

The 1-Versus-2 Queries Problem Revisited*

Rahul Tripathi

Department of Computer Science and Engineering, University of South Florida,
Tampa, FL 33620, USA (Email: tripathi@cse.usf.edu)

Abstract. The *1-versus-2 queries* problem, which has been extensively studied in computational complexity theory, asks in its generality whether every efficient algorithm that makes at most 2 queries to a Σ_k^p -complete set L_k has an efficient simulation that makes at most 1 query to L_k . We obtain solutions to this problem under hypotheses weaker than previously considered. We prove that:

1. For each $k \geq 2$, $P_{tt}^{\Sigma_k^p[2]} \subseteq ZPP^{\Sigma_k^p[1]} \implies PH = \Sigma_k^p$, and
2. $P_{tt}^{NP[2]} \subseteq ZPP^{NP[1]} \implies PH = S_2^p$.

Here, for a complexity class \mathcal{C} and integer $j \geq 1$, we model $ZPP^{\mathcal{C}[j]}$ to be the class of problems solvable by zero-error randomized algorithms that always run in polynomial time, make at most j queries to \mathcal{C} , and succeed with probability only $1/2 + 1/\text{poly}(\cdot)$. This same model of $ZPP^{\mathcal{C}[j]}$, also considered in [CC06], subsumes the class of problems solvable by randomized algorithms that always answer correctly in expected polynomial time and make at most j queries to \mathcal{C} .

Hemaspaandra, Hemaspaandra, and Hempel [HHH98], for $k > 2$, and Buhrman and Fortnow [BF99], for $k = 2$, had obtained the same consequence as of ours in (1) using the stronger hypothesis $P_{tt}^{\Sigma_k^p[2]} \subseteq P^{\Sigma_k^p[1]}$. Fortnow, Pavan, and Sengupta [FPS] had obtained the same consequence as of ours in (2) using the stronger hypothesis $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$.

Our results may also be viewed as steps towards obtaining solutions to the most general form of the 1-versus-2 queries problem: For any $k \geq 1$, whether $P_{tt}^{\Sigma_k^p[2]}$ can be simulated in $BPP^{\Sigma_k^p[1]}$.

1 Introduction

1.1 Background

Krentel [Kre88] studied the functional version of the 1-versus-2 queries problem. Krentel proved that if every deterministic polynomial-time computable function that makes at most two queries to SAT has a deterministic polynomial-time simulation that makes at most one query to SAT, then $P = NP$. That is, if $FP^{NP[2]} \subseteq FP^{NP[1]}$, then $P = NP$.

The decision version of the 1-versus-2 queries problem has been deemed to be more difficult than its functional counterpart. A long succession of work on

* Research supported by the New Researcher Grant of the University of South Florida.

the decision version of the 1-versus-2 queries problem is known in the literature. We review progress made in these work.

Kadin [Kad88] proved that if $P^{NP[2]} \subseteq P^{NP[1]}$, then $NP \subseteq coNP/poly$. Yap [Yap83] showed that the polynomial hierarchy collapses to the third level if $NP \subseteq coNP/poly$, and Cai et al. [CCHO05] improved this collapse of the polynomial hierarchy to S_2^{NP} under the same assumption, i.e., the assumption $NP \subseteq coNP/poly$. Thus, Kadin's result implies that if $P^{NP[2]} \subseteq P^{NP[1]}$, then $PH = S_2^{NP}$. Wagner [Wag89] improved Kadin's result and showed that the polynomial hierarchy collapses to $P^{\Sigma_2^p}$ under the assumption $P^{NP[2]} \subseteq P^{NP[1]}$. Beigel, Chang, and Ogiwara [BCO93] built upon the work of Wagner [Wag89] and Chang and Kadin [CK96] and made further improvements to the solution of $P^{NP[2]} \subseteq P^{NP[1]}$. They showed that if $P^{NP[2]} \subseteq P^{NP[1]}$, then every set in the polynomial hierarchy can be solved by a deterministic polynomial-time Turing machine that makes at most one query to an NP oracle and at most one query to a Σ_2^p oracle. Since it is known that $P^{NP[2]} \subseteq P^{NP[1]}$ if $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$ [CK95], all these results hold assuming the seemingly weaker hypothesis $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$.

Hemaspaandra, Hemaspaandra, and Hempel [HHH98] studied the 1-versus-2 queries problem in a general context. They showed that for $k > 2$, if $P_{tt}^{\Sigma_k^p[2]} \subseteq P^{\Sigma_k^p[1]}$, then $PH = \Sigma_k^p$. They extended this result for the 1-versus-2 queries problem to the result for an even more general problem: the m -versus- $(m+1)$ queries problem. In particular, they showed that for each $m > 0$ and each $k > 2$, if every deterministic polynomial-time Turing machine that makes at most $m+1$ truth-table queries to a Σ_k^p -complete set L_k has a deterministic polynomial-time simulation that makes at most m truth-table queries to L_k , then the boolean hierarchy over Σ_k^p collapses to the m 'th level. That is, they showed that for each $m > 0$ and each $k > 2$, if $P_{tt}^{\Sigma_k^p[m+1]} \subseteq P_{tt}^{\Sigma_k^p[m]}$, then $DIFF_m(\Sigma_k^p) = coDIFF_m(\Sigma_k^p) = BH(\Sigma_k^p)$. Beigel, Chang, and Ogiwara [BCO93] related the collapse of the boolean hierarchy to the collapse of the polynomial hierarchy. Thus, as a consequence of this relationship between the two hierarchies, the result of Hemaspaandra, Hemaspaandra, and Hempel [HHH98] also implies that for each $m > 0$ and each $k > 2$, if $P_{tt}^{\Sigma_k^p[m+1]} \subseteq P_{tt}^{\Sigma_k^p[m]}$, then the polynomial hierarchy can be solved by a deterministic polynomial-time Turing machine that makes $m-1$ truth-table queries to Σ_{k+1}^p and unbounded queries (strictly speaking, polynomially many queries since a deterministic polynomial-time Turing machine cannot make more than a polynomial number of queries) to Σ_k^p .

