Honors Thesis
Bounded Query Functions With Limited Output Bits II

Dalibor Zelený
University of Maryland, Baltimore County

May 29, 2007

Abstract

We solve some open questions in the area of bounded query function classes with limited output bits. In particular, we demonstrate a collapse of the Polynomial Hierarchy to Σ_3^{P} as a consequence of both $\mathrm{PF}_3^{\mathrm{NP}\|[4]} \subseteq \mathrm{PF}_3^{\mathrm{NP}[3]}$ and $\mathrm{PF}_2^{\mathrm{NP}\|[5]} \subseteq \mathrm{PF}_2^{\mathrm{NP}[3]}$. We also generalize the result for the former condition, and discuss some limitations of our generalization.

1 Introduction

The study of bounded query classes gives some insight into the structure of the Polynomial Hierarchy. One way of studying bounded query classes is via the comparison of different oracle access mechanisms. An oracle Turing machine can ask its queries one at a time, and use answers to previous queries to determine the next query. We refer to such queries as serial or adaptive. If, instead, the Turing machine asks all its queries at once, and receives all the answers at once, we call the queries parallel.

The class of languages decided by deterministic polynomial time Turing machines that make k serial queries to an NP oracle is called $\mathrm{P}^{\mathrm{NP}[k]}$, and if the deterministic polynomial time Turing machines make l parallel queries instead, we get the class $\mathrm{P}^{\mathrm{NP}\|[l]}$. We can replace P with PF to get classes of functions that are computed by the Turing machines mentioned earlier. If we add a subscript to PF, for example $\mathrm{PF}_3^{\mathrm{NP}\|[4]}$, we get classes of functions whose outputs have a limited length. We give precise definitions of these complexity classes in the next section.

Our work is motivated by some earlier results. Beigel [2] shows that $\mathrm{P}^{\mathrm{NP}[k]} = \mathrm{P}^{\mathrm{NP}\|[2^k-1]}$. This is equivalent to saying that $\mathrm{PF}_1^{\mathrm{NP}[k]} = \mathrm{PF}_1^{\mathrm{NP}\|[2^k-1]}$. Beigel, Kummer, and Stephan [3] show that this equality doesn't hold for any number of output bits. If we disregard the number of output bits, Beigel, Kummer, and Stephan [3] show that $\mathrm{PF}^{\mathrm{NP}[k]} = \mathrm{PF}^{\mathrm{NP}\|[2^k-1]}$ implies $\mathrm{P} = \mathrm{NP}$. Chang and Squire [7] demonstrate a collapse of the Polynomial Hierarchy to Σ_3^{P} for the special case when $\mathrm{PF}_2^{\mathrm{NP}[3]} = \mathrm{PF}_2^{\mathrm{NP}\|[3]}$, and generalize this result. Their generalization leaves some open questions, some of which we resolve in this paper.

We show that $\mathrm{PF}_2^{\mathrm{NP}\|[5]} \subseteq \mathrm{PF}_2^{\mathrm{NP}[3]}$ and $\mathrm{PF}_3^{\mathrm{NP}\|[4]} \subseteq \mathrm{PF}_3^{\mathrm{NP}[3]}$ both cause a collapse of the Polynomial Hierarchy to Σ_3^{P}. We also generalize the proof of the collapse of PH to Σ_3^{P} for the case $\mathrm{PF}_3^{\mathrm{NP}\|[4]} \subseteq \mathrm{PF}_3^{\mathrm{NP}[3]}$, and get a generalization that encompasses more cases than the one done by Chang and Squire [7].

We gain this improvement by modifying the proof technique used by Chang and Squire [7]. Just like they did, we find a function $\mathcal{Q}$ that can be computed using a $\mathrm{PF}_m^{\mathrm{NP}\|[l]}$ machine, and assume it can be computed by some $\mathrm{PF}_m^{\mathrm{NP}[k]}$ machine. We use this assumption to get a $\leq_m^{\mathrm{P/poly}}$-reduction from BL_l to coBL_l, which causes a collapse of the Boolean Hierarchy, which, in turn, causes a collapse of the Polynomial Hierarchy [10, 12]. We also use the advisees technique—which was first used by Amir, Beigel, and Gasarch [1]—to find one incorrect value of χ_k^{SAT}, which yields a polynomial time algorithm for satisfiability [3] that will be necessary to complete the reduction from BL_l to coBL_l.

Before we present our proofs, we cover some necessary definitions, notations and results in Section 2. After that, in Section 3, we solve the two open questions posed by Chang and Squire, and make a generalization in Section 4. Finally, we discuss some limitations of our work in Section 5.

2 Preliminaries

We present some definitions necessary for the understanding of bounded query functions. The reader should be familiar with basic complexity classes such as P, PF, NP, and P/poly. Knowledge of the NP-complete language SAT and polynomial-time many-one reductions ($\leq_m^{\mathrm{P}}$-reductions) is also assumed. We also assume that the reader is familiar with oracle Turing machines and some complexity classes utilizing them, such as P^{NP} and the Polynomial Hierarchy (PH). The reader can find these definitions in [9, 11]. Some other definitions are given later in this section.

First, we introduce some notation. If x is a string, we will use $|x|$ to denote the length of the string. For a set S, we will use $\|S\|$ to denote the cardinality of S. We use the notation $\langle x_1, \ldots, x_k \rangle$ to denote a k-tuple of strings. We use χ^{SAT} (and χ_k^{SAT} in the multivalued case) for the characteristic function of SAT. Also, unless stated otherwise, all logarithms have base 2.

There are two main ways how an oracle Turing machine can ask its queries. The queries can be asked in a series. We call such queries *serial* or *adaptive* (these two terms will be used interchangably). Another way is to ask all the queries all at once (in parallel), in which case we call the queries *parallel*.

When a Turing machine uses adaptive queries, it uses answers to all previous queries, as well as the input string, to determine the next query. The computation of an oracle Turing machine that uses adaptive queries can be viewed as a tree. We call this tree the *oracle query tree*. The oracle query tree induced by the computation of a $\mathrm{PF}_2^{\mathrm{NP}[2]}$ machine (one that computes a function with a two-bit output and makes two adaptive queries to its SAT oracle) is shown in Figure 1.

Chang and Squire [7] order outputs in the oracle query tree induced by the computation of an oracle Turing machine M on input x. The ordering of the outputs for the oracle query tree in Figure 1 would be $\langle 00, 11, 10, 11 \rangle$. This ordering is called the *output sequence* of M on input x. In general, an output in leaf 1 comes before the output in leaf 2 in the output sequence if the query where paths to the two leaves split is answered "no" on the path to leaf 1 and "yes" on the path to leaf 2. We borrow the notation from [7] and use $\mathrm{OUT}(M, x) = \langle 00, 11, 10, 11 \rangle$ to denote that the output sequence of M on input x is $\langle 00, 11, 10, 00 \rangle$.

Consider the output sequence of an oracle Turing machine that makes adaptive queries. A subsequence of its output sequence consisting of a string of all zeros followed by a string of all ones, or vice versa, is called a *mind change*. In Figure 1, the first and second leaf form a mind change, and also the second and the fourth leaf form a mind change. If we chain these two together, we can say that the machine makes two mind changes or that there is a chain of two mind changes in the output sequence of M on input x.

An oracle Turing machine that uses parallel access to its oracle will compute all its oracle queries and a *truth table*. It will ask the queries at once and get the answers at once. It will then consult the truth table to determine its next course of action.

Now that we have compared the two basic oracle access mechanisms, we can define complexity classes that will be the objects of our study.

Definition 1 (Bounded Query Function Classes) *Let k, l, m be positive integers. We define the following complexity classes:*

$\mathrm{PF}^{\mathrm{NP}[k]}$ *is the class of functions that can be computed by a polynomial-time Turing machine with an* NP *oracle using at most k serial oracle queries.*

$\mathrm{PF}^{\mathrm{NP}\|[l]}$ *is similar to* $\mathrm{PF}^{\mathrm{NP}[k]}$*, but all oracle queries have to be made at once (in parallel) and all the answers will be received at once.*

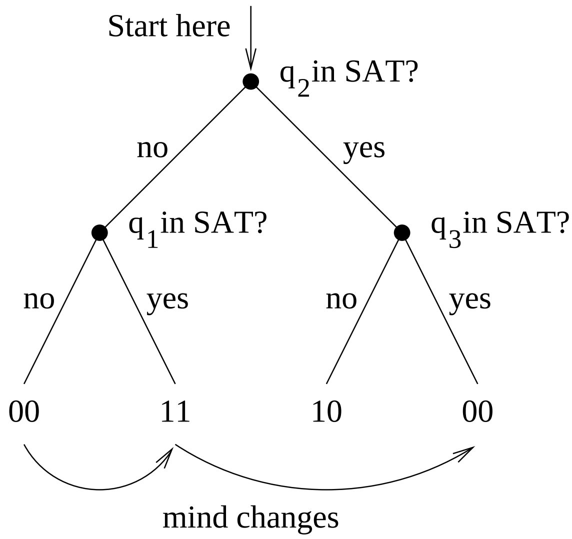

Figure 1: The oracle query tree induced by the computation of a $\mathrm{PF}_2^{\mathrm{SAT}[2]}$ machine. On the path that outputs 11, the machine does some computations and then asks whether q_2 is a satisfiable boolean formula. The SAT oracle says that it is not. The machine uses this answer in its computation and eventually computes another oracle query, q_1. The oracle says that the query is satisifiable. The machine uses this answer and continues computing until it outputs 11. The two arrows also indicate the two mind changes made by this machine.

$\mathrm{PF}_m^{\mathrm{NP}[k]}$ *is a class of functions that output m bits and make at most k serial queries to an* NP *oracle. A similar definition can be made for function classes that make parallel queries and have a limited amount of output bits.*

The language BL_k is commonly used as the $\leq_m^{\mathrm{P}}$-complete language for the k-th level of the Boolean Hierarchy. Individual levels of the Boolean Hierarchy, as well as some of their complete languages, can be found for example in [6]. Even more information about the Boolean Hierarchy can be found in [4, 5]. The complete languages contain k-tuples of boolean formulas that satisfy the predicates shown below.

