

The 1-Versus-2 Queries Problem Revisited*

Rahul Tripathi[†]

Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620, USA
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 language L_k has an efficient simulation that makes at most 1 query to L_k . We obtain solutions to this problem for hypotheses weaker than previously considered. We prove that:

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

Here, for any complexity class \mathcal{C} and integer $j \geq 1$, we define $ZPP^{\mathcal{C}[j]}$ to be the class of problems solvable by zero-error randomized algorithms that run in polynomial time, make at most j queries to \mathcal{C} , and succeed with probability at least $1/2 + 1/\text{poly}(\cdot)$. This same definition 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 ours in (I) using the stronger hypothesis $P_{tt}^{\Sigma_k^p[2]} \subseteq P^{\Sigma_k^p[1]}$. Fortnow, Pavan, and Sengupta [FPS08] had obtained the same consequence as ours in (II) using the stronger hypothesis $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$.

Our results may also be viewed as steps towards obtaining solutions to arguably 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 $BPP^{\Sigma_k^p[1]}$.

Keywords: computational complexity, complexity classes, bounded queries, zero-error algorithms.

*A preliminary version of this paper appeared in *Proceedings of the 18th International Symposium on Algorithms and Computation (2007)* [Tri07].

[†]Research supported by the New Researcher Grant of the University of South Florida, Tampa.

1 Introduction

1.1 Background

Krentel [Kre88] studied the functional version of the 1-versus-2 queries problem. He 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 $P = NP$. There has also been work on the functional versions of other related query problems (see, e.g., [Kre88, ABG03, CP07]).

The decision version of the 1-versus-2 queries problem seems to be more difficult than its functional counterpart. A long list of work on the decision version of the 1-versus-2 queries problem has appeared in the literature. We review the progress made on this problem. Kadin [Kad88] proved that if $P^{NP[2]} \subseteq P^{NP[1]}$, then $NP \subseteq coNP/poly$. Yap [Yap83] proved that the polynomial hierarchy collapses to the third level if $NP \subseteq coNP/poly$, and Cai et al. [CCHO05] improved this collapse to S_2^{NP} under the same assumption.¹ Thus, Kadin's result implied 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 proved that if $P^{NP[2]} \subseteq P^{NP[1]}$, then every language in the polynomial hierarchy can be solved by a deterministic polynomial-time oracle 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 involving the hypothesis $P^{NP[2]} \subseteq P^{NP[1]}$ also hold if we assume the weaker hypothesis $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$.

Hemaspaandra, Hemaspaandra, and Hempel [HHH98] studied the 1-versus-2 queries problem in a more general context. They proved that for $k > 2$, if $P_{tt}^{\Sigma_k^p[2]} \subseteq P^{\Sigma_k^p[1]}$, then $PH = \Sigma_k^p$, and 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 proved 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 language 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 proved 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 $DIFF_m(\Sigma_k^p) = coDIFF_m(\Sigma_k^p) = BH(\Sigma_k^p)$.² It is known that if the boolean hierarchy collapses, then the polynomial hierarchy

¹ S_2^p , introduced independently by Russell and Sundaram [RS98] and by Canetti [Can96], is called the symmetric alternation class. The definition of S_2^{NP} is the same as that of S_2^p (see Definition 2.3) except with the following difference: In S_2^p , the polynomial-time predicate R has no oracle access whereas in S_2^{NP} , R has access to an NP oracle. It is known that $P^{\Sigma_2^p} \subseteq S_2^{NP} \subseteq \Sigma_3^p \cap \Pi_3^p$.

²For any complexity class \mathcal{C} , the boolean hierarchy $BH(\mathcal{C})$ over \mathcal{C} [CGH⁺88, CGH⁺89] is defined as follows: $BH(\mathcal{C}) =_{df} \bigcup_{m \geq 1} DIFF_m(\mathcal{C})$, where $DIFF_1(\mathcal{C}) = \mathcal{C}$ and for all $m \geq 1$, $DIFF_{m+1}(\mathcal{C}) = \{L \mid (\exists L_1 \in \mathcal{C})(\exists L_2 \in$

also collapses (see, e.g., [Kad88,Wag89,CK96,BCO93]). Thus as a consequence of this relationship between the two hierarchies (in particular, via a result of Beigel, Chang, and Ogiwara [BCO93]), the result of Hemaspaandra, Hemaspaandra, and Hempel [HHH98] also implied 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 every language in the polynomial hierarchy can be solved by a deterministic polynomial-time oracle Turing machine that makes $m - 1$ truth-table queries to Σ_{k+1}^p and an unbounded 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] proved that if $P_{tt}^{\Sigma_2^p[2]} \subseteq P^{\Sigma_2^p[1]}$, then $PH = \Sigma_2^p$. They also proved that no relativizable proof technique can establish a similar result for NP. That is, they showed that the result “if $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$, then $PH = NP$ ” cannot be proved using relativizable proof techniques. In spite of this negative result, they were able to obtain several other consequences, though weaker than the much sought after consequence $PH = NP$, of the $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$ hypothesis including (a) locally either $NP = coNP$ or NP has polynomial-size circuits, (b) $PH = BPP^{NP[1]}$, (c) $\Sigma_2^p \subseteq \Pi_2^p/1$, (d) $\Sigma_2^p = UP^{NP[1]} \cap RP^{NP[1]}$, and (e) $P^{NP} = P^{NP[1]}$. Our result in this paper for the hypothesis $P_{tt}^{NP[2]} \subseteq ZPP^{NP[1]}$ subsumes the consequences (a) and (c) obtained by Buhrman and Fortnow [BF99] for their stronger hypothesis $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$. However, we could not prove the consequences (b), (d), and (e) assuming $P_{tt}^{NP[2]} \subseteq ZPP^{NP[1]}$ because of a technical reason related to the fact that a ZPP computation may have multiple computation paths corresponding to choices of random strings.

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 proved 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 $DIFF_m(\Sigma_k^p) = coDIFF_m(\Sigma_k^p) = BH(\Sigma_k^p)$. 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 [FPS08] improved upon the result of Buhrman and Fortnow [BF99] by showing that if $P_{tt}^{NP[2]} \subseteq P^{NP[1]}$, then $PH = S_2^p$. Note that S_2^p satisfies $P^{NP} \subseteq S_2^p$ [RS98] and $S_2^p \subseteq ZPP^{NP}$ ([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 that is known to be 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]. This lemma 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 formulas of length n such that every circuit of size n^k fails to produce any

$DIFF_m(C)[L = L_1 - L_2]$. We refer the reader to the papers [CGH⁺88,CGH⁺89] for the original source, motivation, structural properties, and applications of the boolean hierarchy.

satisfying assignment for at least one formula from S . This lemma has found applications elsewhere (see [PSV06]).

Recently, Chakaravarthy and Roy [CR06] introduced the 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 theoretic 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 an 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 $\text{PH} = \text{NO}_2^p \cap \text{YO}_2^p$. This gives an improvement over the solution given by Fortnow, Pavan, and Sengupta [FPS08].

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 $\text{PH} = \text{NP}$. The NP machine hypothesis postulates that there exist an $0 < \epsilon < 1$ 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 for the hypothesis $P_{tt}^{\Sigma_k^p[2]} \subseteq \text{ZPP}^{\Sigma_k^p[1]}$, for integers $k \geq 1$. Note that $P^{\Sigma_k^p[1]}$ is a subclass of $\text{ZPP}^{\Sigma_k^p[1]}$, and so the hypothesis we consider here, namely, $P_{tt}^{\Sigma_k^p[2]} \subseteq \text{ZPP}^{\Sigma_k^p[1]}$, is weaker than the previously considered hypothesis $P_{tt}^{\Sigma_k^p[2]} \subseteq P^{\Sigma_k^p[1]}$ (see [Kad88,Wag89,BCO93,HHH98,BF99,FPS08,CR06,CP07]). Our results may also be seen 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 $\text{BPP}^{\Sigma_k^p[1]}$ simulation.

The classes $\text{ZPP}^{\Sigma_k^p[1]}$, 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 $S_2^p \subseteq \text{ZPP}^{\text{NP}}$; however, the question of whether this inclusion is also an equality is open. Cai and Chakaravarthy [CC06] proved that if one empowers a ZPP algorithm so that it is allowed to make at most one query to an NP oracle (for any input and random string), then such an algorithm can be simulated in S_2^p . In other words, they proved that $\text{ZPP}^{\text{NP}[1]} \subseteq S_2^p$. For $k \geq 2$, they proved that $\text{ZPP}^{\Sigma_k^p[1]} \subseteq P^{\Sigma_k^p[2]}$. They observed that $\text{BPP} \subseteq \text{ZPP}^{\text{NP}[1]}$ (also proven implicitly in some other papers) and used this observation to infer that it is hard to prove $\text{ZPP}^{\text{NP}[1]} \subseteq P^{\text{NP}}$ unconditionally since otherwise it would 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 $\text{ZPP}^{\mathcal{C}[1]}$ algorithms, for an arbitrary complexity class \mathcal{C} , is required to be at least $1/2 + 1/\text{poly}(\cdot)$.

2 Preliminaries

Our alphabet is $\Sigma = \{0, 1\}$. \mathbb{N} is the set of all nonnegative integers. We assume familiarity with basic notions in computational complexity theory such as complexity classes (P, NP,

ZPP, BPP, Σ_k^p , Π_k^p , $P^{\Sigma_k^p}$, PH, etc.), decision problems (SAT, Σ_k^p -complete languages), 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 a 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 an input x and let $M(x; r) \in \{\text{ACCEPT}, \text{REJECT}, ?\}$ also denote the outcome of a randomized Turing machine M on an input x and a 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 2.1 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 any complexity classes \mathcal{C} and \mathcal{D} ,

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

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 any complexity class \mathcal{C} ,

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

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

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

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 $ZPP^{C[j]}$, where C is the oracle and j is a bound on the number of queries to C made by the algorithm. Notice that a $ZPP^{C[j]}$ algorithm never yields a wrong answer, though on any input it may fail to produce an answer with bounded probability.

Some subtlety is involved when we talk about the success probability of a bounded query randomized algorithm (e.g., a $ZPP^{C[j]}$ 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 $ZPP^{C[j]}$ algorithms since an application of the same technique

would result in an algorithm that asks more than the allowed number j of queries to C . Hence, we need to specify 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 at least $1/2 + 1/\text{poly}(\cdot)$. 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 2.2 ([CC06]) *Let \mathcal{C} be a complexity class and $j \geq 1$ be an integer. A language $L \in \text{ZPP}^{C[j]}$ 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. *For 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,*

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

Remarks on the robustness of the class $\text{ZPP}^{C[j]}$ defined in Definition 2.2. The class ZPP has two equivalent definitions:

- (A) The class of problems solvable by randomized algorithms that always answer correctly and run in expected polynomial time.
- (B) The class of problems solvable by zero-error randomized algorithms that run in polynomial time and succeed with probability at least $1/\text{poly}(\cdot)$, where the probability is over the uniform choice of random strings.