Hemaspaandra, Hemaspaandra, and Hempel [HHH98] left open the case $k = 2$ in their solution to the 1-versus-2 queries problem, which was subsequently settled by Buhrman and Fortnow [BF99]. Buhrman and Fortnow [BF99]

showed that if $P_{tt}^{\Sigma_2^p[2]} \subseteq P^{\Sigma_2^p[1]}$, then $PH = \Sigma_2^p$. They also showed that no relativizable proof technique can establish a similar result for NP. That is, they showed that the result “if $P_{tt}^{\text{NP}[2]} \subseteq P^{\text{NP}[1]}$, then $PH = \text{NP}$ ” cannot be proved using relativizable proof techniques. In spite of this negative result, they managed to obtain several other consequences, though weaker than the much sought after consequence $PH = \text{NP}$, of the $P_{tt}^{\text{NP}[2]} \subseteq P^{\text{NP}[1]}$ hypothesis including (a) locally either $\text{NP} = \text{coNP}$ or NP has polynomial-size circuits, (b) $PH = \text{BPP}^{\text{NP}[1]}$, (c) $\Sigma_2^p \subseteq \Pi_2^p/1$, (d) $\Sigma_2^p = \text{UP}^{\text{NP}[1]} \cap \text{RP}^{\text{NP}[1]}$, and (e) $P^{\text{NP}} = P^{\text{NP}[1]}$.

In another paper, for the case $k = 2$, Hemaspaandra, Hemaspaandra, and Hempel [HHH05] extended the solution of the 1-versus-2-queries problem by Buhrman and Fortnow [BF99] to the solution of the m -versus- $(m + 1)$ queries problem. Hemaspaandra, Hemaspaandra, and Hempel showed that even for the case $k = 2$ and for all $m > 0$, if $P_{tt}^{\Sigma_k^p[m+1]} \subseteq P_{tt}^{\Sigma_k^p[m]}$, then $\text{DIFF}_m(\Sigma_k^p) = \text{coDIFF}_m(\Sigma_k^p) = \text{BH}(\Sigma_k^p)$. We mention here that for the case $k = 1$ and for any $m > 1$, no solution is known for the following version of the m -versus- $(m + 1)$ queries problem: whether $P_{tt}^{\Sigma_k^p[m+1]}$ has a simulation in $P_{tt}^{\Sigma_k^p[m]}$.

Fortnow, Pavan, and Sengupta [FPS] improved upon the result of Buhrman and Fortnow [BF99] to show that if $P_{tt}^{\text{NP}[2]} \subseteq P^{\text{NP}[1]}$, then $PH = S_2^p$, where S_2^p satisfies $S_2^p \subseteq \text{ZPP}^{\text{NP}}$ ([Cai07]) $\subseteq \Sigma_2^p \cap \Pi_2^p$. This was the first solution to the 1-versus-2 queries problem for the case $k = 1$ (i.e., NP oracle) that achieved a collapse of the polynomial hierarchy to a level not above Σ_2^p . A major ingredient in their proof was a lemma, whose proof ideas were inspired from the paper of Bshouty et al. [BCG⁺96], that states that for every $n > 0$ and every $k > 0$, if SAT has no circuit of size n^{k+2} at length n , then there exists a polynomial-size collection S of satisfiable formulae of length n such that every circuit of size n^k fails to produce any satisfying assignment for at least one formula from S . This lemma has found applications elsewhere (see [PSV06]).

Recently, Chakaravarthy and Roy [CR06] introduced new classes O_2^p , YO_2^p , and NO_2^p as subclasses of S_2^p . These classes were introduced with the motivation of improving several existing complexity results such as the classical Karp-Lipton theorem [KL80] and a theorem of Yap [Yap83]. Chakaravarthy and Roy [CR06] showed that these new classes have implication also on the solution to the 1-versus-2-queries problem. They proved that if $P_{tt}^{\text{NP}[2]} \subseteq P^{\text{NP}[1]}$, then $PH = \text{NO}_2^p \cap \text{YO}_2^p$. This gives a slight improvement over the solution given by Fortnow, Pavan, and Sengupta [FPS].