- $\mathrm{BL}_1 = \mathrm{SAT}$
- $\mathrm{BL}_{2k} = \{\langle x_1, \ldots, x_{2k}\rangle \mid \langle x_1, \ldots, x_{2k-1}\rangle \in \mathrm{BL}_{2k-1} \wedge x_{2k} \in \overline{\mathrm{SAT}}\}$
- $\mathrm{BL}_{2k+1} = \{\langle x_1, \ldots, x_{2k+1}\rangle \mid \langle x_1, \ldots, x_{2k}\rangle \in \mathrm{BL}_{2k} \vee x_{2k+1} \in \mathrm{SAT}\}$
- $\mathrm{coBL}_k = \{\langle x_1, \ldots, x_k\rangle \mid \langle x_1, \ldots, x_k\rangle \notin \mathrm{BL}_k\}$

We can expand the definitions above to get a concrete example of a $\leq_m^{\mathrm{P}}$-complete language for the fourth level of the Boolean Hierarchy. The language BL_4 is a set of 4-tuples of boolean formulas $\langle x_1, x_2, x_3, x_4\rangle$ that satisfy the predicate $[(x_1 \in \mathrm{SAT} \wedge x_2 \notin \mathrm{SAT}) \vee x_3 \in \mathrm{SAT}] \wedge x_4 \notin \mathrm{SAT}$. We can see that a 4-tuple $\langle x_1, x_2, x_3, x_4\rangle$ is in BL_4 if the rightmost satisfiable formula in $\langle x_1, x_2, x_3, x_4\rangle$ has an odd index. This holds for any BL_k. The complement of BL_4 is the language coBL_4 that contains 4-tuples of boolean formulas that satisfy the predicate $[(x_1 \notin \mathrm{SAT} \vee x_2 \in \mathrm{SAT}) \wedge x_3 \notin \mathrm{SAT}] \vee x_4 \in \mathrm{SAT}$.

We call a k-tuple $\langle F_1', \ldots, F_k'\rangle$ of boolean formulas *nested* if there exists an index i such that $F_1', \ldots, F_i'$ are all satisfiable and $F_{i+1}', \ldots, F_k'$ are all unsatisfiable. We can convert any k-tuple $\langle F_1, \ldots, F_k\rangle$ of boolean formulas into a nested one by letting $F_j' = \bigvee_{r \geq j} F_r$. Notice that the rightmost satisfiable formula in $\langle F_1, \ldots, F_k\rangle$ has an odd index if and only if the rightmost satisfiable formula in $\langle F_1', \ldots, F_k'\rangle$ has an odd index. Hence, $\langle F_1, \ldots, F_k\rangle \in \mathrm{BL}_k \iff \langle F_1', \ldots, F_k'\rangle \in \mathrm{BL}_k$.

We define $\mathrm{ODD}_k^{\mathrm{SAT}}$ to be the language of k-tuples of boolean formulas such that an odd number of formulas in the k-tuple is satisfiable. Notice that for nested k-tuples $\langle F_1', \ldots, F_k' \rangle$, $\langle F_1', \ldots, F_k' \rangle \in \mathrm{BL}_k \iff \langle F_1', \ldots, F_k' \rangle \in \mathrm{coBL}_k$.

We define a function $\mathcal{Q}_{43} \in \mathrm{PF}_3^{\mathrm{NP}\|[4]}$ that takes 4-tuples of boolean formulas to bit strings of length 3. The first bit of the function $\mathcal{Q}_{43}$ will be 1 if $\langle F_1, F_2, F_3, F_4 \rangle \in \mathrm{BL}_4$, and 0 otherwise. For nested inputs, we will also require the value of $\mathcal{Q}_{43}$ for nested sequences $\langle F_1', F_2', F_3', F_4' \rangle$ to be 111 or 000. We are not interested in the exact values of $\mathcal{Q}_{43}$ in all cases. Table 1 summarizes the cases we are interested in. We will use this function later in our proof. We will assume that it can be computed by a $\mathrm{PF}_3^{\mathrm{NP}[3]}$ machine and show that this will let us reduce BL_4 to coBL_4, a key step in our proofs.

$F_i \in \mathrm{SAT}$?				
F_1	F_2	F_3	F_4	$\mathcal{Q}_{43}$
0	0	0	0	000
1	0	0	0	111
1	1	0	0	000
1	1	1	0	111
1	1	1	1	000
0	1	0	0	010
0	1	1	0	100
0	1	1	1	011
1	0	1	0	101
1	0	1	1	010
0	0	1	0	110
0	0	1	1	001

Table 1: The values of $\mathcal{Q}_{43}(\langle F_1, F_2, F_3, F_4 \rangle)$ for some combinations of satisfiabilities of the four formulas. The only restriction on values of $\mathcal{Q}_{43}$ for other combintations of satisfiabilities of the four formulas is that the first bit of $\mathcal{Q}_{43}$ indicate whether $\langle F_1, F_2, F_3, F_4 \rangle \in \mathrm{BL}_4$.

We present two more results we will use in our proofs. The first one is a modification of a lemma used by Chang and Squire [7]. The second one is a special case of a result about enumerability proved by Beigel, Kummer, and Stephan [3, Lemma 4.2]. We don't prove either of those. We prove a more general version of Lemma 2 later as Lemma 9, and we refer the reader to [3] for a proof of Theorem 3.

Lemma 2 *Let $\langle F_1', F_2', F_3', F_4' \rangle$ be a nested 4-tuple of boolean formulas. Suppose that a $\mathrm{PF}_3^{\mathrm{NP}[3]}$ machine* M *computes $\mathcal{Q}_{43}$ and that $\mathrm{OUT}(\mathrm{M}, \langle F_1', F_2', F_3', F_4' \rangle)$ doesn't contain $\langle 000, 111, 000, 111, 000 \rangle$ as a subsequence. Then we can compute in polynomial time a 4-tuple $\langle G_1, G_2, G_3, G_4 \rangle$ of boolean formulas such that $\langle F_1', F_2', F_3', F_4' \rangle \in \mathrm{BL}_4 \iff \langle G_1, G_2, G_3, G_4 \rangle \in \mathrm{coBL}_4$.*

Theorem 3 *If there exists a function $f \in \mathrm{PF}$ that outputs k bits, and $\chi_k^{\mathrm{SAT}}(\langle x_1, \ldots, x_k \rangle) \neq f(\langle x_1, \ldots, x_k \rangle)$ for any k-tuple $\langle x_1, \ldots, x_k \rangle$ of boolean formulas, then* $\mathrm{P} = \mathrm{NP}$.

3 Proofs of Open Problems

We prove that $\mathrm{PF}_3^{\mathrm{NP}\|[4]} \subseteq \mathrm{PF}_3^{\mathrm{NP}[3]}$ implies the collapse of the Polynomial Hierarchy to its third level. The proof is similar to the one in [7], but uses a different definition of advisees and uses it to eliminate one possibility for $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2 \rangle)$.

Theorem 4 $\mathrm{PF}_3^{\mathrm{NP}\|[4]} \subseteq \mathrm{PF}_3^{\mathrm{NP}[3]} \Longrightarrow \mathrm{PH} = \Sigma_3^{\mathrm{P}}$

Proof: It is easy to see that $\mathcal{Q}_{43}$ can be computed by a polynomial time Turing machine N that makes at most 4 parallel queries to an NP oracle. We assume that it is also computable by a polynomial time Turing machine M that makes at most 3 serial queries to an NP oracle, and show that in that case it is possible to reduce BL_4 to coBL_4 using a polynomial-time many-one reduction with polynomial advice. By Kadin [10], this implies that $\overline{\mathrm{SAT}} \in \mathrm{NP/poly}$, which causes a collapse of the Polynomial Hierarchy by Yap [12]. To complete the proof, it is necessary to construct a polynomial-time many-one reduction with polynomial advice that reduces BL_4 to coBL_4. In other words, the proof is completed by showing that if $\mathcal{Q}_{43} \in \mathrm{PF}_3^{\mathrm{NP}[3]}$, $\mathrm{BL}_4 \leq_m^{\mathrm{P/poly}} \mathrm{coBL}_4$.

Suppose that a $\mathrm{PF}_3^{\mathrm{NP}[3]}$ machine M can compute $\mathcal{Q}_{43}$. We use this fact to give a $\leq_m^{\mathrm{P/poly}}$ reduction h from BL_4 to coBL_4. The reduction h gets a 4-tuple of boolean formulas $\langle F_1, F_2, F_3, F_4\rangle$ as input. First, it will construct the nested version of the input, $\langle F_1', F_2', F_3', F_4'\rangle$.

Let H_1, H_2 be boolean formulas. We define the set $\mathrm{ADVISEES}(\langle H_1, H_2\rangle)$ as the set of 4-tuples of boolean formulas $\langle x_1, x_2, x_3, x_4\rangle$ such that at least one of the following holds:

1. $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 11$ and $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$ doesn't contain $\langle 000, 111, 000, 111, 000\rangle$ as a subsequence.

2. $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 10$ and at least one of the elements of $\mathrm{OUT}_2 = \{000, 111, 101, 010\}$ is not present in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$.

3. $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 01$ and at least one of the elements of $\mathrm{OUT}_1 = \{000, 010, 100, 011\}$ is not present in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$.

4. $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 00$ and at least one of the elements of $\mathrm{OUT}_0 = \{000, 110, 001\}$ is not present in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$.

Notice that h can compute $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$ in polynomial time because there are only a constant number of computation paths M can take.

Suppose a 4-tuple $\langle F_1', F_2', F_3', F_4'\rangle$ is an advisee of $\langle H_1, H_2\rangle$. For each of the four cases, we describe how to construct in polynomial time a 4-tuple $\langle G_1, G_2, G_3, G_4\rangle$ of boolean formulas such that $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff \langle G_1, G_2, G_3, G_4\rangle \in \mathrm{coBL}_4$.

1. If $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 11$, we can use Lemma 2 to get the desired 4-tuple of boolean formulas.

2. There are four possibilities when $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 10$.

   (a) If 000 is missing in the output sequence, at least one of F_1, F_2, F_3, F_4 is satisfiable. This is because if all were unsatisfiable, M would have to output 000 because it computes $\mathcal{Q}_{43}$. However, 000 is not in the output sequence of M, so M cannot output it. This means that $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff (F_2 \vee F_4) \notin \mathrm{SAT} \vee (F_3 \in \mathrm{SAT} \wedge F_4 \notin \mathrm{SAT})$. We rewrite this as $F_4 \notin \mathrm{SAT} \wedge (F_3 \in \mathrm{SAT} \vee F_2 \notin \mathrm{SAT})$. We can fit this in the normal form for coBL_4 and let $G_1 = F_2$, $G_2 = F_3$, $G_3 = F_4$, $G_4 = \mathrm{FALSE}$. We can use K-map simplification to make this conclusion. Figure 2 illustrates the thought process.