These equivalences are not known to hold for $\text{ZPP}^{C[j]}$. Our definition of $\text{ZPP}^{C[j]}$ in Definition 2.2 requires the success probability to be at least $1/2 + 1/\text{poly}(\cdot)$ (instead of $1/\text{poly}(\cdot)$), and thus may only partially capture the interpretation of $\text{ZPP}^{C[j]}$ as in (B). However, our definition of $\text{ZPP}^{C[j]}$ does indeed fully capture the interpretation of $\text{ZPP}^{C[j]}$ as in (A), and so in this sense our definition of $\text{ZPP}^{C[j]}$ 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 \mathcal{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 outputs $?$ 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 ,

runs 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 2.2.

From the above remarks, it follows that $\text{ZPP}^{\Sigma_k^p[1]}$ subsumes the class \mathcal{D} of problems solvable by randomized algorithms that always answer correctly, run in expected polynomial time, and make at most 1 query to Σ_k^p . In this paper, we prove that a $\text{P}^{\Sigma_k^p[2]}$ algorithm cannot be simulated by a $\text{ZPP}^{\Sigma_k^p[1]}$ algorithm unless the polynomial hierarchy collapses. The definition of $\text{ZPP}^{\Sigma_k^p[1]}$ allows to conclude a similar result for algorithms defining the class \mathcal{D} . That is, it follows from our result and from the containment of the class \mathcal{D} in $\text{ZPP}^{\Sigma_k^p[1]}$ that a $\text{P}^{\Sigma_k^p[2]}$ algorithm cannot be simulated by an algorithm of the type \mathcal{D} unless the polynomial hierarchy collapses. Thus, using the definition of $\text{ZPP}^{\Sigma_k^p[1]}$ as in Definition 2.2 makes our results stronger than the ones we could have for algorithms of the type \mathcal{D} .

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

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

Here, we define $(\exists^p w) =_{df} (\exists w \in \Sigma^* : |w| \leq p(|x|))$ and $(\forall^p w) =_{df} (\forall w \in \Sigma^* : |w| \leq p(|x|))$.

The class S_2^p contains both MA and P^{NP} [RS98]. Cai [Cai07] proved that S_2^p is a subclass of ZPP^{NP} . Fortnow et al. [FIKU05] gave an alternative proof of this result and identified problems complete for the class promise- S_2^p , the promise version of S_2^p . They proved that the problem of approximating the value of a succinctly given zero-sum matrix game to within an additive factor is complete for promise- S_2^p . They also proved another problem, a version of Succinct Set Cover, to be complete for promise- S_2^p .

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

Definition 2.4 ([KL80]) For any complexity class \mathcal{C} and function $f : \mathbb{N} \rightarrow \mathbb{N}$, \mathcal{C}/f denotes the class of all languages L such that for some $C \in \mathcal{C}$ and for some 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 language L is said to be in \mathcal{C}/poly if and only if $L \in \mathcal{C}/p$ 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 $\phi(x_1 := \text{false}, x_2, \dots, x_n) \in \text{SAT}$. The 2-query disjunctive self-reducibility of SAT on ϕ induces a binary self-reduction tree T of ϕ as follows: The root of T is ϕ , and if ϕ' is a formula in T with free variables x_1, x_2, \dots, x_m , then the two child nodes of ϕ' are $\phi'(x_1 := \text{true}, x_2, \dots, x_m)$ and $\phi'(x_1 := \text{false}, x_2, \dots, x_m)$.

3 Results

3.1 The Case of Integers $k \geq 2$

Hemaspaandra, Hemaspaandra, and Hempel [HHH98] proved 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] proved that this result holds 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 3.1, we derive the same consequences under hypotheses weaker than the ones considered in the above two results.

Assuming the hypothesis $P_{tt}^{\Sigma_k^p[2]} \subseteq \text{ZPP}^{\Sigma_k^p[1]}$, for $k \geq 2$, we prove that for a Σ_k^p -complete language L_k , its complement $\overline{L_k}$ belongs to Σ_k^p . This will prove that if $P_{tt}^{\Sigma_k^p[2]} \subseteq \text{ZPP}^{\Sigma_k^p[1]}$, then $\Pi_k^p \subseteq \Sigma_k^p$ and hence the polynomial hierarchy collapses to Σ_k^p . For technical reasons, explained below while presenting the proof idea of Theorem 3.1, the proof of Theorem 3.1 is separated into two lemmas: Lemma 3.2 handles the case of even integers k and Lemma 3.3 handles the case of odd integers k .

For the sake of clarity, the polynomial-length bounds on quantifiers (\exists, \forall) are omitted almost everywhere throughout the paper.

Theorem 3.1 *For any integer $k \geq 2$,*

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

Proof Theorem 3.1 follows from Lemmas 3.2 and 3.3 stated below. ■ (Theorem 3.1)

We present the proof idea of Theorem 3.1. Fix an integer $k \geq 2$. Let L_k be a Σ_k^p -complete language. We can assume w.l.o.g. that L_k is defined so that for all $x \in \Sigma^*$, if k is even then

$$x \in L_k \iff (\exists y_1)(\forall y_2) \cdots (\exists y_{k-1})[\phi_{x, y_1, y_2, \dots, y_{k-1}} \notin \text{SAT}], \quad (3.a)$$

and if k is odd then

$$x \in L_k \iff (\exists y_1)(\forall y_2) \cdots (\forall y_{k-1})[\phi_{x, y_1, y_2, \dots, y_{k-1}} \in \text{SAT}]. \quad (3.b)$$

Here $\phi_{x, y_1, y_2, \dots, y_{k-1}}$ is a boolean formula computable in deterministic polynomial time given $x, y_1, y_2, \dots, y_{k-1}$. We also define a language D , where

$$\begin{aligned} D &=_{df} (L_k \times \overline{\text{SAT}}) \cup (\overline{L_k} \times \text{SAT}) \\ &= \{ \langle x, \psi \rangle \mid (x \in L_k \text{ and } \psi \notin \text{SAT}) \text{ or } (x \notin L_k \text{ and } \psi \in \text{SAT}) \}. \end{aligned}$$

Clearly, $D \in P^{(\text{NP}, \Sigma_k^p)}$, and hence by the hypothesis $P^{(\text{NP}, \Sigma_k^p)} \subseteq \text{ZPP}^{\Sigma_k^p[1]}$, 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 $D = L(N^{L_k[1]})$.

We partition all input strings x into the disjoint sets NICE and NON-NICE as follows: A string $x \in \text{NICE}$ if and only if there exist a formula ψ and a random string r (both of length bounded by some fixed polynomial in $|x|$) 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 τ and in fact $\tau \in L_k$.

On the other hand, a string $x \in \text{NON-NICE}$ if and only if $x \notin \text{NICE}$. (Thus, NON-NICE is the same as the set $\Sigma^* - \text{NICE}$.) The strings x that belong to the set NICE are called *nice*, otherwise they are called *non-nice* strings or *not nice*.

Because the base computation of $N^{L_k[1]}$ is a ZPP computation and because a ZPP computation is never wrong, we can easily show that if input strings x are restricted to nice strings, then $\overline{L_k}$ is in Σ_k^p . More precisely, we can describe a Σ_k^p -procedure $\sigma_k^{\text{NICE}}(\cdot)$ such that for all $x \in \text{NICE}$,

$$x \in \overline{L_k} \iff \sigma_k^{\text{NICE}}(x) = \text{ACCEPT}. \quad (3.c)$$

For a nice string x , the Σ_k^p -procedure $\sigma_k^{\text{NICE}}(x)$ 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 this case, we notice that for every formula ψ and for every random string r (both of length bounded by some fixed polynomial in $|x|$) 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.” Note that the base computation of $N^{L_k[1]}$ is a ZPP computation and a ZPP computation yields correct answer with high probability and never yields a wrong answer. Therefore, for every polynomially length-bounded formula ψ , the fraction of polynomially length-bounded random strings r for which $N^{L_k[1]}(\langle x, \psi \rangle; r)$ reaches a definite outcome, which would be ACCEPT if $\langle x, \psi \rangle \in D$ and REJECT if $\langle x, \psi \rangle \notin D$ since $D = L(N^{L_k[1]})$, 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 polynomially length-bounded formula ψ , there is a large fraction of polynomially length-bounded random strings r such that

- the query τ made by $N^{L_k[1]}(\langle x, \psi \rangle; r)$ to L_k is answered “no,” and
- $N^{L_k[1]}(\langle x, \psi \rangle; r) = \text{ACCEPT}$ if $\langle x, \psi \rangle \in D$ and $N^{L_k[1]}(\langle x, \psi \rangle; r) = \text{REJECT}$ if $\langle x, \psi \rangle \notin D$.

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 an oracle. We then make the crucial observation that if x is not nice, then for every polynomially length-bounded 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 polynomially length-bounded 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.

Notice that in the simulation of N_{no} on input $\langle x, \psi \rangle$, if the answers supplied to the oracle queries corresponding to different choices of random strings r are correct, then the outcomes

of N_{no} are either ACCEPT and ? or REJECT and ?. But if the answers supplied to the oracle queries for choices of random strings r are incorrect, then the outcomes of N_{no} on such r 's could be ACCEPT, REJECT, and ?.

Since a BPP computation can be performed in P/poly, if x is not nice then there is a deterministic polynomial-time simulation of N_{no} that uses a polynomial-length 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 an oracle.) Thus, it follows that if x is not nice, then for every polynomially length-bounded ψ , we can determine whether $\langle x, \psi \rangle \in D$ in deterministic polynomial time when given this advice string.