More recently, Chang and Purini [CP07] proved that if the NP *machine hypothesis* holds, then $P_{tt}^{\text{NP}[2]} \subseteq P^{\text{NP}[1]}$ implies $PH = \text{NP}$. The NP machine hypothesis postulates that there exist an $\epsilon > 0$ and a nondeterministic polynomial-

time Turing machine N such that $L(N) = 0^*$ and for any 2^{n^ϵ} -time bounded deterministic Turing machine M , $M(0^n)$ produces an accepting path of $N(0^n)$ only for finitely many n .

1.2 Our Results

In this paper, we obtain solutions to the 1-versus-2 queries problem under the hypotheses $P_{tt}^{\Sigma_k^p[2]} \subseteq ZPP^{\Sigma_k^p[1]}$, for integers $k \geq 1$. Note that $P^{\Sigma_k^p[1]}$ is a subclass of $ZPP^{\Sigma_k^p[1]}$, and so the hypotheses we consider here, namely, $P_{tt}^{\Sigma_k^p[2]} \subseteq ZPP^{\Sigma_k^p[1]}$, are weaker than the previously considered hypotheses (see the papers [Kad88,Wag89,BCO93,HHH98,BF99,FPS,CR06]), namely, $P_{tt}^{\Sigma_k^p[2]} \subseteq P^{\Sigma_k^p[1]}$. Our results may also be looked upon as steps towards obtaining solutions to the general form of the 1-versus-2 queries problem, which can be stated as whether every $P_{tt}^{\Sigma_k^p[2]}$ algorithm has a $BPP^{\Sigma_k^p[1]}$ simulation.

The classes $ZPP^{\Sigma_k^p[1]}$, considered in the hypotheses of the statements of our results, have been recently used in some inclusion relationships between complexity classes. Cai [Cai07] proved that $S_2^p \subseteq ZPP^{NP}$; however, the question of whether this inclusion is also an equality is open. Cai and Chakaravarthy [CC06] showed that if one restricts a ZPP algorithm so that it is allowed to make at most one query to an NP oracle, then such an algorithm can be simulated in S_2^p . In other words, they proved that $ZPP^{NP[1]} \subseteq S_2^p$. For $k \geq 2$, they proved that $ZPP^{\Sigma_k^p[1]} \subseteq P^{\Sigma_k^p[2]}$. They observed that $BPP \subseteq ZPP^{NP[1]}$ (also proven implicitly in some other papers) and used this observation to infer that it is unlikely to prove $ZPP^{NP[1]} \subseteq P^{NP}$ unconditionally since otherwise it will yield an unconditional containment of BPP in P^{NP} , which has been open for a long time.

In all these results, the success probability of $ZPP^{\mathcal{C}[1]}$ algorithms, for an arbitrary complexity class \mathcal{C} , is considered to be only $1/2 + 1/\text{poly}(\cdot)$.

2 Preliminaries

Our alphabet is $\Sigma = \{0, 1\}$. We assume familiarity with basic notions in computational complexity theory such as complexity classes (P , NP , ZPP , BPP , Σ_k^p , Π_k^p , PH , etc.), decision problems (SAT, Σ_k^p -complete sets), oracles, reductions (\leq_m^p , \leq_T^p , \leq_{tt}^p), and Turing machines (deterministic, randomized). Let $\langle \cdot, \cdot \rangle$ be a multi-arity, polynomial-time computable, and polynomial-time invertible pairing function. DPTM (DPOTM) will stand for “deterministic polynomial-time (oracle) Turing machine,” and RPTM (RPOTM) will stand for “randomized polynomial-time (oracle) Turing machine.” For a deterministic Turing machine M and string $x \in \Sigma^*$, we use $M(x)$ to denote the computation

of M on input x . Similarly, for a randomized Turing machine M and strings $x, r \in \Sigma^*$, we use $M(x; r)$ to denote the computation of M on input x and random string r . At few places, we abuse notation for the sake of brevity and let $M(x) \in \{\text{Accept}, \text{Reject}\}$ also denote the outcome of a deterministic Turing machine M on input x and let $M(x; r) \in \{\text{Accept}, \text{Reject}, ?\}$ also denote the outcome of a randomized Turing machine M on input x and random string r (which sense is being used for $M(x)$ or $M(x; r)$ will be clear from the context).

We will need the following definition:

Definition 1. 1. [HHH98] For any oracles C and D , let $M^{(C,D)}$ denote a DPOTM M making at most one query to C and at most one query to D in a truth-table fashion, i.e., in parallel. Then, for complexity classes \mathcal{C} and \mathcal{D} ,

$$\mathcal{P}^{(\mathcal{C}, \mathcal{D})} =_{df} \{L \subseteq \Sigma^* \mid (\exists C \in \mathcal{C})(\exists D \in \mathcal{D})(\exists \text{ DPOTM } M)[L = L(M^{(C,D)})]\}.$$

2. For any oracle C and integer $j \geq 0$, let $M^{C[j]}$ denote a DPOTM M making at most j adaptive, i.e., sequential, queries to C . Then, for a complexity class \mathcal{C} ,

$$\mathcal{P}^{C[j]} =_{df} \{L \subseteq \Sigma^* \mid (\exists C \in \mathcal{C})(\exists \text{ DPOTM } M)[L = L(M^{C[j]})]\}.$$

In this paper, we will consider zero-error randomized polynomial-time algorithms that can make at most j queries, for some $j \geq 1$, to an oracle C . The class of decision problems solvable by such algorithms is denoted by $\text{ZPP}^{C[j]}$, where C is the oracle and j is a bound on the number of queries to C made by the algorithm.

