Amplifying $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ and the Two Queries Problem

Richard Chang[†] Suresh Purini[†]

University of Maryland Baltimore County

Abstract

*This paper shows a complete upward collapse in the Polynomial Hierarchy (*PH*) if for* ZPP*, two queries to a* SAT *oracle is equivalent to one query. That is,*

$$\mathrm{ZPP}^{\mathrm{SAT}[1]} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]} \Longrightarrow \mathrm{ZPP}^{\mathrm{SAT}[1]} = \mathrm{PH}.$$

These ZPP *machines are required to succeed with probability at least $1/2 + 1/p(n)$ on inputs of length n for some polynomial $p(n)$. This result builds upon recent work by Tripathi [16] who showed a collapse of* PH *to* $\mathrm{S}_2^{\mathrm{P}}$*. The use of the probability bound of $1/2 + 1/p(n)$ is justified in part by showing that this bound can be amplified to $1 - 2^{-n^k}$ for $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ computations. This paper also shows that in the deterministic case,*

$$\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]} \Longrightarrow \mathrm{PH} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}$$

where the $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ machine achieves a probability of success of $1/2 - 2^{-n^k}$.

1 Introduction

The two queries problem has been studied extensively, beginning with Kadin [14] who showed that $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$ implies that the Polynomial Hierarchy (PH) collapses to the Σ_3^{P} level. Subsequent results [1, 8, 17, 18] brought the collapse further down, to within Δ_3^{P}. A breakthrough in the proof techniques came when Hemaspaandra, Hemaspaandra and Hempel [13] showed that if the queries were made to a Σ_3^{P} oracle (instead of SAT), then $\mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}$ which is a *downward collapse*. Buhrman and Fortnow [2] improved this technique and made it work for queries to a Σ_2^{P} oracle. Fortnow, Pavan and Sengupta [11] then showed that

$$\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]} \Longrightarrow \mathrm{PH} \subseteq \mathrm{S}_2^{\mathrm{P}}$$

which finally brought the collapse of PH below the Σ_2^{P} level.

One interesting thing about the class $\mathrm{S}_2^{\mathrm{P}}$ is its relationship to $\mathrm{ZPP}^{\mathrm{SAT}}$ [3, 4]:

$$\mathrm{ZPP}^{\mathrm{SAT}[1]} \subseteq \mathrm{S}_2^{\mathrm{P}} \subseteq \mathrm{ZPP}^{\mathrm{SAT}}.$$

Since Buhrman and Fortnow [2] also showed that

$$\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]} \Longrightarrow \mathrm{P}^{\mathrm{SAT}} \subseteq \mathrm{P}^{\mathrm{SAT}[1]},$$

we have the collapse

$$\begin{aligned}&\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]} \Longrightarrow \\ &\quad \mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]} \subseteq \mathrm{ZPP}^{\mathrm{SAT}\|[2]} \\ &\quad = \mathrm{S}_2^{\mathrm{P}} = \mathrm{ZPP}^{\mathrm{SAT}} = \mathrm{PH}.\end{aligned}$$

Note that $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$ does not immediately imply that $\mathrm{ZPP}^{\mathrm{SAT}[1]} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]}$ because a ZPP machine can accept, reject or output "don't know". It takes two queries to SAT to cover all three possibilities. However, this collapse is tantalizingly close to a complete upward collapse of PH down to $\mathrm{P}^{\mathrm{SAT}[1]}$. Indeed, Chang and Purini [10] showed that under the NP Machine Hypothesis, $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$ implies that $\mathrm{PH} = \mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{NP}$. This gives us some hope of proving a complete upward collapse without the additional assumption of the NP Machine Hypothesis.

Recently, Tripathi [16] considered the two queries problem in the ZPP setting. He extended the result of Fortnow, Pavan and Sengupta [11] and showed that

$$\mathrm{ZPP}^{\mathrm{SAT}[1]} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]} \Longrightarrow \mathrm{PH} \subseteq \mathrm{S}_2^{\mathrm{P}}.$$

Here, the ZPP machines are required to achieve a probability of success of $1/2 + 1/poly$. In this paper, we build on Tripathi's result and show a complete upward collapse for ZPP:

$$\mathrm{ZPP}^{\mathrm{SAT}[1]} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]} \Longrightarrow \mathrm{PH} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}.$$

Thus, we are able to prove in the ZPP setting what has not been achieved in the deterministic setting.