For even integers k , we use the expression

$$x \notin L_k \iff (\forall y_1)(\exists y_2) \cdots (\forall y_{k-1})[\phi_{x,y_1,y_2,\dots,y_{k-1}} \in \text{SAT}]$$

given by Statement (3.a) in order to describe a Σ_k^p -procedure for deciding the membership of non-nice strings in $\overline{L_k}$. We show in Lemma 3.2 that for any even integer $k \geq 2$ we can use the definition of D , the P/poly simulation of N_{no} , and the self-reducibility of SAT to find a satisfying assignment for each instance $\phi_{x,y_1,y_2,\dots,y_{k-1}}$ belonging to SAT in deterministic polynomial time. As a result, we show in Lemma 3.2 that for every even integer $k \geq 2$, there is a Σ_k^p -procedure $\sigma_k^{\text{NON-NICE}}(\cdot)$ such that for all $x \in \text{NON-NICE}$,

$$x \in \overline{L_k} \iff \sigma_k^{\text{NON-NICE}}(x) = \text{ACCEPT}. \quad (3.d)$$

For odd integers k , the expression for determining whether $x \notin L_k$ is

$$x \notin L_k \iff (\forall y_1)(\exists y_2) \cdots (\exists y_{k-1})[\phi_{x,y_1,y_2,\dots,y_{k-1}} \notin \text{SAT}]$$

by Statement (3.b). That is, in this case for a non-nice input x , we need to prove that $\phi_{x,y_1,y_2,\dots,y_{k-1}} \notin \text{SAT}$. However, we cannot certify in deterministic polynomial time that some unsatisfiable instance $\phi_{x,y_1,y_2,\dots,y_{k-1}}$ is not in SAT unless we know for sure the advice string r' required for the P/poly simulation of N_{no} . Therefore, in Lemma 3.3, for a non-nice input x , we first describe a Σ_3^p -procedure that finds this correct advice string r' and then we use r' to certify in deterministic polynomial time that all unsatisfiable instances $\phi_{x,y_1,y_2,\dots,y_{k-1}}$ are not in SAT. As a result, we show in Lemma 3.3 that for every odd integer $k \geq 3$, there is a Σ_k^p -procedure $\sigma_k^{\text{NON-NICE}}(\cdot)$ such that for all $x \in \text{NON-NICE}$,

$$x \in \overline{L_k} \iff \sigma_k^{\text{NON-NICE}}(x) = \text{ACCEPT}. \quad (3.e)$$

Finally, since Σ_k^p is closed under union operation, we combine the Σ_k^p -procedures $\sigma_k^{\text{NICE}}(\cdot)$ and $\sigma_k^{\text{NON-NICE}}(\cdot)$, where the first one appears in Statement (3.c) and the second one appears in Statement (3.d) for even integers $k \geq 2$ and in Statement (3.e) for odd integers $k \geq 3$, to obtain for every integer $k \geq 2$, a Σ_k^p -procedure that accepts $\overline{L_k}$. Thus, it follows that $\Sigma_k^p = \Pi_k^p = \text{PH}$.

We next compare the proof technique of Theorem 3.1 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 language $A \in P^{B[1]}$ 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]³ used the fact that $P^{\Sigma_k^p[1]}$ has \leq_m^p -complete languages. 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 languages. In fact, there are relativized worlds where ZPP and many other promise complexity classes do not have complete languages (see, e.g., [Sip82,HJV93]).

We now come to the statement and the proof of Lemma 3.2.

Lemma 3.2 *Let $k \geq 2$ be an arbitrary even integer. Then the following holds:*

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

Proof Let L_k denote a Σ_k^p -complete language such that for every $x \in \Sigma^*$:

$$x \in L_k \iff (\exists y_1)(\forall y_2) \cdots (\exists y_{k-1})[\phi_{x, y_1, y_2, \dots, y_{k-1}} \notin \text{SAT}],$$

where $\phi_{x, y_1, y_2, \dots, y_{k-1}}$ is a boolean formula computable in deterministic polynomial time given $x, y_1, y_2, \dots, y_{k-1}$. Without loss of generality assume that $|\phi_{x, y_1, y_2, \dots, y_{k-1}}| = m$, where m is some polynomial in $|x|$. Define a language D , where

$$\begin{aligned} D &=_{df} (L_k \times \overline{\text{SAT}}) \cup (\overline{L_k} \times \text{SAT}) \\ &= \{\langle x, \psi \rangle \mid (x \in L_k \text{ and } \psi \notin \text{SAT}) \text{ or } (x \notin L_k \text{ and } \psi \in \text{SAT})\}. \end{aligned}$$

Clearly, $D \in P^{(\text{NP}, \Sigma_k^p)}$. By the assumption $P^{(\text{NP}, \Sigma_k^p)} \subseteq ZPP^{\Sigma_k^p[1]}$, it follows that $D \in ZPP^{\Sigma_k^p[1]}$.

Let $N^{L_k[1]}$ be a $ZPP^{\Sigma_k^p[1]}$ machine such that $D = L(N^{L_k[1]})$ and the success probability of $N^{L_k[1]}$ is at least $1/2 + 1/\text{poly}(\cdot)$, where $\text{poly}(\cdot)$ denotes some fixed polynomial. Thus, the following holds for all $\langle x, \psi \rangle$:

$$\begin{aligned} \langle x, \psi \rangle \in D &\implies \text{Prob}_r[N^{L_k[1]}(\langle x, \psi \rangle; r) = \text{ACCEPT}] \geq \frac{1}{2} + \frac{1}{\text{poly}(|\langle x, \psi \rangle|)} \\ &\text{and} \\ \langle x, \psi \rangle \notin D &\implies \text{Prob}_r[N^{L_k[1]}(\langle x, \psi \rangle; r) = \text{REJECT}] \geq \frac{1}{2} + \frac{1}{\text{poly}(|\langle x, \psi \rangle|)}. \end{aligned} \tag{3.f}$$

³In 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 $\text{PH} = \Sigma_k^p$.

Notice that for all $\langle x, \psi \rangle \in D$ and for all r , if $N^{L_k[1]}(\langle x, \psi \rangle; r)$ does not output ACCEPT then it outputs ?, and for all $\langle x, \psi \rangle \notin D$ and for all r , if it does not output REJECT then it outputs ?.

We call a string x *nice* if and only if there exist a formula ψ , where $|\psi| \leq m$, and a polynomially length-bounded string r such that the following holds:

$$(\star) N^{L_k[1]}(\langle x, \psi \rangle; r) \text{ makes a query } \tau \text{ for some } \tau \in L_k \text{ and } N^{L_k[1]}(\langle x, \psi \rangle; r) \in \{\text{ACCEPT}, \text{REJECT}\}.$$

A nice string does not need to be in $\overline{L_k}$. The following Σ_k^p -procedure decides the membership of nice strings x in $\overline{L_k}$: Guess polynomially length-bounded ψ and r , and test whether (\star) holds. More precisely, we show in Claim 1 that if x is nice, then the following statement is true:

$$x \notin L_k \iff (\exists \psi)(\exists r)(\exists y_1)(\forall y_2) \cdots (\exists y_{k-1}) [Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \text{ or } Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1})],$$

where $Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1}) =_{df}$

- $\psi \in \text{SAT}$ and
- $N^{L_k[1]}(\langle x, \psi \rangle; r)$ queries τ and
- $N(\langle x, \psi \rangle; r) = \text{ACCEPT}$ when the answer to the query τ is considered “yes” and
- $\phi_{\tau, y_1, y_2, \dots, y_{k-1}} \notin \text{SAT}$,

and $Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1}) =_{df}$

- $N^{L_k[1]}(\langle x, \psi \rangle; r)$ queries τ and
- $N(\langle x, \psi \rangle; r) = \text{REJECT}$ when the answer to the query τ is considered “yes” and
- $\psi \notin \text{SAT}$ and
- $\phi_{\tau, y_1, y_2, \dots, y_{k-1}} \notin \text{SAT}$.

Claim 1 *The following statements are true:*

1. $(x \text{ is nice and } x \notin L_k) \implies$

$$(\exists \psi : |\psi| \leq m)(\exists r)(\exists y_1)(\forall y_2) \cdots (\exists y_{k-1}) \left[Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \text{ or } Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \right].$$

- 2.

$$(\exists \psi : |\psi| \leq m)(\exists r)(\exists y_1)(\forall y_2) \cdots (\exists y_{k-1}) \left[Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \text{ or } Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \right]$$

$$\implies x \notin L_k.$$

Proof of Claim 1. Recall that a nice string does not need to be in $\overline{L_k}$. Suppose that x is nice and that $x \in \overline{L_k}$. By the definition of a nice string, there exist a formula ψ and a string r (both of length bounded by some fixed polynomial in $|x|$) such that

- $N^{L_k[1]}(\langle x, \psi \rangle; r)$ makes a query τ and $\tau \in L_k$, and
- $N^{L_k[1]}(\langle x, \psi \rangle; r) \in \{\text{ACCEPT}, \text{REJECT}\}$.

If $\psi \in \text{SAT}$, then by the definition of D and the assumption that $x \notin L_k$, it follows that $\langle x, \psi \rangle \in D$; so, $N^{L_k[1]}(\langle x, \psi \rangle; r) = \text{ACCEPT}$ since $D = L(N^{L_k[1]})$. Furthermore, since $\tau \in L_k$, we have

$$(\exists y_1)(\forall y_2) \cdots (\exists y_{k-1})[\phi_{\tau, y_1, y_2, \dots, y_{k-1}} \notin \text{SAT}]$$

by the definition of L_k . Thus, if $\psi \in \text{SAT}$, then we get

$$(\exists \psi)(\exists r)(\exists y_1)(\forall y_2) \cdots (\exists y_{k-1})[Q_1(x, \psi, r, y_1, \dots, y_{k-1})].$$

Similarly, we can show that if $\psi \notin \text{SAT}$, then we get

$$(\exists \psi)(\exists r)(\exists y_1)(\forall y_2) \cdots (\exists y_{k-1})[Q_2(x, \psi, r, y_1, \dots, y_{k-1})].$$

Next suppose that the statement

$$(\exists \psi)(\exists r)(\exists y_1)(\forall y_2) \cdots (\exists y_{k-1})[Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1})]$$

is true. (The proof under the assumption

$$“(\exists \psi)(\exists r)(\exists y_1)(\forall y_2) \cdots (\exists y_{k-1})[Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1})]”$$

is similar.) Then, by the definition of L_k and Q_1 , it follows that $\tau \in L_k$. Therefore, the query τ made by $N(\langle x, \psi \rangle; r)$ to L_k is answered “yes,” and so by the definition of Q_1 , $N^{L_k[1]}(\langle x, \psi \rangle; r) = \text{ACCEPT}$. Since $N^{L_k[1]}$ accepts D with zero-error, we conclude that $\langle x, \psi \rangle \in D$. By the definition of Q_1 , we note that $\psi \in \text{SAT}$. Hence, $x \notin L_k$ by the definition of D . ■ (Claim 1)

We now deal with the case that x is not nice. In this case, we have

$$(\forall \psi : |\psi| \leq m)(\forall r)[N^{L_k[1]}(\langle x, \psi \rangle; r) \text{ queries } \tau \text{ and } \tau \notin L_k, \text{ or } N^{L_k[1]}(\langle x, \psi \rangle; r) = ?]. \quad (3.g)$$

Let N_{no} be an RPTM that on input $\langle x, \psi \rangle$, simulates $N(\langle x, \psi \rangle)$ on a polynomially length-bounded uniform random string r , answers “no” to the query τ made by N , and outputs $N(\langle x, \psi \rangle; r)$. Thus, in particular, $N_{\text{no}}(\langle x, \psi \rangle; r)$ denotes the outcome of $N^{L_k[1]}(\langle x, \psi \rangle; r)$ when the answer to the query τ is considered “no.” Recall that for all $\langle x, \psi \rangle \in D$ and for all r , if $N^{L_k[1]}(\langle x, \psi \rangle; r)$ does not output ACCEPT then it outputs ?, and for all $\langle x, \psi \rangle \notin D$ and for all r , if it does not output REJECT then it outputs ?. Also, note that N_{no} does not require an oracle.