Some subtlety is involved when we talk about the success probability of a bounded query randomized algorithm (e.g., a $\text{ZPP}^{C[j]}$ algorithm). In particular, while the success probability of a ZPP algorithm can be amplified from $1/n^{O(1)}$ to $1/2 + 1/n^{O(1)}$ using a standard technique, which involves running the algorithm on several independently chosen random strings and making a decision based on the outcome of the runs, the same technique does not work for the case of $\text{ZPP}^{C[j]}$ algorithms since an application of the same technique could result in an algorithm that asks more than the allowed number (j) of queries to C . Hence, we need to fix our assumption regarding the success probability of $\text{ZPP}^{C[j]}$ algorithms (because different bounds on the success probability of $\text{ZPP}^{C[j]}$ algorithms could define different complexity classes).

Throughout this paper, we require the success probability of $\text{ZPP}^{C[j]}$ algorithms to be only $1/2 + 1/\text{poly}(\cdot)$. Cai and Chakaravarthy [CC06] imposed the same requirement on the success probability of $\text{ZPP}^{C[j]}$ algorithms in proving their results $\text{ZPP}^{\text{NP}[1]} \subseteq \text{S}_2^p$ and $\text{ZPP}^{\Sigma_k^p[1]} \subseteq \text{P}^{\Sigma_k^p[2]}$ for integers $k \geq 2$.

We next formally define $\text{ZPP}^{C[j]}$ for any arbitrary complexity class \mathcal{C} and integer $j \geq 1$.

Definition 2 ([CC06]). Let \mathcal{C} be a complexity class and $j \geq 1$ be an integer. A set $L \in \text{ZPP}^{\mathcal{C}[j]}$ if there exist a RPOTM $M(\cdot; \cdot)$, an oracle $C \in \mathcal{C}$, and a polynomial $p(\cdot)$ such that M satisfies the following requirements for any input $x \in \Sigma^*$:

1. On any random string r , $M(x; r)$ makes at most j adaptive (i.e., sequential) queries to the oracle C .
2. For any random string r , the output $M^{C[j]}(x; r)$ belongs to $\{\text{Accept}, \text{Reject}, ?\}$ such that (a) if $x \in L$, then $M^{C[j]}(x; r) \in \{\text{Accept}, ?\}$ and (b) if $x \notin L$, then $M^{C[j]}(x; r) \in \{\text{Reject}, ?\}$. That is, the machine M has zero-error.
3. M succeeds with probability at least $1/2 + 1/p(\cdot)$. That is,

$$\text{Prob}_r \left[M^{C[j]}(x; r) = \text{Accept or } M^{C[j]}(x; r) = \text{Reject} \right] \geq \frac{1}{2} + \frac{1}{p(|x|)}.$$

Remarks on the robustness of the class $\text{ZPP}^{\mathcal{C}[j]}$ defined in Definition 2. The class ZPP has two equivalent definitions:

1. The class of problems solvable by randomized algorithms that always answer correctly and run in expected polynomial time.
2. The class of problems solvable by zero-error randomized algorithms that always run in polynomial time and succeed with probability at least $1/\text{poly}(\cdot)$ (i.e., answer correctly on at least $1/\text{poly}(\cdot)$ fraction of time but may say “I don’t know” on the remaining fraction of time).

These equivalences are not known to hold for $\text{ZPP}^{\mathcal{C}[j]}$. Our definition of $\text{ZPP}^{\mathcal{C}[j]}$ in Definition 2 requires the success probability to be at least $1/2 + 1/\text{poly}(\cdot)$ (instead of $1/\text{poly}(\cdot)$), and thus may partially capture the interpretation of $\text{ZPP}^{\mathcal{C}[j]}$ as in (2). However, we mention that our definition of $\text{ZPP}^{\mathcal{C}[j]}$ does indeed fully capture the interpretation of $\text{ZPP}^{\mathcal{C}[j]}$ as in (1), and so in this sense our definition of $\text{ZPP}^{\mathcal{C}[j]}$ is robust. To see this, let P be a problem solvable by a randomized algorithm M that always answers correctly in expected polynomial time $p(\cdot)$ and makes at most j queries to C . Consider a randomized algorithm M' that on any input x , simulates $M(x)$ for $4p(|x|)$ steps, outputs according to M if $M(x)$ halts within $4p(|x|)$ steps, and says “I don’t know” if $M(x)$ does not halt within $4p(|x|)$ steps. By Markov’s inequality, it follows that M' is a j -query zero-error randomized algorithm for P , always run in time $4p(\cdot)$, and succeeds with probability at least $3/4$. Thus, P also belongs to the class $\text{ZPP}^{\mathcal{C}[j]}$ defined in Definition 2.