   (b) If 111 is missing in the output sequence, $\chi_4^{\mathrm{SAT}}(\langle F_1, F_2, F_3, F_4\rangle) \neq xx00$ where xx is not 00. It cannot be the case, then, that $(F_1 \vee F_2) \in \mathrm{SAT}$ and $(F_3 \vee F_4) \notin \mathrm{SAT}$. If it were, we would have $\chi_4^{\mathrm{SAT}}(\langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle) = 1000$, which would require M to output 111 that is missing in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$. It follows that $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff F_3 \in \mathrm{SAT} \wedge F_4 \notin \mathrm{SAT}$, and we let $G_1 = \mathrm{TRUE}$, $G_2 = F_3$, $G_3 = F_4$, $G_4 = \mathrm{FALSE}$. Once again, we can use K-maps to see that.

   (c) If 101 is missing in the output sequence, $\chi_4^{\mathrm{SAT}}(\langle F_1, F_2, F_3, F_4\rangle) \neq xx10$, so it is not the case that $F_3 \in \mathrm{SAT}$ and $F_4 \notin \mathrm{SAT}$. Hence, $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff F_1 \in \mathrm{SAT} \wedge (F_2 \vee F_4) \notin \mathrm{SAT}$. We let $G_1 = \mathrm{TRUE}$, $G_2 = F_1$, $G_3 = F_2 \vee F_4$, $G_4 = \mathrm{FALSE}$.

	00	01	11	10
00	0	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	1	1	1

(a) K-map for BL_4.

	00	01	11	10
00	D	0	0	1
01	0	0	0	0
11	0	0	0	0
10	**1**	**1**	**1**	**1**

	00	01	11	10
00	**D**	0	0	**1**
01	0	0	0	0
11	0	0	0	0
10	**1**	1	1	**1**

(b) We know that $\chi_4^{\mathrm{SAT}}(\langle F_1, F_2, F_3, F_4\rangle)$ is not 0000, so we replace that field in the K-map with a "don't care". The two highlighted (in boldface and with larger font) groups cover all the ones.

Figure 2: Using k-maps to find formulas $\langle G_1, G_2, G_3, G_4\rangle$ such that $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff \langle G_1, G_2, G_3, G_4\rangle \in \mathrm{coBL}_4$. Columns correspond to values of $\chi_2^{\mathrm{SAT}}(\langle F_1, F_2\rangle)$, rows correspond to values of $\chi_2^{\mathrm{SAT}}(\langle F_3, F_4\rangle)$. We see that $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4$ if and only if $(F_3 \in \mathrm{SAT} \land F_4 \notin \mathrm{SAT}) \lor (F_1 \in \mathrm{SAT} \land F_4 \notin \mathrm{SAT})$.

(d) If 010 is missing in the output seqeunce, it cannot be the case that $F_4 \in \mathrm{SAT}$, which means that $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff F_3 \in \mathrm{SAT} \lor (F_1 \in \mathrm{SAT} \land F_2 \notin \mathrm{SAT})$. This means that $G_1 = \mathrm{TRUE}$, $G_2 = F_1$, $G_3 = F_2$, $G_4 = F_3$.

3. There are four possibilities when $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 01$.

   (a) If 000 is missing in the output sequence, $\chi_4^{\mathrm{SAT}}(\langle F_1, F_2, F_3, F_4\rangle) \neq x000$. That means that at least one of F_2, F_3, F_4 is satisfiable. Hence, $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff F_3 \in \mathrm{SAT} \land F_4 \notin \mathrm{SAT}$. Then let $G_1 = \mathrm{TRUE}$, $G_2 = F_3$, $G_3 = F_4$, $G_4 = \mathrm{FALSE}$.

   (b) If 010 is missing in the output sequence, that means that $\chi_4^{\mathrm{SAT}}(\langle F_1, F_2, F_3, F_4\rangle) \neq x100$, so it cannot be the case that $F_2 \in \mathrm{SAT}$ and $F_3 \lor F_4 \notin \mathrm{SAT}$. Then $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff (F_3 \in \mathrm{SAT} \land F_4 \notin \mathrm{SAT}) \lor (F_1 \in \mathrm{SAT} \land F_4 \notin \mathrm{SAT})$. This is equivalent to saying that $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff F_4 \notin \mathrm{SAT} \land ((F_1 \lor F_3) \in \mathrm{SAT})$, and we let $G_1 = \mathrm{TRUE}$, $G_2 = F_3$, $G_3 = F_4$, $G_4 = \mathrm{FALSE}$.

   (c) If 100 is missing in the output sequence, it follows that $\chi_4^{\mathrm{SAT}}(\langle F_1, F_2, F_3, F_4\rangle) \neq xx10$, so it cannot be the case that $F_3 \in \mathrm{SAT}$ and $F_4 \notin \mathrm{SAT}$. In that case, $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff F_1 \in \mathrm{SAT} \land (F_2 \lor F_4) \notin \mathrm{SAT}$. Hence, we let $G_1 = \mathrm{TRUE}$, $G_2 = F_1$, $G_3 = F_2 \lor F_4$, $G_4 = \mathrm{FALSE}$.

   (d) If 011 is missing in the output sequence, it cannot be the case that $F_4 \in \mathrm{SAT}$. Then $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff F_3 \in \mathrm{SAT} \lor (F_1 \in \mathrm{SAT} \land F_2 \notin \mathrm{SAT})$, and we let $G_1 = \mathrm{TRUE}$, $G_2 = F_1$, $G_3 = F_2$, $G_4 = F_3$.

4. There are three possiblilities when $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 00$.

   (a) If 000 is missing in the output sequence, $\chi_4^{\mathrm{SAT}}(\langle F_1, F_2, F_3, F_4\rangle) \neq xx00$, so it is not the case that $F_3 \lor F_4 \notin \mathrm{SAT}$. Then $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff F_4 \notin \mathrm{SAT}$, so we let $G_1 = \mathrm{F}_4$, and $G_2 = G_3 = G_4 = \mathrm{FALSE}$.

   (b) If 110 is missing in the output sequence, it follows that $\chi_4^{\mathrm{SAT}}(\langle F_1, F_2, F_3, F_4\rangle) \neq xx10$, so it cannot be the case that $F_3 \in \mathrm{SAT}$ and $F_4 \notin \mathrm{SAT}$. In that case, $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff F_1 \in \mathrm{SAT} \land (F_2 \lor F_4) \notin \mathrm{SAT}$. Hence, we let $G_1 = \mathrm{TRUE}$, $G_2 = F_1$, $G_3 = F_2 \lor F_4$, $G_4 = \mathrm{FALSE}$.

   (c) If 001 is missing in the output seqeunce, it cannot be the case that $F_4 \in \mathrm{SAT}$. Then $\langle F_1, F_2, F_3, F_4\rangle \in \mathrm{BL}_4 \iff F_3 \in \mathrm{SAT} \lor (F_1 \in \mathrm{SAT} \land F_2 \notin \mathrm{SAT})$, and we let $G_1 = \mathrm{TRUE}$, $G_2 = F_1$, $G_3 = F_2$, $G_4 = F_3$.

Let $\langle H_1, H_2\rangle$ and $\langle F_1, F_2, F_3, F_4\rangle$ be given. Furthermore, suppose that $\langle F_1, F_2, F_3, F_4\rangle$ is not an advisee of $\langle H_1, H_2\rangle$. Then it must be the case that

1. $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 11$ implies that $\langle 000, 111, 000, 111, 000\rangle$ appears as a subsequence in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$. We say that the presence of $\langle 000, 111, 000, 111, 000\rangle$ in the output sequence is an indicator of 11 as the value of $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle)$, or simply the *indicator of* 11. Note that the presence of $\langle 000, 111, 000, 111, 000\rangle$ in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$ does not imply that $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = \langle 1, 1\rangle$. It merely indicates a possibility
2. $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 10$ implies that all elements of $\mathrm{OUT}_2 = \{000, 111, 101, 010\}$ are present in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$. The presence of all elements of OUT_2 in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$ is an indicator of 10.
3. $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 01$ implies that all elements of $\mathrm{OUT}_1 = \{000, 010, 100, 011\}$ are present in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$. The presence of all elements of OUT_1 in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$ is an indicator of 01.
4. $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle) = 00$ implies that all elements of $\mathrm{OUT}_0 = \{000, 110, 001\}$ are present in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$. The presence of all elements of OUT_0 in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$ is an indicator of 00.

Notice that for any pair of $\langle H_1, H_2\rangle$ and $\langle F_1, F_2, F_3, F_4\rangle$, it cannot be the case that $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$ contains $\langle 000, 111, 000, 111, 000\rangle$ as a subsequence and also contains all of the elements of all of the three sets OUT_i. This is because there are only eight computation paths in the oracle query tree of M, and M would have to output 000 or 111 on five of them, which would leave us with only three paths for the remaining six bit strings of length 3.

We define $\mathrm{INCORRECT}(H_1, H_2, F_1, F_2, F_3, F_4)$ to be a bit string of length 2 whose indicator is not present in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$. If more than one indicator is not present, we pick an arbitrary bit string of length 2 whose indicator is not present in the output sequence. Notice that if $\langle F_1, F_2, F_3, F_4\rangle$ is not an advisee of $\langle H_1, H_2\rangle$, the indicator of the correct value of $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle)$ is present in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge F_1', H_2 \wedge F_2', F_3', F_4'\rangle)$, so if $\langle F_1, F_2, F_3, F_4\rangle$ is not an advisee of $\langle H_1, H_2\rangle$, $\mathrm{INCORRECT}(H_1, H_2, F_1, F_2, F_3, F_4) \neq \chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle)$.

We now construct polynomial advice that will let us reduce BL_4 to coBL_4 using a $\leq_m^{\mathrm{P/poly}}$ reduction. We assume that an OR of 4 formulas of length n has length at most $9n$, and that an OR of 2 formulas of length n has length at most $5n$. We make this assumption because the formulas we will deal with will be ORs of either 2 or 4 (or less) formulas. The construction starts with sets S_0 consisting of all 4-tuples of boolean formulas of length at most $9n$ and T_0 consisting of all 2-tuples of boolean formulas of length at most $5n$, and proceeds in steps, starting with step 0. We describe the i-th step of the construction.

Step i: For each $\langle H_1, H_2\rangle \in T_i$, find $A_i(\langle H_1, H_2\rangle) = \mathrm{ADVISEES}(\langle H_1, H_2\rangle) \cap S_i$. There are two cases to consider.

1. There is a 2-tuple $\langle H_1, H_2\rangle \in T_i$ such that $\|A_i\| \geq \|S_i\|/32$. We pick one such tuple $\langle H_1, H_2\rangle$, and put $\langle H_1, H_2\rangle$ and $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle)$ in the advice. We also remove $\langle H_1, H_2\rangle$ from T_i to form T_{i+1}, and remove $A_i(\langle H_1, H_2\rangle)$ from S_i to get S_{i+1}.