If x is not nice, then Statement (3.f) and Statement (3.g) imply that for every ψ such that $|\psi| \leq m$,

$$\begin{aligned} \langle x, \psi \rangle \in D &\implies \text{Prob}_r[N_{\text{no}}(\langle x, \psi \rangle; r) = \text{ACCEPT}] \geq \frac{1}{2} + \frac{1}{\text{poly}(|x|)} \\ &\text{and} \\ \langle x, \psi \rangle \notin D &\implies \text{Prob}_r[N_{\text{no}}(\langle x, \psi \rangle; r) = \text{REJECT}] \geq \frac{1}{2} + \frac{1}{\text{poly}(|x|)}, \end{aligned} \tag{3.h}$$

where $\text{poly}(\cdot)$ denotes some fixed polynomial.

Notice that in the simulation of N_{no} on input $\langle x, \psi \rangle$, if the answers supplied to the oracle queries corresponding to different choices of random strings r are correct, then the outcomes of N_{no} are either ACCEPT and ? or REJECT and ?. But if the answers supplied to the oracle queries for choices of random strings r are incorrect, then the outcomes of N_{no} on such r 's could be ACCEPT, REJECT, and ?.

We amplify the success probability of N_{no} using a standard technique. Specifically, define an RPTM \widehat{N} that on input $\langle x, \psi \rangle$, where $|\psi| \leq m$, performs $m \cdot \text{poly}^2(|x|)$ independent simulations of $N_{\text{no}}(\langle x, \psi \rangle; r)$ (i.e., in every such simulation, r is chosen uniformly at random and independently of the polynomially length-bounded random strings picked in previous simulations), accepts if *majority* of the simulations accept, and rejects if *majority* of the simulations reject. Since N_{no} does not require an oracle, \widehat{N} also does not require an oracle.

By an application of Chernoff bounds, it follows from Statement (3.h) that for every ψ such that $|\psi| \leq m$,

$$\begin{aligned} \langle x, \psi \rangle \in D &\implies \text{Prob}_r[\widehat{N}(\langle x, \psi \rangle; r) = \text{ACCEPT}] \geq 1 - \frac{1}{2^{m+1}} \\ &\text{and} \\ \langle x, \psi \rangle \notin D &\implies \text{Prob}_r[\widehat{N}(\langle x, \psi \rangle; r) = \text{REJECT}] \geq 1 - \frac{1}{2^{m+1}}. \end{aligned} \tag{3.i}$$

Thus, it follows from Statement (3.i) and by the probabilistic method that there exists a polynomially length-bounded string r' such that for every ψ such that $|\psi| \leq m$,

$$\begin{aligned} \langle x, \psi \rangle \in D &\iff \widehat{N}(\langle x, \psi \rangle; r') = \text{ACCEPT} \\ &\text{and} \\ \langle x, \psi \rangle \notin D &\iff \widehat{N}(\langle x, \psi \rangle; r') = \text{REJECT}. \end{aligned} \tag{3.j}$$

We say that a string r' is correct for x , if r' satisfies: For every ψ such that $|\psi| \leq m$,

$$\begin{aligned} \psi \in \text{SAT} &\iff \widehat{N}(\langle x, \psi \rangle; r') = \text{ACCEPT} \\ &\text{and} \\ \psi \notin \text{SAT} &\iff \widehat{N}(\langle x, \psi \rangle; r') = \text{REJECT}. \end{aligned} \tag{3.k}$$

If a string r' is not correct for x , then we say that r' is incorrect for x .

We can define a deterministic polynomial-time predicate $P(x, \psi, r')$, where $|\psi| \leq m$ and r' is the presumed random string used by $\widehat{N}(\langle x, \psi \rangle)$, such that

$$\begin{aligned} P(x, \psi, r') \text{ is true} &\implies \psi \in \text{SAT} \\ &\text{and} \\ P(x, \psi, r') \text{ is false} &\implies \psi \notin \text{SAT} \text{ or } r' \text{ is incorrect for } x. \end{aligned} \tag{3.1}$$

The predicate $P(x, \psi, r')$ uses the self-reducibility of SAT and the relation

$$(\psi \in \text{SAT} \iff \widehat{N}(\langle x, \psi \rangle; r') = \text{ACCEPT}) \text{ and } (\psi \notin \text{SAT} \iff \widehat{N}(\langle x, \psi \rangle; r') = \text{REJECT})$$

to determine whether a satisfying assignment for ψ exists. If a satisfying assignment for ψ is found, then $P(x, \psi, r')$ returns *true*, otherwise $P(x, \psi, r')$ returns *false*. Clearly, if r' is correct for x , then for every ψ such that $|\psi| \leq m$, it holds that $\psi \in \text{SAT} \iff P(x, \psi, r')$ is true.

A non-nice string does not need to be in \overline{L}_k . We use the relation

$$x \in L_k \iff (\exists y_1)(\forall y_2) \cdots (\exists y_{k-1})[\phi_{x, y_1, \dots, y_{k-1}} \notin \text{SAT}]$$

to show in Claim 2 that there is a Σ_k^p -procedure that decides the membership of non-nice strings x in \overline{L}_k . This is the first place in the proof of Lemma 3.2 where we crucially require that k is an even integer.

Claim 2 *The following statements are true:*

1. $(x \text{ is not nice and } x \notin L_k) \implies (\exists r')(\forall y_1)(\exists y_2) \cdots (\forall y_{k-1})[P(x, \phi_{x, y_1, y_2, \dots, y_{k-1}}, r')]$.
2. $(\exists r')(\forall y_1)(\exists y_2) \cdots (\forall y_{k-1})[P(x, \phi_{x, y_1, y_2, \dots, y_{k-1}}, r')] \implies x \notin L_k$.

Proof of Claim 2. Recall that a non-nice string does not need to be in \overline{L}_k . Suppose that x is not nice and that $x \in \overline{L}_k$. Then by the definition of L_k , we have that the statement

$$(\forall y_1)(\exists y_2) \cdots (\forall y_{k-1})[\phi_{x, y_1, y_2, \dots, y_{k-1}} \in \text{SAT}]$$

is true. Also, then by the definition of D , Statement (3.j) and Statement (3.k) imply that there exists a polynomially length-bounded correct string r' for x . Thus, for every ψ such that $|\psi| \leq m$, we can deterministically determine whether $\psi \in \text{SAT}$ using the polynomial-time predicate $P(x, \psi, r')$ defined in Statement (3.1). In particular, we can determine whether $\phi_{x, y_1, y_2, \dots, y_{k-1}} \in \text{SAT}$ via deciding $P(x, \phi_{x, y_1, y_2, \dots, y_{k-1}}, r')$ in deterministic polynomial time. Hence, the consequence follows.

Next suppose that the statement

$$(\exists r')(\forall y_1)(\exists y_2) \cdots (\forall y_{k-1})[P(x, \phi_{x, y_1, y_2, \dots, y_{k-1}}, r')] \tag{3.m}$$

is true. Then by the definition of P (see Statement (3.1)), we have that the statement

$$(\forall y_1)(\exists y_2) \cdots (\forall y_{k-1})[\phi_{x, y_1, y_2, \dots, y_{k-1}} \in \text{SAT}]$$

is true. Note that it does not matter if the r' in Statement (3.m) is one of the correct r' for x since, for any ψ such that $|\psi| \leq m$, $P(x, \psi, r')$ returns true only if a witness for the membership of ψ in SAT is found. Hence, it follows from the definition of L_k that $x \notin L_k$. \blacksquare (Claim 2)

From Claims 1 and 2, we get Claim 3.

Claim 3 For every $x \in \Sigma^*$, $x \notin L_k \iff [\sigma_k^{\text{NICE}}(x) \text{ or } \sigma_k^{\text{NON-NICE}}(x)]$, where

$$\sigma_k^{\text{NICE}}(x) \equiv (\exists \psi : |\psi| \leq m)(\exists r)(\exists y_1)(\forall y_2) \cdots (\exists y_{k-1}) \left[Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \text{ or } Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \right]$$

and

$$\sigma_k^{\text{NON-NICE}}(x) \equiv (\exists r')(\forall y_1)(\exists y_2) \cdots (\forall y_{k-1}) \left[P(x, \phi_{x, y_1, y_2, \dots, y_{k-1}}, r') \right].$$

Proof of Claim 3. Suppose that $x \notin L_k$. If x is nice, then Claim 1(1) implies that $\sigma_k^{\text{NICE}}(x)$ is true, and so the consequence is true. If x is not nice, then Claim 2(1) implies that $\sigma_k^{\text{NON-NICE}}(x)$ is true, and so the consequence is true.

The other direction of the implication follows directly from Claim 1(2) and Claim 2(2). \blacksquare (Claim 3)

Notice that both $\sigma_k^{\text{NICE}}(\cdot)$ and $\sigma_k^{\text{NON-NICE}}(\cdot)$ are Σ_k^p -predicates, and so the *or* of these two predicates is also a Σ_k^p -predicate. Hence, it follows from Claim 3 that $\overline{L}_k \in \Sigma_k^p$. This proves that $\Sigma_k^p = \Pi_k^p = \text{PH}$, since L_k is Σ_k^p -complete. \blacksquare (Lemma 3.2)

We next come to the statement and the proof of Lemma 3.3.

Lemma 3.3 Let $k \geq 3$ be an arbitrary odd integer. Then the following holds:

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

Proof The proof differs from that of Lemma 3.2 in essentially the way we deal with non-nice strings. Let L_k denote a Σ_k^p -complete language such that for every $x \in \Sigma^*$:

$$x \in L_k \iff (\exists y_1)(\forall y_2) \cdots (\forall y_{k-1}) [\phi_{x, y_1, y_2, \dots, y_{k-1}} \in \text{SAT}].$$

Here $\phi_{x, y_1, y_2, \dots, y_{k-1}}$ is a boolean formula computable in deterministic polynomial time given $x, y_1, y_2, \dots, y_{k-1}$. Let $|\phi_{x, y_1, y_2, \dots, y_{k-1}}| = m$, where m is some polynomial in $|x|$. We define a set D , a $\text{ZPP}^{\Sigma_k^p[1]}$ machine N , and the notion of *nice* strings as in the proof of Lemma 3.2.

A nice string does not need to be in \overline{L}_k . The following Σ_k^p -procedure decides the membership of nice strings x in \overline{L}_k :

$$x \notin L_k \iff (\exists \psi)(\exists r)(\exists y_1)(\forall y_2) \cdots (\forall y_{k-1}) [Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \text{ or } Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1})],$$

where $Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1}) =_{df}$

- $N^{L_k[1]}(\langle x, \psi \rangle; r)$ queries τ and
- $N(\langle x, \psi \rangle; r) = \text{ACCEPT}$ when the answer to the query τ is considered “yes” and
- $\psi \in \text{SAT}$ and
- $\phi_{\tau, y_1, y_2, \dots, y_{k-1}} \in \text{SAT}$,

and $Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1}) =_{df}$

- $\psi \notin \text{SAT}$ and
- $N^{L_k[1]}(\langle x, \psi \rangle; r)$ queries τ and
- $N(\langle x, \psi \rangle; r) = \text{REJECT}$ when the answer to the query τ is considered “yes” and
- $\phi_{\tau, y_1, y_2, \dots, y_{k-1}} \in \text{SAT}$.