The class S_2^p was introduced independently by Russell and Sundaram [RS98] and by Canetti [Can96]. A formal definition of S_2^p is as follows:

Definition 3 ([Can96,RS98]). S_2^p is the class of all sets L for which there exist a polynomial-time predicate $R(\cdot, \cdot, \cdot)$ and a polynomial $p(\cdot)$ such that for all $x \in \Sigma^*$,

$$\begin{aligned} x \in L &\implies (\exists^p y)(\forall^p z)R(x, y, z), \text{ and} \\ x \notin L &\implies (\exists^p z)(\forall^p y)\neg R(x, y, z), \end{aligned}$$

where for any w , $(\exists^p w) =_{df} (\exists w : |w| \leq p(|x|))$ and $(\forall^p w) =_{df} (\forall w : |w| \leq p(|x|))$.

For a complexity class \mathcal{C} , the complexity class \mathcal{C}/poly is defined as follows:

Definition 4 ([KL80]). For any complexity class \mathcal{C} and function $f : \mathbb{N} \rightarrow \mathbb{N}$, \mathcal{C}/f denotes the class of all sets L such that for some $C \in \mathcal{C}$ and for some arbitrary function $h : \mathbb{N} \rightarrow \Sigma^*$ satisfying $(\forall n)[|h(n)| = f(n)]$, it holds that for all $x \in \Sigma^*$, $x \in L \iff \langle x, h(|x|) \rangle \in C$.

A set L is said to be in \mathcal{C}/poly if and only if $L \in \mathcal{C}/p(n)$ for some polynomial p .

An important structural property of SAT is its 2-query disjunctive self-reducibility. This means that for any boolean formula $\phi(x_1, x_2, \dots, x_n)$ of n variables, $\phi \in \text{SAT}$ if and only if $\phi(x_1 := \text{true}, x_2, \dots, x_n) \in \text{SAT}$ or $\phi(x_1 := \text{false}, x_2, \dots, x_n) \in \text{SAT}$.

Due to the space limit, all proofs are omitted. They will appear in the full version of the paper.

3 Results

Hemaspaandra, Hemaspaandra, and Hempel [HHH98] showed that for any $k > 2$, if $P_{tt}^{\Sigma_k^p[2]} \subseteq P^{\Sigma_k^p[1]}$, then $\text{PH} = \Sigma_k^p$. Buhrman and Fortnow [BF99] showed that this result extends also for the case $k = 2$: If $P_{tt}^{\Sigma_2^p[2]} \subseteq P^{\Sigma_2^p[1]}$, then $\text{PH} = \Sigma_2^p$. In Theorem 5, we derive the same consequences under hypotheses weaker than the ones considered in the above two results.

Assuming the hypothesis $P_{tt}^{\Sigma_k^p[2]} \subseteq ZPP^{\Sigma_k^p[1]}$, for $k \geq 2$, we show that for a Σ_k^p -complete set L_k , its complement $\overline{L_k}$ belongs to Σ_k^p . This will prove that if $P_{tt}^{\Sigma_k^p[2]} \subseteq ZPP^{\Sigma_k^p[1]}$, then $\text{PH}_k^p \subseteq \Sigma_k^p$ and hence the polynomial hierarchy collapses to Σ_k^p .

Theorem 5. For any integer $k \geq 2$,

$$P^{(\text{NP}, \Sigma_k^p)} \subseteq ZPP^{\Sigma_k^p[1]} \implies \text{PH} = \Sigma_k^p.$$

We present the proof idea of Theorem 5. Fix an integer $k \geq 2$. Let L_k be a Σ_k^p -complete set. We define a set $D =_{df} L_k \times \overline{\text{SAT}} \cup \overline{L_k} \times \text{SAT} = \{\langle x, \psi \rangle \mid (x \in L_k \text{ and } \psi \notin \text{SAT}) \text{ or } (x \notin L_k \text{ and } \psi \in \text{SAT})\}$. Clearly, $D \in \text{P}^{(\text{NP}, \Sigma_k^p)}$, and hence by the hypothesis $\text{P}^{(\text{NP}, \Sigma_k^p)} \subseteq \text{ZPP}^{\Sigma_k^p[1]}$, we have $D \in \text{ZPP}^{\Sigma_k^p[1]}$ via a $\text{ZPP}^{\Sigma_k^p[1]}$ machine $N^{L_k[1]}$ such that $D = L(N^{L_k[1]})$. For the sake of clarity, in the discussion that follows we omit referring to the polynomial-length bounds on quantifiers (\exists, \forall).

Because the base computation of $N^{L_k[1]}$ is a ZPP computation, we can easily describe a Σ_k^p -procedure to determine $x \notin L_k$ for an input string x , if there exist a formula ψ and a random string r such that $N^{L_k[1]}(\langle x, \psi \rangle; r)$ has a definite outcome, i.e., the outcome is either Accept or Reject but not ?, and $N^{L_k[1]}(\langle x, \psi \rangle; r)$ makes a query $\tau \in L_k$. We call such input strings x *nice*. For nice x , this Σ_k^p -procedure will include (1) a guess for the formula ψ , (2) a guess for the random string r , (3) a Σ_k^p -procedure for $\tau \in L_k$, and (4) a Σ_2^p -procedure for even k and a Π_2^p -procedure for odd k that depend on whether the outcome $N^{L_k[1]}(\langle x, \psi \rangle; r)$ is Accept or Reject, and whether ψ is in SAT or $\overline{\text{SAT}}$.