   If $\|S_{i+1}\| \leq 16$, we put all 4-tuples $\langle x_1, x_2, x_3, x_4\rangle \in S_{i+1}$ and $\chi_4^{\mathrm{SAT}}(\langle x_1, x_2, x_3, x_4\rangle)$ in the advice. After that, we terminate the advice construction, and indicate in the advice that the construction terminated in case 1. We let $S = S_{i+1}$ and $T = T_{i+1}$.

2. For all 2-tuples $\langle H_1, H_2\rangle \in T_i$, $\|A_i\| < \|S_i\|/32$. This means that for all $\langle H_1, H_2\rangle \in T_i$, $\mathrm{Prob}_{\mathbf{x} \in S_i}[\mathbf{x} \in \mathrm{ADVISEES}(\langle H_1, H_2\rangle)] < 1/32$. Then there exists a sequence s with a polynomial number of 4-tuples $\langle x_1, x_2, x_3, x_4\rangle \in S_i$ such that for all 2-tuples $\langle H_1, H_2\rangle \in T_i$, we have $\mathrm{INCORRECT}(H_1, H_2, x_1, x_2, x_3, x_4) = \chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle)$ for less than 1/4 of the 4-tuples in s. We put $\langle x_1, x_2, x_3, x_4\rangle$ and $\chi_2^{\mathrm{SAT}}(\langle x_1, x_2, x_3, x_4\rangle)$ in the advice for each element of the polynomial-size sequence s and terminate the advice construction. We indicate in the advice that its construction terminated in case 2. We let $S = S_i$ and $T = T_i$.

We show that the polynomial sequence s of elements of S_i mentioned in the previous paragraph exists. First note that $\|T\| \leq 2^{10n}$. For all elements $\langle H_1, H_2\rangle \in T$ we have $\mathrm{Prob}_{\mathbf{x}\in S}[\mathbf{x} \in \mathrm{ADVISEES}(\langle H_1, H_2\rangle)] < 1/32$ by construction. We show that there exists a sequence of $44n + 4$ 4-tuples in S such that for all $\langle H_1, H_2\rangle \in T$, we have $\mathrm{INCORRECT}(H_1, H_2, x_1, x_2, x_3, x_4) = \chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle)$ for at most $11n$ of those 4-tuples.

Let $\langle H_1, H_2\rangle$ be given. Suppose we pick uniformly at random $44n + 4$ elements $\langle x_1, x_2, x_3, x_4\rangle$ of S and compute $\mathrm{INCORRECT}(H_1, H_2, x_1, x_2, x_3, x_4)$ for each of them. We are interested in an upper bound on the probability that $\mathrm{INCORRECT}(H_1, H_2, x_1, x_2, x_3, x_4) = \chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle)$ for $11n + 1$ or more of the elements we picked.

The following theorem can be found in [8] as Theorem C.2.

Theorem 5 *The probability that at least k out of n Bernoulli trials are successful is at most $\binom{n}{k}p^k$, where p is the probability of success for an individual trial.*

Using Theorem 5, we see that the probability that $\mathrm{INCORRECT}(H_1, H_2, x_1, x_2, x_3, x_4) = \chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle)$ for $11n + 1$ or more of the elements we picked is at most

$$\binom{44n+4}{11n+1}\left(\frac{1}{32}\right)^{11n+1} \leq 2^{44n+4}\left(\frac{1}{2}\right)^{55n+5} = \left(\frac{1}{2}\right)^{11n+1}.$$

Then the probability that for a particular sequence of $44n + 4$ elements of S there exists a 2-tuple $\langle H_1, H_2\rangle$ such that $\mathrm{INCORRECT}(H_1, H_2, x_1, x_2, x_3, x_4) = \chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle)$ for at least $11n+1$ 4-tuples $\langle x_1, x_2, x_3, x_4\rangle$ in that sequence is at most

$$2^{10n}\left(\frac{1}{2}\right)^{11n+1} = 2^{10n-11n-1} = \left(\frac{1}{2}\right)^{n+1} < \frac{1}{2} \qquad \text{for all } n \geq 1.$$

This implies that there exists a sequence of 4-tuples from S of length $44n + 4$ such that for every $\langle H_1, H_2\rangle \notin T$, $\mathrm{INCORRECT}(H_1, H_2, x_1, x_2, x_3, x_4) = \chi_2^{\mathrm{SAT}}(\langle H_1, H_2\rangle)$ for at most $11n$ of the 4-tuples in that sequence.

We now show how to use the advice to reduce BL_4 to coBL_4 in polynomial time. Our goal is to construct a 4-tuple $\langle G_1, G_2, G_3, G_4\rangle$ of boolean formulas such that $\langle x_1, x_2, x_3, x_4\rangle \in \mathrm{BL}_4$ if and only if $\langle G_1, G_2, G_3, G_4\rangle \in \mathrm{coBL}_4$.

Assume that a $\mathrm{PF}_3^{\mathrm{NP}[3]}$ machine M computes $\mathcal{Q}_{43}$. We can assume without loss of generality that all formulas in $\langle x_1, x_2, x_3, x_4\rangle$ have the same length, n. On input $\langle x_1, x_2, x_3, x_4\rangle$:

1. Construct the nested version of $\langle x_1, x_2, x_3, x_4\rangle$, $\langle x_1', x_2', x_3', x_4'\rangle$.
2. For each 2-tuple $\langle H_1, H_2\rangle$ in the advice, compute $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge x_1', H_2 \wedge x_2', x_3', x_4'\rangle)$. If $\langle x_1', x_2', x_3', x_4'\rangle$ turns out to be an advisee of some $\langle H_1, H_2\rangle$ in the advice, we can construct $\langle G_1, G_2, G_3, G_4\rangle$ using the method we discussed earlier.
3. If advice construction terminated in case 1 of the construction and we didn't constrcut $\langle G_1, G_2, G_3, G_4\rangle$ in the previous step, this means that $\langle x_1, x_2, x_3, x_4\rangle$ is in the advice together with $\chi_4^{\mathrm{SAT}}(\langle x_1, x_2, x_3, x_4\rangle)$. In that case we output a trivial $\langle G_1, G_2, G_3, G_4\rangle$ ($G_1 = G_2 = G_3 = G_4 = \mathrm{TRUE}$ if $\langle x_1, x_2, x_3, x_4\rangle \in \mathrm{BL}_4$ and $G_1 = G_2 = G_3 = \mathrm{TRUE}$, $G_4 = \mathrm{FALSE}$ if $\langle x_1, x_2, x_3, x_4\rangle \notin \mathrm{BL}_4$).
4. If advice construction terminated in case 2 of the construction, we first check if at least one of the elements in the 4-tuple $\langle x_1, x_2, x_3, x_4\rangle$ is in the advice string as one of the formulas. If it is, we also know whether it's satisfiable or not, and we have enough information to construct $\langle G_1, G_2, G_3, G_4\rangle$.

5. If none of the previous four cases apply, we use the advice string to decide the membership of each of x_1, x_2, x_3, x_4 in SAT in polynomial time using Theorem 3. All we need to show is that our advice enables us to compute a function $f(\langle y_1, y_2\rangle)$ that outputs 2 bits and $f(\langle y_1, y_2\rangle) \neq \chi_2^{\mathrm{SAT}}(\langle y_1, y_2\rangle)$ for any boolean formulas y_1, y_2 of length at most $5n$.

   Either $\langle y_1, y_2\rangle$ is in the advice string together with its characteristic function, or it had less than $\|S\|$ advisees in S at the end of advice construction. In the former case, we know $\chi_2^{\mathrm{SAT}}(\langle y_1, y_2\rangle)$ and can pick any arbitrary value that's not $\chi_2^{\mathrm{SAT}}(\langle y_1, y_2\rangle)$ as $f(\langle y_1, y_2\rangle)$. In the latter case, we compute $\mathrm{INCORRECT}(y_1, y_2, F_1, F_2, F_3, F_4)$ for each 4-tuple in the advice. By construction, any value that appears as the output of INCORRECT for at least 1/4 of the 4-tuples in the advice cannot be the correct value of $\chi_2^{\mathrm{SAT}}(\langle y_1, y_2\rangle$, so we pick any such value as $f(\langle y_1, y_2\rangle)$. Since the advice has polynomial length and we can compute indicators in polynomial time, we can compute f in polynomial time.

This completes the proof. □

Now we turn our attention to the other open problem proposed by Chang and Squire [7]. We will show that the containment $\mathrm{PF}_2^{\mathrm{NP}\|[5]} \subseteq \mathrm{PF}_2^{\mathrm{NP}[3]}$ also implies a collapse of the Polynomial Hierarchy. This proof will be even more similar to the one done by Chang and Squire [7] than the one we just did.

Theorem 6 $\mathrm{PF}_2^{\mathrm{NP}\|[5]} \subseteq \mathrm{PF}_2^{\mathrm{NP}[3]} \Longrightarrow \mathrm{PH} = \Sigma_3^{\mathrm{P}}$

Proof Sketch: We will modify the proof of Theorem 4. We will define the first bit of the function $\mathcal{Q}_{52}(\langle x_1, x_2, x_3, x_4, x_5\rangle)$ to be 1 if $\langle x_1, x_2, x_3, x_4, x_5\rangle \in \mathrm{BL}_5$ and 0 otherwise. The second bit will indicate whether $\langle x_1, x_2, x_3, x_4, x_5\rangle \in \mathrm{ODD}_5^{\mathrm{SAT}}$. This function is computable by a $\mathrm{PF}_2^{\mathrm{NP}\|[5]}$ machine N. We will assume that it is also computable by a $\mathrm{PF}_2^{\mathrm{NP}[3]}$ machine M. Since this function is different from the one used in the proof of Theorem 4, we have to rephrase Lemma 2.

Lemma 7 *Suppose that a* $\mathrm{PF}_2^{\mathrm{NP}[3]}$ *machine* M *can compute* $\mathcal{Q}_{52}$ *for nested inputs* $\langle F_1', F_2', F_3', F_4', F_5'\rangle$*. If its output sequence on input* $\langle F_1', F_2', F_3', F_4', F_5'\rangle$ *doesn't contain* $\langle 00, 11, 00, 11, 00, 11\rangle$ *as a subsequence, there exists a* 5*-tuple* $\langle G_1, G_2, G_3, G_4, G_5\rangle$ *of boolean formulas computable in polynomial time such that* $\langle F_1', F_2', F_3', F_4', F_5'\rangle \in \mathrm{BL}_5 \iff \langle G_1, G_2, G_3, G_4, G_5\rangle \in \mathrm{coBL}_5$*.*

Instead of using 2-tuples $\langle H_1, H_2\rangle$ of boolean formulas, we will have single boolean formulas H as advisors of 5-tuples of boolean formulas. We will say that $\langle x_1, x_2, x_3, x_4, x_5\rangle$ of boolean formulas is an advisee of H if at least one of the following happens.