Thus, we can prove the following claim similar to the proof of Claim 1.

Claim 4 *The following statements are true:*

1. $(x \text{ is nice and } x \notin L_k) \implies$

$$(\exists \psi : |\psi| \leq m)(\exists r)(\exists y_1)(\forall y_2) \cdots (\forall y_{k-1})[Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \text{ or } Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1})].$$

- 2.

$$(\exists \psi : |\psi| \leq m)(\exists r)(\exists y_1)(\forall y_2) \cdots (\forall y_{k-1})[Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \text{ or } Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1})]$$

$$\implies x \notin L_k.$$

We now consider the case that x is not nice. Let \widehat{N} be an RPTM defined as in the proof of Lemma 3.2. Then the following holds: There exists a polynomially length-bounded string r' such that for every ψ such that $|\psi| \leq m$,

$$\begin{aligned} \langle x, \psi \rangle \in D &\iff \widehat{N}(\langle x, \psi \rangle; r') = \text{ACCEPT} \\ &\text{and} \\ \langle x, \psi \rangle \notin D &\iff \widehat{N}(\langle x, \psi \rangle; r') = \text{REJECT}. \end{aligned} \tag{3.n}$$

A non-nice string need not be in $\overline{L_k}$. We use the relation

$$x \in L_k \iff (\exists y_1)(\forall y_2) \cdots (\forall y_{k-1})[\phi_{x, y_1, y_2, \dots, y_{k-1}} \in \text{SAT}]$$

to show in Claim 5 that there is a Σ_k^p -predicate that decides the membership of non-nice strings x in $\overline{L_k}$. This is the first place in the proof of Lemma 3.3 where we require that k is an odd integer.

In the remaining part of this proof, we will need the predicate $R(x, \psi, r')$, which we define as follows:

$$R(x, \psi, r') =_{df} \left(\psi \in \text{SAT} \iff [\widehat{N}(\langle x, \psi \rangle; r') = \text{ACCEPT}] \right) \text{ and} \\ \left(\psi \notin \text{SAT} \iff [\widehat{N}(\langle x, \psi \rangle; r') = \text{REJECT}] \right).$$

Claim 5 *The following statements are true:*

1. $(x \text{ is not nice and } x \notin L_k) \implies$

$$(\exists r') \left[(\forall \psi : |\psi| \leq m) R(x, \psi, r') \text{ and} \right. \\ \left. (\forall y_1)(\exists y_2) \cdots (\exists y_{k-1}) [\widehat{N}(\langle x, \phi_{x, y_1, y_2, \dots, y_{k-1}} \rangle; r') = \text{REJECT}] \right].$$

2.

$$(\exists r') \left[(\forall \psi : |\psi| \leq m) R(x, \psi, r') \text{ and} \right. \\ \left. (\forall y_1)(\exists y_2) \cdots (\exists y_{k-1}) [\widehat{N}(\langle x, \phi_{x, y_1, y_2, \dots, y_{k-1}} \rangle; r') = \text{REJECT}] \right]$$

$$\implies x \notin L_k.$$

Proof of Claim 5. Note that a non-nice string does not need to be in $\overline{L_k}$. Suppose that x is not nice and that $x \in \overline{L_k}$. By the definition of L_k and the assumption that $x \notin L_k$, we have that the statement

$$(\forall y_1)(\exists y_2) \cdots (\exists y_{k-1}) [\phi_{x, y_1, y_2, \dots, y_{k-1}} \notin \text{SAT}]$$

is true. Since x is not nice and $x \notin L_k$, it follows from Statement (3.n) and the definition of D that there exists a polynomially length-bounded string r' such that for every ψ , where $|\psi| \leq m$,

$$\begin{aligned} \psi \in \text{SAT} &\iff \widehat{N}(\langle x, \psi \rangle; r') = \text{ACCEPT} \\ &\text{and} \\ \psi \notin \text{SAT} &\iff \widehat{N}(\langle x, \psi \rangle; r') = \text{REJECT}. \end{aligned} \tag{3.o}$$

This proves that $(\forall \psi : |\psi| \leq m) R(x, \psi, r')$. Note that since $|\phi_{x, y_1, y_2, \dots, y_{k-1}}| = m$, we can apply Statement (3.o) with $\phi_{x, y_1, y_2, \dots, y_{k-1}}$ in place of ψ . Thus, we can determine whether $\phi_{x, y_1, y_2, \dots, y_{k-1}} \notin \text{SAT}$ by deciding whether $\widehat{N}(\langle x, \phi_{x, y_1, y_2, \dots, y_{k-1}} \rangle; r') = \text{REJECT}$. Hence, the consequence follows.

For the other direction, suppose that the antecedent in part (2) of the claim is true. Since $(\forall \psi : |\psi| \leq m) R(x, \psi, r')$ and $|\phi_{x, y_1, y_2, \dots, y_{k-1}}| = m$, we can use the definition of $R(x, \psi, r')$ to show that Statement (3.o) is true for $\psi := \phi_{x, y_1, y_2, \dots, y_{k-1}}$ and for the assumed r' . Thus,

we can replace “ $\widehat{N}(\langle x, \phi_{x,y_1,y_2,\dots,y_{k-1}} \rangle; r') = \text{REJECT}$ ” by “ $\phi_{x,y_1,y_2,\dots,y_{k-1}} \notin \text{SAT}$ ” in the statement

$$(\forall y_1)(\exists y_2) \cdots (\exists y_{k-1})[\widehat{N}(\langle x, \phi_{x,y_1,y_2,\dots,y_{k-1}} \rangle; r') = \text{REJECT}],$$

which is known to be true by our assumption. Hence, we have that the statement

$$(\forall y_1)(\exists y_2) \cdots (\exists y_{k-1})[\phi_{x,y_1,y_2,\dots,y_{k-1}} \notin \text{SAT}]$$

is true. This proves, by the definition of L_k , that $x \notin L_k$. ■ (Claim 5)

From Claims 4 and 5, we get Claim 6.

Claim 6 For every $x \in \Sigma^*$, $x \notin L_k \iff [\sigma_k^{\text{NICE}}(x) \text{ or } \sigma_k^{\text{NON-NICE}}(x)]$, where

$$\sigma_k^{\text{NICE}}(x) \equiv (\exists \psi : |\psi| \leq m)(\exists r)(\exists y_1)(\forall y_2) \cdots (\forall y_{k-1}) \left[Q_1(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \text{ or } \right. \\ \left. Q_2(x, \psi, r, y_1, y_2, \dots, y_{k-1}) \right]$$

and

$$\sigma_k^{\text{NON-NICE}}(x) \equiv (\exists r') \left[(\forall \psi : |\psi| \leq m) R(x, \psi, r') \text{ and } \right. \\ \left. (\forall y_1)(\exists y_2) \cdots (\exists y_{k-1})[\widehat{N}(\langle x, \phi_{x,y_1,y_2,\dots,y_{k-1}} \rangle; r') = \text{REJECT}] \right].$$

Proof of Claim 6. Suppose that $x \notin L_k$. If x is nice, then Claim 4(1) implies the consequence. If x is not nice, then Claim 5(1) implies the consequence.

The other direction of the implication follows directly from Claim 4(2) and Claim 5(2). ■ (Claim 6)

Notice that both $\sigma_k^{\text{NICE}}(\cdot)$ and $\sigma_k^{\text{NON-NICE}}(\cdot)$ are Σ_k^p -predicates, and so the *or* of these two predicates is also a Σ_k^p -predicate. Hence, it follows from Claim 6 that $\overline{L}_k \in \Sigma_k^p$. This proves that $\Sigma_k^p = \Pi_k^p = \text{PH}$, since L_k is Σ_k^p -complete. ■ (Lemma 3.3)

Theorem 3.1 cannot be strengthened via a relativizable proof so that the statement “ $\text{P}_{tt}^{\text{NP}[2]} \subseteq \text{ZPP}^{\text{NP}[1]} \implies \text{PH} = \text{NP}$ ” holds because of the following result of Buhrman and Fortnow [BF99]: There exists a relativized world where $\text{NP} \neq \text{coNP}$ but $\text{P}^{\text{NP}[1]} = \text{PSPACE}$.

We mention that our idea of partitioning input strings into *nice* and *non-nice* strings in the proofs of Lemmas 3.2 and 3.3 is inspired from the work of Cai and Chakaravarthy [CC06]. In their proof of the result $\text{ZPP}^{\text{NP}[1]} \subseteq \text{S}_2^p$, they defined nice strings to be those input strings x of a $\text{ZPP}^{\text{SAT}[1]}$ computation M for which there is some random string r such that M accepts or rejects x along r and the oracle query τ made by M along r is satisfiable. They showed that there is an easy-to-describe S_2^p proof system for nice strings and there is also an S_2^p proof system for non-nice strings. Our proofs of Lemmas 3.2 and 3.3 also have a similar flavor. We defined, though in a manner different from theirs, *nice* and *non-nice* strings, and showed that if input strings are restricted to nice strings, then $\overline{L}_k \in \Sigma_k^p$, and if

they are restricted to non-nice strings, then also $\overline{L_k} \in \Sigma_k^p$. (Here, L_k is the Σ_k^p -complete set used in the description of the proof idea of Theorem 3.1 and in the proofs of Lemmas 3.2 and 3.3.) However, we would like to stress that the results of Cai and Chakaravathy [CC06] do not have any direct, obvious bearings on the solutions (particularly, our solutions) to the 1-versus-2 queries problem.

3.2 The Case of $k = 1$

We next consider the hypothesis $P_{tt}^{\text{NP}[2]} \subseteq ZPP^{\text{NP}[1]}$. We show in Corollary 3.12 that the polynomial hierarchy collapses to S_2^p under this hypothesis. (Note that S_2^p is known to lie between P^{NP} and $\Sigma_2^p \cap \Pi_2^p$.) Fortnow, Pavan, and Sengupta [FPS08] 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., $\text{coNP} = \text{NP}$) or SAT is decidable by a polynomial-size circuit. The proof of Fortnow, Pavan, and Sengupta [FPS08] was built on the consequence of this result.

In Theorem 3.4, we derive the consequence stated in the aforementioned result of Buhrman and Fortnow [BF99] under the weaker hypothesis $P_{tt}^{\text{NP}[2]} \subseteq ZPP^{\text{NP}[1]}$. Since the proof of Fortnow, Pavan, and Sengupta [FPS08] was built on this same consequence, we obtain a collapse of the polynomial hierarchy to S_2^p under the hypothesis $P_{tt}^{\text{NP}[2]} \subseteq ZPP^{\text{NP}[1]}$.