The difficult part of the proof is when the input string x is not nice. In that case, we notice that for every formula ψ and for every random string r such that $N^{L_k[1]}(\langle x, \psi \rangle; r)$ reaches a definite outcome, i.e., the outcome $\in \{\text{Accept}, \text{Reject}\}$, the query τ made by $N^{L_k[1]}(\langle x, \psi \rangle; r)$ to L_k must be answered “no.” Since the base computation of $N^{L_k[1]}$ is a ZPP computation, therefore, for every formula ψ , the fraction of random strings r for which $N^{L_k[1]}(\langle x, \psi \rangle; r)$ reaches a definite outcome is *large* (actually, this fraction is at least $1/2 + 1/\text{poly}(|\langle x, \psi \rangle|)$). Thus, it follows that if x is not nice, then for every formula ψ , there is a large fraction of random strings r such that (1) $N^{L_k[1]}(\langle x, \psi \rangle; r)$ reaches a definite outcome $\in \{\text{Accept}, \text{Reject}\}$ and (2) the query τ made by $N^{L_k[1]}(\langle x, \psi \rangle; r)$ to L_k is answered “no.” At this point, we define an RPTM N_{no} that on input $\langle x, \psi \rangle$, simulates $N(\langle x, \psi \rangle)$ on a uniform random string r , answers “no” to the query τ made by N , and outputs $N(\langle x, \psi \rangle; r)$. Note that N_{no} does not require oracle. We then make a crucial observation that if x is not nice, then for every ψ , N_{no} determines $\langle x, \psi \rangle \in D$ in a BPP fashion. That is, if x is not nice, then the following conditions hold for every ψ : If $\langle x, \psi \rangle \in D$, then $N_{\text{no}}(\langle x, \psi \rangle)$ accepts with high probability, and if $\langle x, \psi \rangle \notin D$, then $N_{\text{no}}(\langle x, \psi \rangle)$ rejects with high probability. Since a BPP computation can be performed in P/poly , therefore, if x is not nice, then there is deterministic polynomial-time simulation of N_{no} that uses a polynomial-size advice string. (The P/poly simulation of N_{no} requires amplifying the success probability of N_{no} , which can be accomplished without any trouble, unlike the case of a bounded query randomized algorithm, since N_{no} does not require oracle.) Thus, it follows that if x is not nice, then for every ψ , we can determine $\langle x, \psi \rangle \in D$ in deterministic polynomial-time when

given this advice string. For $k \geq 2$, we use the definition of D , the expression for \overline{L}_k , the P/poly simulation of N_{no} , and the self-reducibility of SAT to show that for a non-nice input x , there is also a Σ_k^p -procedure to determine $x \notin L_k$ (details omitted due to lack of space). We combine the two Σ_k^p -procedures (one for nice input strings and the other for non-nice input strings) to get a Σ_k^p -procedure for \overline{L}_k .

We mention that our idea of partitioning input strings into *nice* and *non-nice* strings is inspired from the proof of $ZPP^{\text{NP}[1]} \subseteq S_2^p$ by Cai and Chakaravarthy [CC06]. However, we would like to stress that the results of Cai and Chakaravarthy [CC06] do not have any direct, obvious bearings on the solutions (particularly, our solutions) to the 1-versus-2 queries problem.

We compare the proof technique of Theorem 5 with those used previously in obtaining solutions to the 1-versus-2-queries problem. The proof of the statement “if $P_{tt}^{\Sigma_2^p[2]} \subseteq P^{\Sigma_2^p[1]}$, then $\text{PH} = \Sigma_2^p$ ” by Buhrman and Fortnow [BF99] used the following observation: If a set $A \in P^{B[1]}$ via a DPOTM M such that $A = L(M^{B[1]})$, then there is a deterministic polynomial-time computable function $h : \Sigma^* \rightarrow \Sigma^* \times \{+, -\}$ such that $x \in A$ if and only if either $h(x) = (z, +)$ and $z \in B$ or $h(x) = (z, -)$ and $z \notin B$. Here z is the query made by $M(x)$ to B if the outcome of $M(x)$ depends on the answer to the query, and z is some fixed string known to be in (or out of) B if the outcome of $M(x)$ is independent of the answer to the query. In our proof, the query made by a $ZPP^{B[1]}$ computation $M^{B[1]}$ may vary with the choice of random string r . Therefore, for our proof, we cannot say anything about the existence of a deterministic polynomial-time computable function h along the lines of the proof in [BF99]. The proof of the statement “for every integer $k > 2$, if $P_{tt}^{\Sigma_k^p[2]} \subseteq P^{\Sigma_k^p[1]}$, then $\text{PH} = \Sigma_k^p$ ” by Hemaspaandra, Hemaspaandra, and Hempel [HHH98]¹ used the fact that $P^{\Sigma_k^p[1]}$ has \leq_m^p -complete sets. For our proof, we cannot use arguments similar to those in the paper [HHH98], since it is not known whether $ZPP^{\Sigma_k^p[1]}$ has complete sets.