1. $H \in \mathrm{SAT}$ and $\mathrm{OUT}(\mathrm{M}, \langle H \wedge F_1', F_2', F_3', F_4', F_5'\rangle)$ doesn't contain $\langle 00, 11, 00, 11, 00, 11\rangle$ as a subsequence.

2. $H \notin \mathrm{SAT}$ and $\mathrm{OUT}(\mathrm{M}, \langle H \wedge F_1', F_2', F_3', F_4', F_5'\rangle)$ contains $\langle 00, 11, 00, 11, 00, 11\rangle$ as a subsequence.

We show how construct a 5-tuple $\langle G_1, G_2, G_3, G_4, G_5\rangle$ that belongs to coBL_5 if and only if $\langle F_1, F_2, F_3, F_4, F_5\rangle \in \mathrm{BL}_5$, where $\langle F_1, F_2, F_3, F_4, F_5\rangle$ is an advisee of H.

1. If $H \in \mathrm{SAT}$ is an advisor of $\langle F_1, F_2, F_3, F_4, F_5\rangle$, we use Lemma 7.

2. If $H \notin \mathrm{SAT}$ is an advisor of $\langle F_1, F_2, F_3, F_4, F_5\rangle$, we notice that the output sequence of M on input $\langle H \wedge F_1', F_2', F_3', F_4', F_5'\rangle$ contains at most two terms that are neither 00 nor 11. Without loss of generality assume there are two of them, and call them a and b. Notice that $\mathcal{Q}_{52}(\langle H \wedge F_1', F_2', F_3', F_4', F_5'\rangle)$ cannot be 11 because if the rightmost satisfiable formula in $\mathcal{Q}_{52}(\langle H \wedge F_1', F_2', F_3', F_4', F_5'\rangle)$ has an even index, then, since $\langle F_1', F_2', F_3', F_4', F_5'\rangle$ is nested, $\langle H \wedge F_1', F_2', F_3', F_4', F_5'\rangle \notin \mathrm{ODD}_5^{\mathrm{SAT}}$. Now there are three cases to consider.

(a) If $a = b = 10$, we know that $\langle H \wedge F'_1, F'_2, F'_3, F'_4, F'_5 \rangle \notin \mathrm{ODD}_5^{\mathrm{SAT}}$ because all leaves of the oracle query tree induced by the computation of M on input $\langle H \wedge F'_1, F'_2, F'_3, F'_4, F'_5 \rangle$ that hold the right value of $\mathcal{Q}_{52}$ have 0 as its second bit, which means that the rightmost satisfiable formula in $\langle H_1, H_2, H_3, H_4, H_5 \rangle$ doesn't have an even index. This means that $\langle F_1, F_2, F_3, F_4, F_5 \rangle \in \mathrm{BL}_5$ if and only if at least one of F_1, F_3, F_5 is satisfiable. We let $G_1 = G_2 = G_3 = \mathrm{TRUE}$, $G_4 = F_1 \vee F_3 \vee F_5$, and $G_5 = \mathrm{FALSE}$.

(b) If $a = b = 01$, it must be the case that $F_4 \notin \mathrm{SAT} \vee F_5 \in \mathrm{SAT}$ because 10 is not present in the output sequence. Then $\langle F_1, F_2, F_3, F_4, F_5 \rangle \in \mathrm{BL}_5$ if and only if $[F_1 \in \mathrm{SAT} \wedge (F_2 \vee F_4) \notin \mathrm{SAT}] \vee F_3 \in \mathrm{SAT}$, and we let $G_1 = \mathrm{TRUE}$, $G_2 = F_1$, $G_3 = F_2 \vee F_4$, $G_4 = F_3$, and $G_5 = \mathrm{FALSE}$.

(c) If $a = 10$ and $b = 01$ or vice versa, 01 appears only once in the output sequence of M on input $\langle H \wedge F'_1, F'_2, F'_3, F'_4, F'_5 \rangle$. The machine M asks queries q_1, q_2, and q_3 on the computation path of on which 01 is output. For simplicity, call this path p. Some of the queries on path p are answered "yes", some "no". We take all the "yes" queries and AND them together to get a boolean formula τ, and OR all the "no" queries together to get a boolean formula φ. The machine M will follow path p if $\tau \in \mathrm{SAT}$ and $\varphi \notin \mathrm{SAT}$. Notice that if p is the path that M takes, $\langle H \wedge F'_1, F'_2, F'_3, F'_4, F'_5 \rangle \in \mathrm{ODD}_5^{\mathrm{SAT}}$, which means $\langle F_1, F_2, F_3, F_4, F_5 \rangle \notin \mathrm{BL}_5$, so $\langle F_1, F_2, F_3, F_4, F_5 \rangle \in \mathrm{BL}_5$ only if p is not the path that M takes on input $\langle F_1, F_2, F_3, F_4, F_5 \rangle$. If p is not the path that M takes, we're in case (a), and $\langle F_1, F_2, F_3, F_4, F_5 \rangle \in \mathrm{BL}_5$ if and only if $F_1 \vee F_3 \vee F_5 \in \mathrm{SAT}$. Then $\langle F_1, F_2, F_3, F_4, F_5 \rangle \in \mathrm{BL}_5$ if and only if $(\tau \notin \mathrm{SAT} \vee \varphi \in \mathrm{SAT}) \wedge (F_1 \vee F_3 \vee F_5) \in \mathrm{SAT}$, which happens if and only if $(\tau \notin \mathrm{SAT} \wedge (F_1 \vee F_3 \vee F_5) \in \mathrm{SAT}) \vee (\varphi \wedge (F_1 \vee F_3 \vee F_5)) \in \mathrm{SAT}$. We let $G_1 = \mathrm{TRUE}$, $G_2 = F_1 \vee F_3 \vee F_5$, $G_3 = \tau$, $G_4 = \varphi \wedge (F_1 \vee F_3 \vee F_5)$, and $G_5 = \mathrm{FALSE}$.

The advice is constructed the same way as in the proof of Theorem 4. It is used differently, however. Instead of elminating one value of $\chi_2^{\mathrm{SAT}}(\langle H_1, H_2 \rangle)$ as we did in the proof of Theorem 4, we will use the idea of Chang and Squire [7], and notice that if H is not an advisor of $\langle F_1, F_2, F_3, F_4, F_5 \rangle$, $H \in \mathrm{SAT}$ if and only if $\mathrm{OUT}(\mathrm{M}, \langle H \wedge F'_1, F'_2, F'_3, F'_4, F'_5 \rangle)$ contains $\langle 00, 11, 00, 11, 00, 11 \rangle$ as a subsequence. The polynomial-length sequence of 5-tuples in the advice will, instead, have the property that if H is not in the advice, H will be satisfiable if and only if for a three-fourths' majority of 5-tuples in the advice, $\mathrm{OUT}(\mathrm{M}, \langle H \wedge F'_1, F'_2, F'_3, F'_4, F'_5 \rangle)$ contains $\langle 00, 11, 00, 11, 00, 11 \rangle$ as a subsequence. An argument similar to the one made in the previous proof can be made to show that such polynomial-size set of 5-tuples exists. □

4 Generalization

The technique used in the proof of Theorem 4 in the previous section can be generalized to prove the following theorem.

Theorem 8 *For all k, l and m such that $1 < m \leq k < l \leq 2^k - 1$, if $l > 2^k - 2^m + 1$ then* $\mathrm{PF}_m^{\mathrm{NP}\|[l]} \subseteq \mathrm{PF}_m^{\mathrm{NP}[k]} \Rightarrow \mathrm{PH} = \Sigma_3^{\mathrm{P}}$.

Proof: We need to make a few modifications to the proof of Theorem 4 from the previous section. The necessary modifications are listed below.

1. Define a function $\mathcal{Q}$ similar to $\mathcal{Q}_{43}$.

2. Generalize Lemma 2.

3. Generalize the definition of ADVISEES() using $\mathcal{Q}$ and the $\mathrm{PF}_m^{\mathrm{NP}[k]}$ machine M that can compute it.

4. Show how to reduce BL_l to coBL_l when some l-tuple of boolean formulas is an advisee.

5. Revise the construction of the advice.

6. Argue that if advice construction terminates in case 2, that is, with some l-tuples that neither have an advisor in the advice nor are in the advice themselves, we can still, given an l-tuple of formulas $\langle x_1, \ldots, x_l \rangle$, eliminate one possibility for $\chi_l^{\mathrm{SAT}}(\langle x_1, \ldots, x_l \rangle)$.

We carry out these steps in the order listed. We start with a generalization of the function $\mathcal{Q}_{43}$. We define the function $\mathcal{Q}$ to be a function mapping l-tuples $\langle F_1, \ldots, F_l \rangle$ of boolean formulas to binary strings of length m. If an l-tuple $\langle F_1, \ldots, F_l \rangle$ of boolean formulas is nested, all bits of the output are the same and indicate whether $\langle F_1, \ldots, F_l \rangle \in \mathrm{BL}_l$. For non-nested inputs, $\mathcal{Q}$ outputs the last m bits of $\chi_l^{\mathrm{SAT}}(\langle F_1, \ldots, F_l \rangle)$. The second part can be done because $l > m$, and we do it because we want $\mathcal{Q}$ to be onto, as we see later. It is also easy to see that $\mathcal{Q} \in \mathrm{PF}_m^{\mathrm{NP}\|[l]}$. Finally, let s be a bit string of length m. Notice that all non-nested k-tuples $\langle F_1, \ldots, F_l \rangle$ for which $\chi_m^{\mathrm{SAT}}(\langle F_{l-m+1}, \ldots, F_l \rangle) = s$ will map to the same string of length m under $\mathcal{Q}$, which is another desired property of $\mathcal{Q}$.

We show that $\mathcal{Q}$ is an onto mapping. Pick a bit string s of length m. We will find a non-nested l-tuple of boolean formulas $\mathbf{F}$ such that $\mathcal{Q}(\mathbf{F}) = s$. There exists an m-tuple $\langle F_1, \ldots, F_m \rangle$ of boolean formulas such that $\chi_m^{\mathrm{SAT}}(\langle F_1, \ldots, F_m \rangle) = s$. We append this m-tuple at the end of an $(l-m)$-tuple $\langle \mathrm{FALSE}, \ldots, \mathrm{FALSE} \rangle$. Now $\mathbf{F} = \mathcal{Q}(\langle \mathrm{FALSE}, \ldots, \mathrm{FALSE}, F_1, \ldots, F_m \rangle)$ is an l-tuple of boolean formulas that maps to s if $\mathbf{F}$ is non-nested, which happens if at least one of the formulas in $\langle F_1, \ldots, F_m \rangle$ is satisfiable. If all formulas in $\langle F_1, \ldots, F_m \rangle$ are unsatisfiable, F is a nested l-tuple of unsatisfiable boolean formulas. Then $\mathcal{Q}(\mathbf{F}) = s = 00\ldots0$ because $\mathbf{F} \notin \mathrm{BL}_l$.