Theorem 3.4 *If $P_{tt}^{\text{NP}[2]} \subseteq ZPP^{\text{NP}[1]}$, 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 $\text{NP} = \text{coNP}$, 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 bounded by a fixed polynomial in n .*
2. *There is a circuit of size n^k that decides SAT at length n .*

For the proof of Theorem 3.4, we will need the following languages:

- $C =_{df} \overline{\text{SAT}} \times \text{SAT}$
 $= \{ \langle \phi, \psi \rangle \mid \phi \in \overline{\text{SAT}} \text{ and } \psi \in \text{SAT} \}.$
- $D =_{df} (\text{SAT} \times \overline{\text{SAT}}) \cup (\overline{\text{SAT}} \times \text{SAT})$
 $= \{ \langle \phi, \psi \rangle \mid (\phi \in \text{SAT} \text{ and } \psi \in \overline{\text{SAT}}) \text{ or } (\phi \in \overline{\text{SAT}} \text{ and } \psi \in \text{SAT}) \}.$

Note that both C and D are in $P_{tt}^{\text{NP}[2]}$, and so by the assumption $P_{tt}^{\text{NP}[2]} \subseteq ZPP^{\text{NP}[1]}$, they are in $ZPP^{\text{NP}[1]}$. Let $N_1^{\text{SAT}[1]}$ and $N_2^{\text{SAT}[1]}$ be $ZPP^{\text{NP}[1]}$ machines such that $C = L(N_1^{\text{SAT}[1]})$, $D = L(N_2^{\text{SAT}[1]})$, and both $N_1^{\text{SAT}[1]}$ and $N_2^{\text{SAT}[1]}$ succeed with probability at least $1/2 + 1/\text{poly}(\cdot)$, for some fixed polynomial $\text{poly}(\cdot)$.

We use the framework of “easy-hard” arguments in the proof of Theorem 3.4. These arguments were introduced by Kadin [Kad88], and since then they and their variations have found applications in several papers (see [BCO93,CK96,HHH98,BF99,HHH05,CP07]).

The proof of Theorem 3.4 builds on a variation of “easy-hard” arguments developed by Buhrman and Fortnow [BF99]. Their proof technique involved partitioning unsatisfiable formulas 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 (see Definitions 3.5 and 3.6) retain the properties, which we identify in Proposition 3.7, that Buhrman and Fortnow made use of in their proof.

We next define four sets of formulas: EASY-I, EASY-II, EASY-III, and EASY-IV. All of these sets are contained in $\overline{\text{SAT}}$. Also, for each of these sets, the formulas that belong to the set have a short proof of membership. In other words, each of these sets is in NP.

Definition 3.5 *The sets of formulas EASY-I, EASY-II, EASY-III, and EASY-IV are defined as follows:*

1. *A formula ϕ is in EASY-I if and only if*
 - (a) *there exist a formula ψ of length $|\phi|$ and a polynomially length-bounded string r such that*
 - $N_1^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) = \text{ACCEPT}$ and
 - $N_1^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r)$ makes a query τ and $\tau \in \text{SAT}$.
2. *A formula ϕ is in EASY-II if and only if*
 - (a) $\phi \in \text{EASY-I}$, or
 - (b) ϕ is a leaf of a self-reduction tree and is false, or
 - (c) ϕ self-reduces to two EASY-I formulas.
3. *A formula ϕ is in EASY-III if and only if*
 - (a) $\phi \in \text{EASY-II}$, or
 - (b) *there exist a formula $\psi \in \text{EASY-II}$ of length $|\phi|$ and a polynomially length-bounded string r such that*
 - $N_1^{\text{SAT}[1]}(\langle \psi, \phi \rangle; r) = \text{REJECT}$ and
 - $N_1^{\text{SAT}[1]}(\langle \psi, \phi \rangle; r)$ makes a query τ and $\tau \in \text{SAT}$.
4. *A formula ϕ is in EASY-IV if and only if*
 - (a) $\phi \in \text{EASY-III}$, or
 - (b) *there exist a formula $\psi \in \text{SAT}$ of length $|\phi|$ and a polynomially length-bounded string r such that*
 - $N_2^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) = \text{ACCEPT}$ and
 - $N_2^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r)$ makes a query τ and $\tau \in \text{SAT}$,

or

- (c) *there exist a formula $\psi \in \text{EASY-III}$ of length $|\phi|$ and a polynomially length-bounded string r such that*
- $N_2^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) = \text{REJECT}$ and
 - $N_2^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r)$ makes a query τ and $\tau \in \text{SAT}$.

In the above definition of the set EASY-I of formulas, the formula ψ in fact belongs to SAT since $\langle \phi, \psi \rangle \in L(N_1^{\text{SAT}[1]}) = \overline{\text{SAT}} \times \text{SAT}$. Also, in the above definition of the set EASY-II of formulas, we use the self-reduction algorithm for SAT. Recall from the end of Section 2 that the self-reduction algorithm for SAT on any boolean formula $\phi(x_1, x_2, \dots, x_n)$ induces a binary self-reduction tree T as follows: The root of T is ϕ , and if ϕ' is a formula in T with free variables x_1, x_2, \dots, x_m , then the two child nodes of ϕ' are $\phi'(x_1 := \text{true}, x_2, \dots, x_m)$ and $\phi'(x_1 := \text{false}, x_2, \dots, x_m)$.

We next define four sets of formulas: HARD-I, HARD-II, HARD-III, and HARD-IV. All of these sets are contained in $\overline{\text{SAT}}$. Also, for each of these sets, the formulas that do not belong to the set have a short proof of nonmembership. In other words, each of these sets is in coNP.

Definition 3.6 *The sets of formulas HARD-I, HARD-II, HARD-III, and HARD-IV are defined as follows: For each $i \in \{\text{I, II, III, IV}\}$,*

$$\text{HARD-}i = \overline{\text{SAT}} \cap \overline{\text{EASY-}i}.$$

- Proposition 3.7**
1. *For each $i \in \{\text{I, II, III, IV}\}$, every formula ϕ is either in SAT or in EASY- i or in HARD- i .*
 2. *For each $i \in \{\text{I, II, III, IV}\}$, EASY- $i \in \text{NP}$ and HARD- $i \in \text{coNP}$.*
 3. $\text{EASY-I} \subseteq \text{EASY-II} \subseteq \text{EASY-III} \subseteq \text{EASY-IV} \subseteq \overline{\text{SAT}}$.
 4. $\text{HARD-IV} \subseteq \text{HARD-III} \subseteq \text{HARD-II} \subseteq \text{HARD-I} \subseteq \overline{\text{SAT}}$.
 5. *If there is a formula $\phi \in \text{HARD-I}$, then there is a formula $\alpha \in \text{HARD-I} \cap \text{EASY-II}$.*

Proof Propositions 3.7(1)–(4) follow easily from Definitions 3.5 and 3.6. So, we prove Proposition 3.7(5) only. This proof is the same as the proof of Lemma 5.8 by Buhrman and Fortnow [BF99]. Let ϕ_1 be a deepest formula in the self-reduction tree of ϕ such that $\phi_1 \in \text{HARD-I}$. Note that $\phi_1 \notin \text{SAT}$ since $\phi_1 \in \text{HARD-I}$ and by definition $\text{HARD-I} \subseteq \overline{\text{SAT}}$. Then either ϕ_1 is a leaf of the self-reduction tree or ϕ_1 self-reduces to two EASY-I formulas. In both the cases, $\phi_1 \in \text{EASY-II}$ by Definition 3.5(2). ■ (Proposition 3.7)

We introduce in Definition 3.8 the notion of the P/poly-separator between two disjoint sets at a fixed length. This notion is a natural generalization of the notion of the *polynomial-time separator*, which Buhrman and Fortnow [BF99] (informally) coined in their proof argument. (In the context of Definition 3.8, the term *polynomial-time separator* would be called P-separator.)

Definition 3.8 Fix a DPTM M and a polynomial p . For each $n \in \mathbb{N}$, let $\text{adv}(n)$ be an advice string of length $p(n)$ for M . We say that $[M; \text{adv}(n)]$ is a P/poly-separator between disjoint sets S_1 and S_2 at length n if the following holds: For every $x \in \Sigma^n$,

$$\begin{aligned} x \in S_1 &\implies M(x; \text{adv}(n)) = \text{ACCEPT}, \\ x \in S_2 &\implies M(x; \text{adv}(n)) = \text{REJECT}, \text{ and} \\ x \notin S_1 \cup S_2 &\implies M(x; \text{adv}(n)) \in \{\text{ACCEPT}, \text{REJECT}\}. \end{aligned}$$

Here, $M(x; \text{adv}(n))$ denotes the output of M on input x and advice string $\text{adv}(n)$.

Our plan of proving Theorem 3.4 is as follows. We will first show that there are DPTMs M_1 and M_2 , and polynomials $p_1(\cdot)$ and $p_2(\cdot)$ such that for every integer n , the following holds: If there is a HARD-IV formula of length n then there are advice strings $\text{adv}_1(n)$ of length $p_1(n)$ and $\text{adv}_2(n)$ of length $p_2(n)$ such that

1. $[M_1; \text{adv}_1(n)]$ is a P/poly-separator between SAT and EASY-III at length n .
2. $[M_2; \text{adv}_2(n)]$ is a P/poly-separator between SAT and HARD-III at length n .

Statement (1) will be shown in Lemma 3.10 and Statement (2) will be shown using Proposition 3.7 and Lemma 3.9. From Statements (1) and (2), we will show that there is a polynomial $p(\cdot)$ such that for all n , if there is a HARD-IV formula at length n , then SAT is accepted by a circuit of size $p(n)$. On the other hand, if there is no HARD-IV formula at length n , then EASY-IV and $\overline{\text{SAT}}$ will coincide at length n by the definition of HARD-IV. Hence, in this case at length n , $\overline{\text{SAT}}$ will be accepted by the NP algorithm for EASY-IV. Thus, we will have at each length n , SAT accepted by a P/poly algorithm or $\overline{\text{SAT}}$ accepted by an NP algorithm. So, the consequence of Theorem 3.4 will follow.

We now come to the detailed proof of Theorem 3.4. We start with proving Lemma 3.9.