We next consider the hypothesis $P_{tt}^{\text{NP}[2]} \subseteq ZPP^{\text{NP}[1]}$. We show in Corollary 7 that the polynomial hierarchy collapses to S_2^p under this hypothesis. (Note that S_2^p is known to lie in between P^{NP} and $\Sigma_2^p \cap \Pi_2^p$.) Fortnow, Pavan, and Sengupta [FPS] obtained the same consequence under the stronger hypothesis $P_{tt}^{\text{NP}[2]} \subseteq P^{\text{NP}[1]}$. That is, they proved that if $P_{tt}^{\text{NP}[2]} \subseteq P^{\text{NP}[1]}$, then $\text{PH} = S_2^p$. Earlier, Buhrman and Fortnow [BF99] showed that if $P_{tt}^{\text{NP}[2]} \subseteq P^{\text{NP}[1]}$, then locally either every unsatisfiable formula has a short proof of unsatisfiability (i.e.,

¹ In fact, Hemaspaandra, Hemaspaandra, and Hempel [HHH98] proved a somewhat stronger statement. They proved that for any $0 \leq i < j < k$ and $i < k - 2$, if $P^{(\Sigma_j^p, \Sigma_k^p)} \subseteq P^{(\Sigma_i^p, \Sigma_k^p)}$, then $\text{PH} = \Sigma_k^p$.

coNP = NP) or SAT is decidable by a polynomial-size circuit. The proof of Fortnow, Pavan, and Sengupta [FPS] was built on the consequence of this result.

In Theorem 6, we derive the consequence stated in the aforementioned result of Buhrman and Fortnow [BF99] under the weaker hypothesis $P_{tt}^{NP[2]} \subseteq ZPP^{NP[1]}$. Since the proof of Fortnow, Pavan, and Sengupta [FPS] was built on this same consequence, we are able to obtain a collapse of the polynomial hierarchy to S_2^p under the hypothesis $P_{tt}^{NP[2]} \subseteq ZPP^{NP[1]}$.

Theorem 6. *If $P_{tt}^{NP[2]} \subseteq ZPP^{NP[1]}$, then there exist a polynomial-time predicate R and a constant $k > 0$ such that for every n one of the following statements is true:*

1. *Locally NP = coNP, i.e., for every boolean formula ϕ of length n , it holds that $\phi \notin \text{SAT} \iff (\exists w)R(\phi, w)$, where $|w|$ is polynomial in n .*
2. *there is a circuit of size n^k that decides SAT at length n .*

The proof of Theorem 6 builds on a technique developed by Buhrman and Fortnow [BF99]. Their proof technique involved partitioning unsatisfiable formulas ($\overline{\text{SAT}}$) into sets EASY- i and HARD- i , for each $i \in \{I, II, III, IV\}$. Then, they made use of properties of these sets in proving their aforementioned result. We modify the definitions of these sets for our purpose. We show that our new definitions of EASY- i and HARD- i sets retain some of the properties that Buhrman and Fortnow made use of in their proof. Our proof also makes use of a new notion called ‘‘P/poly-separator.’’ This notion is a natural generalization of the notion of *polynomial-time separator*, which Buhrman and Fortnow [BF99] coined in their proof argument.

Assume that the consequent in the statement of Theorem 6 holds. Fortnow, Pavan, and Sengupta [FPS] used this assumption to show that the polynomial hierarchy collapses to S_2^p . Recently, using the same assumption, Chakaravarthy and Roy [CR06] showed that PH collapses to $NO_2^p \cap YO_2^p$. Thus, we get:

Corollary 7. $P_{tt}^{NP[2]} \subseteq ZPP^{NP[1]} \implies \text{PH} = S_2^p = NO_2^p \cap YO_2^p$.

In a more recent work, Chang and Purini [CP07] showed that if the NP machine hypothesis holds, then $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$ implies $\text{PH} = \text{NP}$. For this result, they redefined the easy and hard sets in the proof given by Buhrman and Fortnow [BF99] of the statement ‘‘ $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$ implies that locally either $\text{NP} = \text{coNP}$ or $\text{NP} \subseteq \text{P/poly}$.’’ Using their new definition of the easy and hard sets, Chang and Purini [CP07] were able to show that the advice in the P/poly case can be made polynomially shorter than the input length. Thus, they showed

that if the P/poly case occurs infinitely often, then it would be possible to compute satisfiability in subexponential time in a way that violates the NP machine hypothesis.

We observe that the advice strings used in the proof of Theorem 6 include random strings r that can be too long for the NP machine hypothesis. Thus, it does not seem that the technique of Chang and Purini could help in showing “if the NP machine hypothesis holds, then $P_{tt}^{NP[2]} \subseteq ZPP^{NP[1]}$ implies $PH = NP$.”

4 Conclusion and Open Problems

We obtain solutions to the 1-versus-2 queries problem under hypotheses weaker than the previously considered hypotheses, namely, $P_{tt}^{\Sigma_k^p[2]} \subseteq P^{\Sigma_k^p[1]}$. We show that for each $k \geq 2$, if $P_{tt}^{\Sigma_k^p[2]} \subseteq ZPP^{\Sigma_k^p[1]}$, then $PH = \Sigma_k^p$, and if $P_{tt}^{NP[2]} \subseteq ZPP^{NP[1]}$, then $PH = S_2^p$. We list some open problems.