Lemma 9 (Generalization of Lemma 2) *Let $\langle F_1', \ldots, F_l' \rangle$ be a nested l-tuple of boolean formulas. Suppose that a $\mathrm{PF}_m^{\mathrm{NP}[k]}$ machine* M *computes $\mathcal{Q}$ and that* $\mathrm{OUT}(\mathrm{M}, \langle F_1', \ldots, F_l' \rangle)$ *doesn't contain as a subsequence a sequence of length $l+1$ with a chain of l mind changes starting with a string of all zeros. Then we can compute in polynomial time an l-tuple $\langle G_1, \ldots, G_l \rangle$ of boolean formulas such that $\langle F_1', \ldots, F_l' \rangle \in \mathrm{BL}_l \iff \langle G_1, \ldots, G_l \rangle \in \mathrm{coBL}_l$.*

Proof: A special case of the proof was sketched in [7]. We give a proof for the general case.

Nodes in the oracle query tree correspond to queries made by M. If a query is answered "no", M will take the path that goes to the left child of the query that was answered "no". Otherwise it will take the path that goes right. We will label the queries using an inorder traversal of the oracle query tree. The leftmost query will be called q_1; the rightmost one, q_{2^k-1}. We will label each leaf in the oracle query tree with the index of the query with the highest index that was answered "yes" on the path to that leaf. Figure 3 demonstrates this notation using a $\mathrm{PF}_2^{\mathrm{NP}[3]}$ machine as an example.

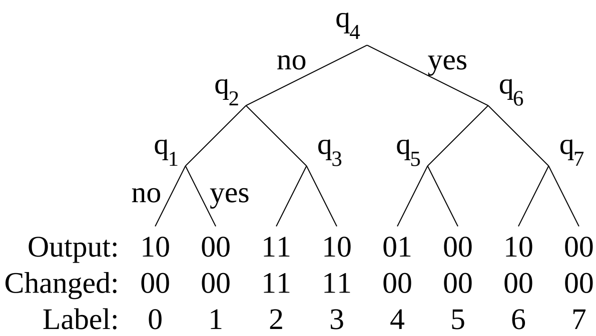

Figure 3: We label leaves of the oracle query tree induced by the computation of a $\mathrm{PF}_2^{\mathrm{NP}[3]}$ machine based on the index of the highest query that was answered "yes" on the path to that leaf. If a query is answered "yes", the machine follows the path that goes to the right from the node representing that query. Otherwise it follows the path that goes left. Since only outputs 00 and 11 make sense on nested inputs, we change outputs other than those to the "nearest" output that is 00 or 11, breaking ties arbitrarily.

Figure 3 also shows outputs made by the $\mathrm{PF}_2^{\mathrm{NP}[3]}$ machine. However, M can output only all-zero or all-one strings on nested inputs. Therefore, without loss of generality, we can make any output that is neither all zeros nor all ones the same as the all-zero or all-one output that's in the nearest leaf, breaking ties arbitrarily. This is also demonstrated in Figure 3. We can do this because we know that M will not follow any path leading to an output that is neither all zeros nor all ones. This also doesn't affect the length of the longest chain of mind changes. As a consequence, we now have blocks of all-zero or all-one strings in the modified leaves of the oracle query tree.

Define φ_i to be the AND of all queries q_j that are answered "yes" on the path to leaf i. We define $\varphi_0 = \mathrm{TRUE}$ because all queries on the path to leaf 0 are answered "no". Finally, we define

$$\phi_i = \bigvee_{j \geq i} \varphi_j.$$

To give some examples, consider the oracle query tree in Figure 3. We will have $\varphi_7 = q_4 \wedge q_6 \wedge q_7$, $\varphi_6 = q_4 \wedge q_6$, $\varphi_5 = q_4 \wedge q_5$, $\phi_6 = (q_4 \wedge q_6 \wedge q_7) \vee (q_4 \wedge q_6)$, and $\phi_5 = (q_4 \wedge q_6 \wedge q_7) \vee (q_4 \wedge q_6) \vee (q_4 \wedge q_5)$.

Notice that $\langle F_1', \ldots, F_l' \rangle \in \mathrm{BL}_l$ only if M outputs a string of all ones. We can also say that $\langle F_1', \ldots, F_l' \rangle \in \mathrm{BL}_l$ only if M follows a path where we set the output to be all ones (even though it may have been something else originally).

Recall that there is no chain of l mind changes starting with an all-zeros string in the output sequence of M. Suppose that all chains of l mind changes in the output sequence of M start with an all-ones string. Also recall that all strings in the output sequence that were not all-zero or all-one were changed to the nearest all-zero or all-one string. This creates $l+1$ "clusters" of identical strings in the output sequence of M. Then $\langle F_1', \ldots, F_l' \rangle \in \mathrm{BL}_l$ if M follows a path that leads to some cluster of all-one strings. Suppose that the first leaf in a cluster of all-one strings has index i and that the last leaf in that cluster has index I. Then M takes a path leading to that cluster if $\phi_i \in \mathrm{SAT}$ and $\phi_{I+1} \notin \mathrm{SAT}$. In Figure 3, the machine M will follow a path that leads to the only cluster of all-one outputs if $\phi_2 \in \mathrm{SAT}$ and $\phi_4 \notin \mathrm{SAT}$. If $i = 0$, we drop the $\phi_i \in \mathrm{SAT}$ requirement, and if $I = 2^k - 1$, we drop the $\phi_{I+1} \notin \mathrm{SAT}$ requirement.

There are $\lceil l/2 \rceil - 1$ clusters of all-one strings for which neither $i = 0$ nor $I = 2^k - 1$. The first cluster of all-one strings has $i = 0$. If l is even, there is also an additional cluster of all-one strings with $I = 2^k - 1$. We give each cluster (from left to right) a number $j \in \{0, 1, \ldots, \lceil l/2 \rceil - 1\}$ if l is odd and $j \in \{0, 1, \ldots, \lceil l/2 \rceil\}$ if l is even, and call i_j the leftmost leaf in cluster j and I_j the rightmost leaf in cluster j.

If l is odd, $\langle F_1', \ldots, F_l' \rangle \in \mathrm{BL}_l$ if

$$\phi_{I_0+1} \notin \mathrm{SAT} \vee (\phi_{i_1} \in \mathrm{SAT} \wedge \phi_{I_1+1} \notin \mathrm{SAT}) \vee \cdots \vee (\phi_{i_{\lceil l/2 \rceil - 1}} \in \mathrm{SAT} \wedge \phi_{I_{\lceil l/2 \rceil - 1}+1}),$$

which can be rewritten as

$$\left(\left[(\phi_{I_0+1} \notin \mathrm{SAT} \vee \phi_{i_1} \in \mathrm{SAT}) \wedge \phi_{I_1+1} \notin \mathrm{SAT} \right] \vee \cdots \vee \phi_{i_{\lceil l/2 \rceil - 1}} \in \mathrm{SAT} \right) \wedge \phi_{I_{\lceil l/2 \rceil - 1}+1} \tag{1}$$

because $\langle \phi_0, \ldots, \phi_{2^k-1} \rangle$ is nested. If l is even, we get that $\langle F_1', \ldots, F_l' \rangle \in \mathrm{BL}_l$ if

$$\left[\left(\left[(\phi_{I_0+1} \notin \mathrm{SAT} \vee \phi_{i_1} \in \mathrm{SAT}) \wedge \phi_{I_1+1} \notin \mathrm{SAT} \right] \vee \cdots \vee \phi_{i_{\lceil l/2 \rceil - 1}} \in \mathrm{SAT} \right) \wedge \phi_{I_{\lceil l/2 \rceil - 1}+1} \right] \vee \phi_{i_{\lceil l/2 \rceil}}. \tag{2}$$

But then if l is odd, $\langle F_1', \ldots, F_l' \rangle \in \mathrm{BL}_l$ if and only if $\langle \phi_{I_0+1}, \phi_{i_1}, \phi_{I_1+1}, \ldots, \phi_{i_{\lceil l/2 \rceil - 1}}, \phi_{I_{\lceil l/2 \rceil - 1}+1} \rangle \in \mathrm{coBL}_l$. Similarly, if l is even, $\langle F_1', \ldots, F_l' \rangle \in \mathrm{BL}_l$ if and only if $\langle \phi_{I_0+1}, \phi_{i_1}, \phi_{I_1+1}, \ldots, \phi_{i_{\lceil l/2 \rceil - 1}}, \phi_{I_{\lceil l/2 \rceil - 1}+1}, \phi_{i_{\lceil l/2 \rceil}} \rangle \in \mathrm{coBL}_l$. Hence, we let $G_1 = \phi_{I_0+1}$, $G_2 = \phi_{i_1}$, and so on until $G_l = \phi_{I_{\lceil l/2 \rceil - 1}+1}$ if l is odd and $G_l = \phi_{i_{\lceil l/2 \rceil}}$ if l is even.

If the longest chain of mind changes is shorter than l, we remove parts of the expressions in (1) or (2) and the formulas $G_1, \ldots, G_l$ can still be constructed. $\square$

Fix an l-tuple $\langle H_1, \ldots, H_l \rangle$ of boolean formulas. Let $\chi_l^{\mathrm{SAT}}(\langle H_1, \ldots, H_l \rangle) = t$ and let $s_1, \ldots, s_r$ be all possible strings that can be output by $\mathcal{Q}$ on inputs $\langle F_1' \wedge H_1, \ldots, F_l' \wedge H_l \rangle$ where $\langle F_1', \ldots, F_l' \rangle$ is the nested version of some l-tuple $\langle F_1, \ldots, F_l \rangle$. The presence of all the strings $s_1, \ldots, s_r$ in $\mathrm{OUT}(\mathrm{M}, \langle F_1' \wedge H_1, \ldots, F_l' \wedge H_l \rangle)$ is an indicator that $\chi_l^{\mathrm{SAT}}(\langle H_1, \ldots, H_l \rangle) = t$. As in the previous section, the indicator that $\chi_l^{\mathrm{SAT}}(\langle H_1, \ldots, H_l \rangle) = 11\ldots1$ is the occurrence of a chain of l mind changes, starting with $00\ldots0$, in $\mathrm{OUT}(\mathrm{M}, \langle F_1' \wedge H_1, \ldots, F_l' \wedge H_l \rangle)$. We say that $\langle F_1, \ldots, F_l \rangle$ is an advisee of $\langle H_1, \ldots, H_l \rangle$ if the indicator of the true value of $\chi_l^{\mathrm{SAT}}(\langle H_1, \ldots, H_l \rangle)$ is not present in $\mathrm{OUT}(\mathrm{M}, \langle F_1' \wedge H_1, \ldots, F_l' \wedge H_l \rangle)$ in its entirety.

