bounded query classes and non-uniform complexity. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 232–243, 1990.

[2] A. Amir, R. Beigel, and W. I. Gasarch. Some connections between bounded query classes and non-uniform complexity. Technical Report TR00-024, Electronic Colloquium on Computational Complexity, 2000. URL=`http://www.eccc.uni-trier.de/eccc/`.

[3] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.

[4] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II*, volume 22 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1990.

[5] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Computer Science*, 84(2):199–223, July 1991.

[6] R. Beigel, R. Chang, and M. Ogiwara. A relationship between difference hierarchies and relativized polynomial hierarchies. *Mathematical Systems Theory*, 26(3):293–310, July 1993.

[7] R. Beigel, M. Kummer, and F. Stephan. Approximable sets. *Information and Computation*, 120(2):304–314, 1995.

[8] 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.

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

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

[11] A. Hoene and A. Nickelsen. Counting, selecting, sorting by query-bounded machines. In *Proceedings of the 10th Symposium on Theoretical Aspects of Computer Science*, volume 665 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

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

[13] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[14] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences*, 39(1):84–100, 1989.

[15] K. Wagner. Bounded query computations. In *Proceedings of the 3rd Structure in Complexity Theory Conference*, pages 260–277, June 1988.

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

[17] C. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26(3):287–300, 1983.