The foremost open problem is to see if our solutions can be extended to the solutions of the general 1-versus-2 queries problem: For any $k \geq 1$, whether there is a simulation of $P_{tt}^{\Sigma_k^p[2]}$ in $BPP^{\Sigma_k^p[1]}$. Buhrman and Fortnow [BF99] asked for implications of the hypothesis $BPP^{NP[2]} = BPP^{NP[1]}$, which is closely related to the problem we posed here. Hemaspaandra, Hemaspaandra, and Hempel [HHH98, HHH05] studied the m -versus- $(m + 1)$ queries problem. They obtained solutions to this problem under the hypothesis $P_{tt}^{\Sigma_k^p[m+1]} \subseteq P_{tt}^{\Sigma_k^p[m]}$, for each $k \geq 2$ and each $m > 0$. It could be possible to obtain the same solutions under the weaker hypothesis $P_{tt}^{\Sigma_k^p[m+1]} \subseteq ZPP_{tt}^{\Sigma_k^p[m]}$, for each $k \geq 2$ and each $m \geq 2$. It is interesting to note that no solution is known for this problem under the hypothesis $P_{tt}^{\Sigma_k^p[m+1]} \subseteq P_{tt}^{\Sigma_k^p[m]}$, for the case $k = 1$ and any $m \geq 2$. We would like to see a resolution of the m -versus- $(m + 1)$ queries problem for this special case not only under the hypothesis $P_{tt}^{NP[m+1]} \subseteq P_{tt}^{NP[m]}$ but also under the weaker hypothesis $P_{tt}^{NP[m+1]} \subseteq ZPP_{tt}^{NP[m]}$. Finally, in this paper we require the success probability of $ZPP^{\mathcal{C}[j]}$ algorithms, for any complexity class \mathcal{C} and integer $j \geq 1$, to be at least $1/2 + 1/\text{poly}(\cdot)$, where $\text{poly}(\cdot)$ can be any arbitrary polynomial. It would be interesting to see whether our results also hold when we require the success probability of $ZPP^{\mathcal{C}[j]}$ algorithms to be less than $1/2$.

References

- [BCG⁺96] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(3):421–433, 1996.

- [BCO93] R. Beigel, R. Chang, and M. Ogiwara. A relationship between difference hierarchies and relativized polynomial hierarchies. *Mathematical Systems Theory*, 26(3):293–310, 1993.
- [BF99] H. Buhrman and L. Fortnow. Two queries. *Journal of Computer and System Sciences*, 59(2):182–194, 1999.
- [Cai07] J. Cai. S_2^P is subset of ZPP^{NP} . *Journal of Computer and System Sciences*, 73(1):25–35, 2007.
- [Can96] R. Canetti. More on BPP and the polynomial-time hierarchy. *Information Processing Letters*, 57(5):237–241, 1996.
- [CC06] J. Cai and V. Chakaravarthy. On zero error algorithms having oracle access to one query. *Journal of Combinatorial Optimization*, 11(2):189–202, 2006.
- [CCHO05] J. Cai, V. Chakaravarthy, L. Hemaspaandra, and M. Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Information and Computation*, 198(1):1–23, 2005.
- [CK95] R. Chang and J. Kadin. On computing boolean connectives of characteristic functions. *Mathematical Systems Theory*, 28(3):173–198, 1995.
- [CK96] R. Chang and J. Kadin. The boolean hierarchy and the polynomial hierarchy: A closer connection. *SIAM Journal on Computing*, 25(2):340–354, 1996.
- [CP07] R. Chang and S. Purini. Bounded queries and the NP machine hypothesis. In *Proceedings of the 22nd Annual IEEE Conference on Computational Complexity*, pages 52–59. IEEE Computer Society Press, June 2007.
- [CR06] V. Chakaravarthy and S. Roy. Oblivious symmetric alternation. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science*, pages 230–241. Springer-Verlag *Lecture Notes in Computer Science #3884*, February 2006.
- [FPS] L. Fortnow, A. Pavan, and S. Sengupta. Proving SAT does not have small circuits with an application to the two queries problem. *Journal of Computer and System Sciences*. To appear.
- [HHH98] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. A downward collapse within the polynomial hierarchy. *SIAM Journal on Computing*, 28(2):383–393, 1998.
- [HHH05] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. Extending downward collapse from 1-versus-2 queries to m -versus- $m + 1$ queries. *SIAM Journal on Computing*, 34(6):1352–1369, 2005.
- [Kad88] J. Kadin. The polynomial time hierarchy collapses if the boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, 1988. Erratum appears in the same journal, 20(2):404.
- [KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th ACM Symposium on Theory of Computing*, pages 302–309. ACM Press, April 1980.
- [Kre88] M. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.
- [PSV06] A. Pavan, R. Santhanam, and N. Vinodchandran. Some results on average-case hardness within the polynomial hierarchy. In *Proceedings of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 188–199, 2006.
- [RS98] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Computational Complexity*, 7(2):152–162, 1998.
- [Wag89] K. Wagner. Number-of-query hierarchies. Technical Report 4, Institut für Informatik, Universität Würzburg, Würzburg, Germany, February 1989.
- [Yap83] C. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983.