Now we need to show how to find an l-tuple $\langle G_1, \ldots, G_l \rangle$ of boolean formulas such that $\langle F_1, \ldots, F_l \rangle \in \mathrm{BL}_l \iff \langle G_1, \ldots, G_l \rangle \in \mathrm{coBL}_l$, given that $\langle F_1, \ldots, F_l \rangle$ is an advisee of $\langle H_1, \ldots, H_l \rangle$. Fix a nested l-tuple $\langle x_1', \ldots, x_l' \rangle$ and suppose that $\mathcal{Q}(\langle x_1' \wedge H_1, \ldots, x_l' \wedge H_l \rangle) = s$ is not in $\mathrm{OUT}(\mathrm{M}, \langle F_1' \wedge H_1, \ldots, F_l' \wedge H_l \rangle)$. Let j be the largest integer such that $H_j \wedge x_j'$ is satisfiable, and let J be the smallest integer greater than j such that $H_j \in \mathrm{SAT}$. Table 2 illustrates the definitions of j and J.

Index (i)	1	2	3	4	5	6	7	8
$x_i' \in \mathrm{SAT}$?	1	1	1	1	1	0	0	0
$H_i \in \mathrm{SAT}$?	1	0	0	1	0	0	1	1

Table 2: Examples of the definitions of j and J from the paragraph above. The highest integer such that $H_j \wedge x_j'$ is 4. The leftmost integer greater than j such that $H_J \in \mathrm{SAT}$ is 7. Hence, we have $j = 4$, $J = 7$.

Suppose that j doesn't exist for any $\langle F_1, \ldots, F_l \rangle$. Then $\chi_l^{\mathrm{SAT}}(\langle H_1, \ldots, H_l \rangle) = 00\ldots0$, and $\langle H_1, \ldots, H_l \rangle$ cannot be an advisor for any l-tuple of formulas. If it were, M wouldn't compute $\mathcal{Q}$ because the indicator that $\chi_l^{\mathrm{SAT}}(\langle H_1, \ldots, H_l \rangle) = 00\ldots0$ is the singleton set $\{00\ldots0\}$ and M has to be able to output the correct value of $\mathcal{Q}$. Hence, we can assume that j always exists.

First assume that J exists. Notice that $J \neq 1$ because that would make $i \leq 0$, which cannot happen.

Recall that we defined $s = \mathcal{Q}(\langle x_1' \wedge H_1, \ldots, x_l' \wedge H_l \rangle)$ for some k-tuple $\langle x_1', \ldots, x_l' \rangle$ and used the tuple $\langle x_1', \ldots, x_l' \rangle$ to define j and J. Since s is not in $\mathrm{OUT}(\mathrm{M}, \langle F_1' \wedge H_1, \ldots, F_l' \wedge H_l \rangle$, it must be the case that either $F_j \vee \cdots \vee F_{J-1} \notin \mathrm{SAT}$ or that $F_J \vee \cdots \vee F_l \in \mathrm{SAT}$. Assume that s does not appear in $\mathrm{OUT}(\mathrm{M}, \langle F_1' \wedge H_1, \ldots, F_l' \wedge H_l \rangle)$. Suppose that $F_j \vee \cdots \vee F_{J-1} \in \mathrm{SAT}$ and $F_J \vee \cdots \vee F_l \notin \mathrm{SAT}$. Then because $\langle F_1', \ldots, F_l' \rangle$ is nested, we must have $F_j' \in \mathrm{SAT}$. By definition of j and because we assume that all of $F_J, \ldots, F_l$ are unsatisfiable, $H_{j+1} \wedge F_{j+1}', \ldots, H_l \wedge F_l'$ are all unsatisfiable. Also, we have $H_i \wedge F_i' \in \mathrm{SAT} \iff H_i \wedge x_i' \in \mathrm{SAT}$ for all $i \in \{l-m+1, \ldots, l\}$ by the choice of j and J, and because $\langle x_1', \ldots, x_l' \rangle$ and $\langle F_1', \ldots, F_l' \rangle$ are nested. Then $\mathcal{Q}(\langle H_1 \wedge F_1', \ldots, H_l \wedge F_l' \rangle) = \mathcal{Q}(\langle H_1 \wedge x_1', \ldots, H_l \wedge x_l' \rangle) = s$. This is a contradiction because M computes $\mathcal{Q}$ and s is not in the output sequence of M on input $\langle H_1 \wedge F_1', \ldots, H_l \wedge F_l' \rangle$, so M can't output s. Hence, either $F_j \vee \cdots \vee F_{J-1} \notin \mathrm{SAT}$ or $F_J \vee \cdots \vee F_l \in \mathrm{SAT}$.

Therefore, the index of the rightmost satisfiable formula in $\langle F_1, \ldots, F_l \rangle$ is either less than j or at least J, which means that the membership of $\langle F_1, \ldots, F_l \rangle$ in BL_l does not depend on the satisfiability of all formulas in $\langle F_1, \ldots, F_l \rangle$. This is sufficient for a reduction from BL_l to coBL_l, which is what we show next.

Now we find a reduction for the general case when $j < l$ and when J exists and is greater than 1. Since either $F_j \vee \cdots \vee F_{J-1} \notin \mathrm{SAT}$ or $F_J \vee \cdots \vee F_l \in \mathrm{SAT}$ holds, it must be the case that $\langle F_1, \ldots, F_l \rangle \in \mathrm{BL}_l$ if and only if

$$\bigvee_{\substack{i \text{ odd} \\ i<j}} \left[(F_i \in \mathrm{SAT}) \wedge \Big(\bigwedge_{\substack{r \text{ even} \\ i<r\leq l}} (F_r \notin \mathrm{SAT}) \Big) \right] \tag{3}$$

or

$$\Big(\bigwedge_{\substack{r \text{ even} \\ J\leq r\leq l}} (F_r \notin \mathrm{SAT}) \Big) \vee \left(\bigvee_{\substack{i \text{ odd} \\ J+1<i\leq l}} \left[(F_i \in \mathrm{SAT}) \wedge \Big(\bigwedge_{\substack{r \text{ even} \\ i<r\leq l}} (F_r \notin \mathrm{SAT}) \Big) \right] \right). \tag{4}$$

Assume that j, J and l are all even. We will deal with all other possibilities later. We notice that both (3) and (4) hold only if $F_l \notin \mathrm{SAT}$. Then at least one of (3) and (4) holds only if $F_l \notin \mathrm{SAT}$ and if at least one of the two statements below is true.

$$\bigvee_{\substack{i \text{ odd} \\ i<j}} \Big[(F_i \in \mathrm{SAT}) \wedge \Big(\bigwedge_{\substack{r \text{ even} \\ i<r\leq l-2}} (F_r \notin \mathrm{SAT})\Big)\Big] \tag{5}$$

$$\Big(\bigwedge_{\substack{r \text{ even} \\ J\leq r\leq l-2}} (F_r \notin \mathrm{SAT})\Big) \vee \Big(\bigvee_{\substack{i \text{ odd} \\ J+1<i\leq l}} \Big[(F_i \in \mathrm{SAT}) \wedge \Big(\bigwedge_{\substack{r \text{ even} \\ i<r\leq l-2}} (F_r \notin \mathrm{SAT})\Big)\Big]\Big) \tag{6}$$

Notice that (6) is true if $F_{l-1} \in \mathrm{SAT}$. We can then say that at least one of (3) and (4) holds if $F_l \notin \mathrm{SAT}$ and if either (5) holds or the condition below holds.

$$(F_{l-1} \in \mathrm{SAT}) \vee \Big(\bigwedge_{\substack{r \text{ even} \\ J\leq r\leq l-2}} (F_r \notin \mathrm{SAT})\Big) \vee \Big(\bigvee_{\substack{i \text{ odd} \\ J+1<i\leq l-2}} \Big[(F_i \in \mathrm{SAT}) \wedge \Big(\bigwedge_{\substack{r \text{ even} \\ i<r\leq l-2}} (F_r \notin \mathrm{SAT})\Big)\Big]\Big) \tag{7}$$

Then $\langle F_1, \dots, F_l\rangle \in \mathrm{BL}_l$ if and only if $[F_l \notin \mathrm{SAT} \wedge (F_{l-1} \in \mathrm{SAT} \vee ((5) \text{ holds } \vee \ (7) \text{ holds}))]$. We can further expand the condition (5) holds $\vee$ (7) holds to conclude that $\langle F_1, \dots, F_l\rangle \in \mathrm{BL}_l$ if and only if $F_l \notin \mathrm{SAT} \wedge (F_{l-1} \in \mathrm{SAT} \vee \cdots \wedge (F_{J+3} \in \mathrm{SAT} \vee X) \cdots)$ where X is true if

$$\Big(\bigvee_{\substack{i \text{ odd} \\ i<j}} \Big[(F_i \in \mathrm{SAT}) \wedge \Big(\bigwedge_{\substack{r \text{ even} \\ i<r\leq J+2}} (F_r \notin \mathrm{SAT})\Big)\Big]\Big) \vee \Big(\bigwedge_{\substack{r \text{ even} \\ J\leq r\leq J+2}} (F_r \notin \mathrm{SAT})\Big). \tag{8}$$

We can further rewrite (8) as

$$\Big(\Big(\bigvee_{\substack{r \text{ even} \\ j\leq r\leq J+2}} F_r\Big) \notin \mathrm{SAT}\Big) \wedge \Big(F_{j-1} \in \mathrm{SAT} \vee \Big[F_{j-2} \notin \mathrm{SAT} \wedge \cdots \wedge ((F_1 \in \mathrm{SAT}) \vee (\mathrm{TRUE} \notin \mathrm{SAT})) \cdots \Big]\Big) \tag{9}$$

Putting this all together, we get $\langle F_1, \dots, F_l\rangle \in \mathrm{BL}_l$ if and only if

$$F_l \notin \mathrm{SAT} \wedge (F_{l-1} \in \mathrm{SAT} \vee \cdots \wedge (F_{J+3} \in \mathrm{SAT} \vee (9) \text{ holds}) \cdots).$$

This is a statement about membership in coBL_l. Hence, we have a reduction from BL_l to coBL_l. We can make a similar argument for other combinations of parities of j, J and l, and also when J doesn't exist at all.