Buhrman and Fortnow [BF99] showed that if there is a formula $\phi \in \text{HARD-I} \cap \text{EASY-II}$, then there is a deterministic polynomial-time separator between SAT and HARD-III.⁴ We show in Lemma 3.9 that if there is a formula $\phi \in \text{HARD-I} \cap \text{EASY-II}$, then there is a P/poly-separator between SAT and HARD-III at length $|\phi|$.⁵

Lemma 3.9 *There exist a DPTM M and a polynomial $p(\cdot)$ such that the following holds for all $n \in \mathbb{N}$: If there is a formula $\phi \in \text{HARD-I} \cap \text{EASY-II}$ of length n , then there is an advice string $\text{adv}(n)$ of length $p(n)$ such that $[M; \text{adv}(n)]$ is a P/poly-separator between SAT and HARD-III at length n .*

Proof Let $\phi \in \text{HARD-I} \cap \text{EASY-II}$. By the definition of HARD-I formulas and the assumption that $\phi \in \text{HARD-I}$, we have that for all $\psi \in \text{SAT}$ of length $|\phi|$ and for all polynomially length-bounded strings r ,

$$\begin{aligned} N_1^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) &\in \{\text{REJECT}, ?\}, \\ &\text{or} \\ N_1^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) &\text{ makes a query } \tau \text{ and } \tau \notin \text{SAT}, \end{aligned} \tag{3.p}$$

⁴Referring to the definitions in [BF99].

⁵Referring to Definitions 3.5 and 3.6 in this paper.

since otherwise ϕ would be in EASY-I. By the definition of HARD-III formulas, we have that the following holds for all formulas $\phi' \in \text{HARD-III}$: For all $\psi' \in \text{EASY-II}$ of length $|\phi'|$ and for all polynomially length-bounded strings r ,

- $N_1^{\text{SAT}[1]}(\langle \psi', \phi' \rangle; r) \in \{\text{ACCEPT}, ?\}$, or
- $N_1^{\text{SAT}[1]}(\langle \psi', \phi' \rangle; r)$ makes a query τ and $\tau \notin \text{SAT}$,

since otherwise ϕ' would be in EASY-III. By switching the order of quantifiers on ϕ' and ψ' , we have that the following holds for all formulas $\psi' \in \text{EASY-II}$: For all $\phi' \in \text{HARD-III}$ of length $|\psi'|$ and for all polynomially length-bounded strings r ,

- $N_1^{\text{SAT}[1]}(\langle \psi', \phi' \rangle; r) \in \{\text{ACCEPT}, ?\}$, or
- $N_1^{\text{SAT}[1]}(\langle \psi', \phi' \rangle; r)$ queries τ and $\tau \notin \text{SAT}$.

Using the assumption that $\phi \in \text{EASY-II}$, and setting ϕ for ψ' and ψ for ϕ' in the above definition of EASY-II, we have that the following holds: For all $\psi \in \text{HARD-III}$ of length $|\phi|$ and for all polynomially length-bounded strings r ,

$$\begin{aligned} N_1^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) &\in \{\text{ACCEPT}, ?\}, \\ &\text{or} \\ N_1^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) &\text{ queries } \tau \text{ and } \tau \notin \text{SAT}. \end{aligned} \tag{3.q}$$

Let $N_{1,\text{no}}$ be an RPTM that on input $\langle \phi, \psi \rangle$, simulates $N_1(\langle \phi, \psi \rangle)$ on a uniform random string r , answers “no” to the query τ asked by N_1 , and outputs $N_1(\langle \phi, \psi \rangle; r)$. Thus, $N_{1,\text{no}}(\langle \phi, \psi \rangle; r)$ denotes the output of $N_1(\langle \phi, \psi \rangle; r)$ when the answer to the query τ is considered “no.” Note that $N_{1,\text{no}}$ does not require an oracle.

Since $\phi \in \text{HARD-I}$, by Proposition 3.7(4), $\phi \notin \text{SAT}$. Thus, by the definition of C , for every $\psi \in \text{SAT}$, $\langle \phi, \psi \rangle \in C$. Also, since $N_1^{\text{SAT}[1]}$ is a $\text{ZPP}^{\text{NP}[1]}$ machine accepting C , it follows that for every $\psi \in \text{SAT}$ and every r , $N_1^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) \in \{\text{ACCEPT}, ?\}$. The success probability of N_1 and Statement (3.p) imply that for every $\psi \in \text{SAT}$ of length $|\phi|$,

$$\text{Prob}_r [N_{1,\text{no}}(\langle \phi, \psi \rangle; r) = \text{ACCEPT}] \geq \frac{1}{2} + \frac{1}{\text{poly}(|\phi|)}. \tag{3.r}$$

Here, $\text{poly}(\cdot)$ denotes some fixed polynomial. Note that we write $\text{poly}(|\phi|)$ instead of $\text{poly}(|\langle \phi, \psi \rangle|)$, since $|\psi|$ is a polynomial in $|\phi|$ (in fact, $|\psi| = |\phi|$).

Since $\phi \notin \text{SAT}$ and every $\psi \in \text{HARD-III}$ formula is in $\overline{\text{SAT}}$ (by Proposition 3.7(4)), it follows from the definition of C that $\langle \phi, \psi \rangle \notin C$. Thus, using the fact that $N_1^{\text{SAT}[1]}$ is a $\text{ZPP}^{\text{NP}[1]}$ machine accepting C , it follows that for every $\psi \in \text{HARD-III}$ and every r , $N_1^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) \in \{\text{REJECT}, ?\}$. The success probability of N_1 and Statement (3.q) imply that for every formula $\psi \in \text{HARD-III}$ of length $|\phi|$,

$$\text{Prob}_r [N_{1,\text{no}}(\langle \phi, \psi \rangle; r) = \text{REJECT}] \geq \frac{1}{2} + \frac{1}{\text{poly}(|\phi|)}. \tag{3.s}$$

By amplifying the success probability of $N_{1,\text{no}}$ and using the probabilistic method as we did in the proof of Lemma 3.2, we can define an RPTM \widehat{N} and a polynomially length-bounded string r' such that for every formula ψ of length $|\phi|$,

$$\begin{aligned} \psi \in \text{SAT} &\implies \widehat{N}(\langle \phi, \psi \rangle; r') = \text{ACCEPT}, \\ &\text{and} \\ \psi \in \text{HARD-III} &\implies \widehat{N}(\langle \phi, \psi \rangle; r') = \text{REJECT}, \end{aligned} \tag{3.t}$$

and if $\psi \notin \text{SAT} \cup \text{HARD-III}$, then $\widehat{N}(\langle \phi, \psi \rangle; r') \in \{\text{ACCEPT}, \text{REJECT}\}$. Let M be a DPTM that on any input ψ of length $|\phi|$, gets $\langle \phi, r' \rangle$ as an advice string and simulates $\widehat{N}(\langle \phi, \psi \rangle; r')$. That is, $M(\psi; \langle \phi, r' \rangle) =_{df} \widehat{N}(\langle \phi, \psi \rangle; r')$. Thus, it follows from Statement (3.t) that $[M; \langle \phi, r' \rangle]$ is a P/poly-separator between SAT and HARD-III at length $|\phi|$. ■ (Lemma 3.9)

Buhrman and Fortnow [BF99] showed that if there is a formula $\phi \in \text{HARD-IV}$, then there is a deterministic polynomial-time separator between SAT and EASY-III.⁶ We show in Lemma 3.10 that if there is a formula $\phi \in \text{HARD-IV}$, then there is a P/poly-separator between SAT and EASY-III at length $|\phi|$.⁷

Lemma 3.10 *There exist a DPTM M and a polynomial $p(\cdot)$ such that the following holds for all $n \in \mathbb{N}$: If there is a formula $\phi \in \text{HARD-IV}$ of length n , then there is an advice string $\text{adv}(n)$ of length $p(n)$ such that $[M; \text{adv}(n)]$ is a P/poly-separator between SAT and EASY-III at length n .*

Proof Let ϕ be in HARD-IV. By the definition of HARD-IV formulas, we have that for all $\psi \in \text{SAT}$ of length $|\phi|$ and for all polynomially length-bounded strings r ,

$$\begin{aligned} N_2^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) &\in \{\text{REJECT}, ?\}, \\ &\text{or} \\ N_2^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) &\text{ makes a query } \tau \text{ and } \tau \notin \text{SAT}, \end{aligned} \tag{3.u}$$

since otherwise ϕ would be in EASY-IV. Similarly, we also have that for all $\psi \in \text{EASY-III}$ of length $|\phi|$ and for all polynomially length-bounded strings r ,

$$\begin{aligned} N_2^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) &\in \{\text{ACCEPT}, ?\}, \\ &\text{or} \\ N_2^{\text{SAT}[1]}(\langle \phi, \psi \rangle; r) &\text{ makes a query } \tau \text{ and } \tau \notin \text{SAT}, \end{aligned} \tag{3.v}$$

since otherwise ϕ would be in EASY-IV.

Let $N_{2,\text{no}}$ be an RPTM that on input $\langle \phi, \psi \rangle$, simulates $N_2(\langle \phi, \psi \rangle)$ on a uniform random string r , answers “no” to the query τ asked by N_2 , and outputs $N_2(\langle \phi, \psi \rangle; r)$. Thus,

⁶Referring to the definitions in [BF99].

⁷Referring to Definitions 3.5 and 3.6 in this paper.

$N_{2,\text{no}}(\langle\phi, \psi\rangle; r)$ denotes the output of $N_2(\langle\phi, \psi\rangle; r)$ when the answer to the query τ is considered “no.” Note that $N_{2,\text{no}}$ does not require an oracle.

Since $\phi \in \text{HARD-IV}$, by Proposition 3.7(4), $\phi \notin \text{SAT}$. Thus, for every $\psi \in \text{SAT}$, $\langle\phi, \psi\rangle \in D$ by the definition of D . Also, since $N_2^{\text{SAT}[1]}$ is a $\text{ZPP}^{\text{NP}[1]}$ machine accepting D , it follows that for every $\psi \in \text{SAT}$ and every r , $N_2^{\text{SAT}[1]}(\langle\phi, \psi\rangle; r) \in \{\text{ACCEPT}, ?\}$. Then the success probability of $N_2^{\text{SAT}[1]}$ and Statement (3.u) imply that for every $\psi \in \text{SAT}$ of length $|\phi|$,

$$\text{Prob}_r [N_{2,\text{no}}(\langle\phi, \psi\rangle; r) = \text{ACCEPT}] \geq \frac{1}{2} + \frac{1}{\text{poly}(|\phi|)}. \quad (3.w)$$

By Proposition 3.7(3) and the definition of D , for every $\psi \in \text{EASY-III}$, we have that $\langle\phi, \psi\rangle \notin D$. Thus, using the fact that $N_2^{\text{SAT}[1]}$ is a $\text{ZPP}^{\text{NP}[1]}$ machine accepting D , it follows that for every $\psi \in \text{EASY-III}$ and for every r , $N_2^{\text{SAT}[1]}(\langle\phi, \psi\rangle; r) \in \{\text{REJECT}, ?\}$. Then the success probability of $N_2^{\text{SAT}[1]}$ and Statement (3.v) imply that for every $\psi \in \text{EASY-III}$ of length $|\phi|$,

$$\text{Prob}_r [N_{2,\text{no}}(\langle\phi, \psi\rangle; r) = \text{REJECT}] \geq \frac{1}{2} + \frac{1}{\text{poly}(|\phi|)}. \quad (3.x)$$

By amplifying the success probability of $N_{2,\text{no}}$ and using the probabilistic method as we did in the proof of Lemma 3.2, we can show that there exist an RPTM \widehat{N} and a polynomially length-bounded string r' such that for every ψ of length $|\phi|$,

$$\begin{aligned} \psi \in \text{SAT} &\implies \widehat{N}(\langle\phi, \psi\rangle; r') = \text{ACCEPT}, \\ &\text{and} \\ \psi \in \text{EASY-III} &\implies \widehat{N}(\langle\phi, \psi\rangle; r') = \text{REJECT}, \end{aligned} \quad (3.y)$$

and if $\psi \notin \text{SAT} \cup \text{EASY-III}$, then $\widehat{N}(\langle\phi, \psi\rangle; r') \in \{\text{ACCEPT}, \text{REJECT}\}$. Let M be a DPTM that on any input ψ of length $|\phi|$, gets $\langle\phi, r'\rangle$ as an advice string and simulates $\widehat{N}(\langle\phi, \psi\rangle; r')$. That is, $M(\psi; \langle\phi, r'\rangle) =_{\text{df}} \widehat{N}(\langle\phi, \psi\rangle; r')$. Thus, it follows from Statement (3.y) that $[M; \langle\phi, r'\rangle]$ is a P/poly-separator between SAT and EASY-III at length $|\phi|$. ■ (Lemma 3.10)

We now proceed to the proof of Theorem 3.4.