We have $l > 2^k - 2^m + 1$. A $\mathrm{PF}_m^{\mathrm{NP}[k]}$ machine M has 2^k computation paths, which means that its output sequence has 2^k terms. If the output sequence of M contained indicators of all possible values of $\chi_l^{\mathrm{SAT}}(\langle H_1, \dots, H_l\rangle)$ on input $\langle H_1 \wedge x'_1, \dots, H_l \wedge x'_l\rangle$, it would have to output all strings of length m because $\mathcal{Q}$ is onto and every bit string of length m belongs to at least one indicator (take $\langle x'_1, \dots, x'_l\rangle = \langle \mathrm{TRUE}, \dots, \mathrm{TRUE}\rangle$; then $\chi_l^{\mathrm{SAT}}(\langle H_1 \wedge x'_1, \dots, H_l \wedge x'_l\rangle) = \chi_l^{\mathrm{SAT}}(\langle H_1, \dots, H_l\rangle)$). Its output sequence would have to have at least $l + 1 + 2^m - 2 > 2^k - 2^m + 1 + 1 + 2^m - 2 = 2^k$ terms, which can't happen. The $l + 1$ comes from the chain of l mind changes indicating $\chi_l^{\mathrm{SAT}}(\langle H_1, \dots, H_l\rangle) = 11\dots1$, the remaining $2^m - 2$ are the remaining bit strings of length m that are not present in the indicator of $11\dots1$. This implies that the indicator of at least one value of χ_l^{SAT} won't be present in its entirety in the output sequence of M on any input.

We have to modify the construction of the advice. We now put an l-tuple of boolean formulas from T_i (the set of potential advisors that have not been put in the advice yet) into the advice if T_i is an advisor for at least $2^{-l-2^{l+1}}\|S_i\|$ elements of S_i (recall that S_i is the set of l-tuples of boolean formulas for which we have not found an advisor yet during the construction of the advice). When advice construction terminates in case 2, all elements of T_i will advise less than $2^{-l-2^{l+1}}\|S_i\|$ l-tuples in S_i.

When advice construciton terminates in case 1, we get a reduction using the argument presented earlier.

Suppose advice construction terminates in case 2 and that we fix an l-tuple $\langle H_1, \ldots, H_l \rangle$ of boolean formulas in T_i. We can find the probability that at least $\frac{1}{2(2^l-1)}q(n)$ out of $q(n)$ randomly picked l-tuples from S_i are advisees of $\langle H_1, \ldots, H_l \rangle$. This also gives us an upper bound on the probability that at least $q(n)/2^l$ out of $q(n)$ randomly picked l-tuples from S_i are advisees of $\langle H_1, \ldots, H_l \rangle$. Also suppose that $p_l(n)$ is the maximum length of an OR of l boolean formulas of length n. We can pick $q(n)$ so that the probabilistic argument in the proof of Theorem 4 still goes through, i.e., we can find a sequence of $q(n)$ l-tuples from S_i such that if $\langle H_1, \ldots, H_l \rangle \notin T_i$, $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge x'_1, \ldots, H_l \wedge x'_l \rangle)$ doesn't contain the indicator of the correct value of $\chi_l^{\mathrm{SAT}}(\langle H_1, \ldots, H_l \rangle)$ for less than $q(n)/2^l$ of those l-tuples $\langle x_1, \ldots, x_l \rangle$. This will ensure that for at least one of the incorrect values of $\chi_l^{\mathrm{SAT}}(\langle H_1, \ldots, H_l \rangle)$, the indicator of this value will not be present in $\mathrm{OUT}(\mathrm{M}, \langle H_1 \wedge x'_1, \ldots, H_l \wedge x'_l \rangle)$ in at least $q(n)/2^l$ cases. We will pick this indicated value as an incorrect value of $\chi_l^{\mathrm{SAT}}(\langle H_1, \ldots, H_l \rangle)$, and get a polynomial time algorithm for satisfiability.

Now we have all the machinery that's necessary to complete the proof of Theorem 8. The algorithm that gives a reduction resembles the one given for the special case $\mathrm{PF}_3^{\mathrm{NP}\|[4]} \subseteq \mathrm{PF}_3^{\mathrm{NP}[3]}$ presented in the previous section. It uses a generalized version of $\mathcal{Q}_{43}$, $\mathcal{Q}$, a generalized Lemma 9 in place of Lemma 2, and a different definition of advisees, where both advisors and advisees are l-tuples of boolean formulas. Advice construction is the same, only the fraction of S_i that should be advised by a formula that's being added to the advice is different. If an l-tuple $\langle F_1, \ldots, F_l \rangle$ is an advisee of some l-tuple $\langle H_1, \ldots, H_l \rangle$ in the advice string, we have a general method to reduce BL_l to coBL_l. If advice construction terminates in case 2, we will have a way of eliminating one possibility for $\chi_l^{\mathrm{SAT}}(\langle H_1, \ldots, H_l \rangle)$ in polynomial time. This will give us a way to decide satisfiability in polynomial time, which is sufficient for a reduction from BL_l to coBL_l. □

5 Summary

We have resolved two open questions posed by Chang and Squire [7]. We showed as Theorems 4 and 6 that both $\mathrm{PF}_3^{\mathrm{NP}\|[4]} \subseteq \mathrm{PF}_3^{\mathrm{NP}[3]}$ and $\mathrm{PF}_2^{\mathrm{NP}\|[5]} \subseteq \mathrm{PF}_2^{\mathrm{NP}[3]}$ imply a collapse of the Polynomial Hierarchy to Σ_3^{P}. We also proved a more general theorem of which $\mathrm{PF}_3^{\mathrm{NP}\|[4]} \subseteq \mathrm{PF}_3^{\mathrm{NP}[3]} \Longrightarrow \mathrm{PH} = \Sigma_3^{\mathrm{P}}$ is a special case.

There are still some unknown cases, however. To see that, we make the following observation.

Observation 10 $\mathrm{PF}_m^{\mathrm{NP}\|[l]} \subseteq \mathrm{PF}_m^{\mathrm{NP}[k]}$ *whenever* $k \geq \lceil \log(l+1) \rceil + m$.

Proof Sketch: Suppose a $\mathrm{PF}_m^{\mathrm{NP}\|[l]}$ machine M computes a function f. Let x be an input to M. Using binary search, it takes $\lceil \log(l+1) \rceil$ queries to an NP oracle to find out how many of the queries are answered "yes" by M's oracle. Now we can use the census trick to find any one bit of the output made by M on input x using one oracle query. Therefore, if we make m more queries, we can find out what each of the m bits output by M on input x is. □

Observation 10 implies that if $l < 2^{k-m}$, $\mathrm{PF}_m^{\mathrm{NP}\|[l]} \subseteq \mathrm{PF}_m^{\mathrm{NP}[k]}$. We also know from our generalization that if $l > 2^k - 2^m + 1$ (and all the other conditions), $\mathrm{PH} = \Sigma_3^{\mathrm{P}}$. Notice that if $k, m \geq 2$, there is a gap between 2^{k-m} and $2^k - 2^m + 1$. In other words, if $k, m \geq 2$, there exists an integer between 2^{k-m} and $2^k - 2^m + 1$. Table 3 demonstrates this claim for some pairs of $k, m \geq 2$. It is not hard to see that the claim holds for all $k, m \geq 2$. We do not know what the consequences are when $\mathrm{PF}_m^{\mathrm{NP}\|[l]} \subseteq \mathrm{PF}_m^{\mathrm{NP}[k]}$ and $2^{k-m} \leq l \leq 2^k - 2^m + 1$.

We have shown a collapse of the Polynomial Hierarchy for one case in this gap by showing that $\mathrm{PF}_2^{\mathrm{NP}\|[5]} \subseteq \mathrm{PF}_2^{\mathrm{NP}[3]} \Longrightarrow \mathrm{PH} = \Sigma_3^{\mathrm{P}}$ as Theorem 6. The remaining cases in this gap remain open, however. A new technique or perhaps a modification of the current technique is necessary to generalize the result proved in Theorem 6.

$m\backslash k$	3	4	5	6	7	8	9
2	1	3	7	15	31	63	127
3		1	3	7	15	31	63
4			1	3	7	15	31
5				1	3	7	15
6					1	3	7
7						1	3

(a) Highest values of l that satisfy $l < 2^{k-m}$

$m\backslash k$	3	4	5	6	7	8	9
2	6	14	30	62	126	254	510
3		10	26	58	122	250	506
4			18	50	114	242	498
5				34	98	226	482
6					66	194	450
7						130	386

(b) Smallest values of l that satisfy $l > 2^k - 2^m + 1$

Table 3: We see that for all pairs of $k, m \geq 2$ such that $k > m$ shown in this table, the difference between the largest value of l such that $l < 2^{k-m}$ and the smallest value of l such that $l > 2^k - 2^m + 1$ is at least 2.

Acknowledgements

I would like to thank Dr. Richard Chang for supervising my work on this paper, for discussing it with me, for explaining some of the background to me, for reviewing this paper, and for his support in general. Our discussions gave me many valuable insights that helped me when I was writing this paper.

References

[1] Amihood Amir, Richard Beigel, and William 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, Washington, DC, USA, 1990. IEEE Computer Society.

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

[3] Richard Beigel, Martin Kummer, and Frank Stephan. Approximable sets. *Information and Computation*, 120(2):304–314, 1995.

[4] Jin-Yi Cai, Thomas Gundermann, Juris Hartmanis, Lane A. Hemachandra, Vivian Sewelson, Klaus Wagner, and Gerd Wechsung. The boolean hierarchy I: Structural properties. *Society for Industrial and Applied Mathematics Journal on Computing*, 17(6):1232–1252, 1988.

[5] Jin-Yi Cai, Thomas Gundermann, Juris Hartmanis, Lane A. Hemachandra, Vivian Sewelson, Klaus Wagner, and Gerd Wechsung. The boolean hierarchy II: Applications. *Society for Industrial and Applied Mathematics Journal on Computing*, 18(1):95–111, 1989.

[6] Richard Chang and Jim Kadin. The boolean hierarchy and the polynomial hierarchy: A closer connection. *SIAM Journal on Computing*, 25(2):340–354, 1996.

[7] Richard Chang and Jon Squire. Bounded query functions with limited output bits. In *CCC '01: Proceedings of the 16th Annual Conference on Computational Complexity*, page 90, Washington, DC, USA, 2001. IEEE Computer Society.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2001.

[9] Lane Hemaspaandra and Mitsunori Ogihara. *Complexity Theory Companion*. Springer, 2002.

[10] Jim Kadin. The polynomial time hierarchy collapses if the boolean hierarchy collapses. *Society for Industrial and Applied Mathematics Journal on Computing*, 17(6):1263–1282, 1988.

[11] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

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