Proof of Theorem 3.4. Fix n . If $\Sigma^n \cap \overline{\text{SAT}} = \text{EASY-IV}$, then by Proposition 3.7(2), the NP algorithm for EASY-IV accepts $\overline{\text{SAT}}$ at length n .

So assume otherwise that there exists $\phi \in \Sigma^n \cap \overline{\text{SAT}}$ such that $\phi \notin \text{EASY-IV}$. Then $\phi \in \text{HARD-IV}$ by Proposition 3.7(1). It follows from Lemma 3.10 that there exist a fixed DPTM M_1 , a fixed polynomial p_1 , and an advice string $\text{adv}_1(n)$ with $|\text{adv}_1(n)| = p_1(n)$ such that $[M_1; \text{adv}_1]$ is a P/poly-separator between SAT and EASY-III at length n .

By Proposition 3.7(4), ϕ also belongs to HARD-I. Thus, by Proposition 3.7(5), there is a formula $\alpha \in \text{HARD-I} \cap \text{EASY-II}$. W.l.o.g., we can assume that $|\alpha| = n$. It follows from

Lemma 3.9 that there exist a fixed DPTM M_2 , a fixed polynomial p_2 , and an advice string $\text{adv}_2(n)$ with $|\text{adv}_2(n)| = p_2(n)$ such that $[M_2; \text{adv}_2]$ is a P/poly-separator between SAT and HARD-III at length n .

Consider the P/poly algorithm M shown in Figure 1.

Algorithm M

Input: $\psi \in \Sigma^n$

Advice (adv): If $\Sigma^n \cap \text{HARD-IV} = \emptyset$, then $\text{adv} := 0$. If $\phi \in \Sigma^n \cap \text{HARD-IV}$, then $\text{adv} := \langle 1, \text{adv}_1, \text{adv}_2 \rangle$, where adv_1 and adv_2 are such that at length n , (1) $[M_1; \text{adv}_1]$ is a P/poly-separator between SAT and EASY-III, and (2) $[M_2; \text{adv}_2]$ is a P/poly-separator between SAT and HARD-III.

1. If $\text{adv} = 0$ then REJECT.
 - ▷ Otherwise, $\text{adv} = \langle 1, \text{adv}_1, \text{adv}_2 \rangle$
2. If $M_1(\psi; \text{adv}_1) \neq \text{ACCEPT}$ then
 - ▷ ψ cannot be in SAT
 - REJECT
3. If $M_2(\psi; \text{adv}_2) \neq \text{ACCEPT}$ then
 - ▷ ψ cannot be in SAT
 - REJECT
4. Else
 - ▷ We now have $M_1(\psi; \text{adv}_1) = \text{ACCEPT}$ and $M_2(\psi; \text{adv}_2) = \text{ACCEPT}$.
 - ▷ Thus, ψ must be in SAT, since ψ cannot be in EASY-III and in HARD-III.
 - ACCEPT

Figure 1: The P/poly algorithm M .

Claim 7 M accepts SAT at length n if $\Sigma^n \cap \text{HARD-IV} \neq \emptyset$.

Proof of Claim 7. Assume that $\Sigma^n \cap \text{HARD-IV} \neq \emptyset$. Then $\text{adv} := \langle 1, \text{adv}_1, \text{adv}_2 \rangle$, where $[M_1; \text{adv}_1]$ is a P/poly-separator between SAT and EASY-III at length n , and (2) $[M_2; \text{adv}_2]$ is a P/poly-separator between SAT and HARD-III at length n .

If $\psi \in \Sigma^n \cap \text{SAT}$, then $M_1(\psi; \text{adv}_1) = \text{ACCEPT}$ and $M_2(\psi; \text{adv}_2) = \text{ACCEPT}$. Thus, in this case M accepts ψ . If $\psi \notin \text{SAT}$, then either $\psi \in \text{EASY-III}$ or $\psi \in \text{HARD-III}$. If $\psi \in \text{EASY-III}$, then $M_1(\psi; \text{adv}_1) = \text{REJECT}$, and so M rejects ψ . If $\psi \in \text{HARD-III}$, then $M_2(\psi; \text{adv}_2) = \text{REJECT}$, and so M rejects ψ . ■ (Claim 7)

Thus, we have shown that there is an NP algorithm and a P/poly algorithm such that if $\Sigma^n \cap \text{HARD-IV} = \emptyset$, then the NP algorithm accepts $\overline{\text{SAT}}$ at length n , and if $\Sigma^n \cap \text{HARD-IV} \neq \emptyset$, then the P/poly algorithm accepts SAT at length n . This completes

the proof of Theorem 3.4

■ (Theorem 3.4)

Assume that the consequence in the statement of Theorem 3.4 holds. Fortnow, Pavan, and Sengupta [FPS08] used this assumption to prove that the polynomial hierarchy PH collapses to S_2^p . Chakaravarthy and Roy [CR06] proved that under the same assumption PH collapses to $\text{NO}_2^p \cap \text{YO}_2^p$. The classes NO_2^p and YO_2^p satisfy $\text{NP} \subseteq \text{NO}_2^p \subseteq S_2^p$ and $\text{coNP} \subseteq \text{YO}_2^p \subseteq S_2^p$ [CR06]. The formal definitions of these classes are as follows:

Definition 3.11 ([CR06]) 1. NO_2^p is the class of all languages L for which there exist a polynomial-time predicate $R(\cdot, \cdot, \cdot)$ and a polynomial $p(\cdot)$ such that the following holds: For all $n \in \mathbb{N}$, there exists a string z^* with $|z^*| = p(n)$ such that for all $x \in \Sigma^n$,

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

2. YO_2^p is the class of all languages L for which there exist a polynomial-time predicate $R(\cdot, \cdot, \cdot)$ and a polynomial $p(\cdot)$ such that the following holds: For all $n \in \mathbb{N}$, there exists a string y^* with $|y^*| = p(n)$ such that for all $x \in \Sigma^n$,

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

Thus from the aforementioned results of [FPS08] and [CR06], we get the following result:

Corollary 3.12 $\text{P}_{tt}^{\text{NP}[2]} \subseteq \text{ZPP}^{\text{NP}[1]} \implies \text{PH} = S_2^p = \text{NO}_2^p \cap \text{YO}_2^p$.

In a recent work, Chang and Purini [CP07] proved that if the NP machine hypothesis holds, then $\text{P}_{tt}^{\text{NP}[2]} \subseteq \text{P}^{\text{NP}[1]}$ 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 “ $\text{P}_{tt}^{\text{NP}[2]} \subseteq \text{P}^{\text{NP}[1]}$ 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 string. Thus, they proved 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 3.4 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 proving “if the NP machine hypothesis holds, then $\text{P}_{tt}^{\text{NP}[2]} \subseteq \text{ZPP}^{\text{NP}[1]}$ implies $\text{PH} = \text{NP}$.”

4 Conclusion and Open Problems

We obtained solutions to the 1-versus-2 queries problem for hypotheses weaker than the previously considered hypotheses $\text{P}_{tt}^{\Sigma_k^p[2]} \subseteq \text{P}^{\Sigma_k^p[1]}$. We showed 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 for 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 for 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 for 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 for the hypothesis $P_{tt}^{NP[m+1]} \subseteq P_{tt}^{NP[m]}$ but also for the weaker hypothesis $P_{tt}^{NP[m+1]} \subseteq ZPP_{tt}^{NP[m]}$. All these open questions may also be worth looking at when we use Turing reductions in place of truth-table reductions (for instance, what happens if $P^{NP[m+1]} \subseteq ZPP^{NP[m]}$, for some nonconstant $m \geq 2$). 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 $1/2$ (e.g., $1/4$).

Acknowledgment We thank the anonymous (conference and journal) referees for helpful suggestions and insightful comments.

References

- [ABG03] A. Amir, R. Beigel, and W. Gasarch. Some connections between bounded query classes and non-uniform complexity. *Information and Computation*, 186(1):104–139, 2003.
- [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. $S_2^p \subseteq ZPP^{NP}$. *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. Chakaravathy. On zero error algorithms having oracle access to one query. *Journal of Combinatorial Optimization*, 11(2):189–202, 2006.
- [CCHO05] J. Cai, V. Chakaravathy, L. Hemaspaandra, and M. Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Information and Computation*, 198(1):1–23, 2005.
- [CGH⁺88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, 1988.
- [CGH⁺89] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy II: Applications. *SIAM Journal on Computing*, 18(1):95–111, 1989.
- [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. Chakaravathy 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.
- [FIKU05] L. Fortnow, R. Impagliazzo, V. Kabanets, and C. Umans. On the complexity of succinct zero-sum games. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, pages 323–332, 2005. To appear in *Computational Complexity*.
- [FPS08] 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*, 74(3):358–363, 2008.
- [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.

- [HJV93] L. Hemaspaandra, S. Jain, and N. Vereshchagin. Banishing robust Turing completeness. *International Journal of Foundations of Computer Science*, 4(3):245–265, 1993.
- [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.
- [Sip82] M. Sipser. On relativization and the existence of complete sets. In *Proceedings of the 9th International Colloquium on Automata, Languages, and Programming*, pages 523–531. Springer-Verlag *Lecture Notes in Computer Science #140*, 1982.
- [Tri07] R. Tripathi. The 1-versus-2 queries problem revisited. In *Proceedings of the 18th International Symposium on Algorithms and Computation*, pages 137–147. Springer-Verlag *Lecture Notes in Computer Science #4835*, December 2007.
- [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(3):287–300, 1983.