These results require the ZPP machines to have a probability of success of at least $1/2 + 1/poly$. Without

[†]Address: Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA. Email: chang@umbc.edu, suresh1@umbc.edu.

any oracle queries, ZPP with $1/poly$ probability is equivalent to ZPP with $1-1/exp$ probability by amplification. However, amplification is difficult for $\mathrm{ZPP}^{\mathrm{SAT}[k]}$ because directly simulating a $\mathrm{ZPP}^{\mathrm{SAT}[k]}$ machine t times uses tk queries. Tripathi points out that the same $1/2+1/poly$ bound was used by Cai and Chakaravarthy [4] to show that $\mathrm{BPP} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}$ and $\mathrm{ZPP}^{\mathrm{SAT}[1]} \subseteq \mathrm{S}_2^{\mathrm{P}}$.

On the other hand, $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ with just $1/poly$ probability of success is enough to collapse PH:

$$\mathrm{P}^{\mathrm{SAT}\|[2]} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/poly} \Longrightarrow \mathrm{PH} \subseteq \Sigma_3^{\mathrm{P}}.$$

This is because $\mathrm{P}^{\mathrm{SAT}\|[2]} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/poly}$ induces an $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction from $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ to $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$ with probability $1/poly$, which is enough to collapse PH [9, 15]. (We use $\mathrm{ZPP}^{\mathrm{SAT}[q(n)]}_{\alpha}$ to denote languages recognized by $\mathrm{ZPP}^{\mathrm{SAT}[q(n)]}$ machines which achieve a success probability of at least α when the probability bound is not clear from context.)

So, is there a correct probability bound to consider for $\mathrm{ZPP}^{\mathrm{SAT}[1]}$? Is there a natural choice? We are not able to fully answer these questions. However, we are able to narrow down the choices, because it turns out that you *can* amplify $\mathrm{ZPP}^{\mathrm{SAT}[1]}$. We show that:

$$\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/poly} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/4} \quad \text{and}$$
$$\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp}.$$

Without these amplifications, it might be the case that $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/n^2}$ differs from $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/n^3}$, which would make the definitions of these complexity classes highly non-robust. The difficulty in these constructions is finding a way to simulate several paths of a $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ computation without making additional queries. Here we rely on the fact that SAT and $\overline{\mathrm{SAT}}$ both have ORs. I.e.,

$$F_1 \vee F_2 \vee \cdots \vee F_t \in \mathrm{SAT} \iff \exists i,\ F_i \in \mathrm{SAT}$$
$$F_1 \wedge F_2 \wedge \cdots \wedge F_t \in \overline{\mathrm{SAT}} \iff \exists i,\ F_i \in \overline{\mathrm{SAT}}.$$

(In general, we assume that formulas do not share any variables.) Thus, these techniques would not extend to $\mathrm{ZPP}^{\mathrm{SAT}[k]}$, for $k \geq 2$, since they would require $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ to have ORs.

Nevertheless, these amplifications shed some light on the two queries problem in the deterministic setting. We can show that

$$\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]} \Longrightarrow$$
$$\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}}$$
$$\subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$$
$$\subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2-1/exp} = \mathrm{ZPP}^{\mathrm{SAT}} = \mathrm{PH}.$$

Thus, the two "gaps" in the upward collapse can be viewed as gaps in the probabilistic amplification of $\mathrm{ZPP}^{\mathrm{SAT}[1]}$. One gap occurs at

$$\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp}.$$

Can we use the polynomially more queries in $\mathrm{P}^{\mathrm{SAT}}$ to offset the ability of $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp}$ to output "don't know" very infrequently? This first gap can also be viewed as a question about derandomizing $\mathrm{ZPP}^{\mathrm{SAT}[1]}$. The second gap occurs at

$$\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2-1/exp} = \mathrm{PH}.$$

Is it possible to amplify a $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ computation with less than 1/2 probability of success to $1/2+1/poly$? Can this be done with the assumption that $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$? Answers to these questions would lead to the final resolution of the two queries problem.

The rest of the paper is organized as follows. In Section 2, we provide the usual definitions and discuss the classification of the 1-query trees in a $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ computation. In Section 3, we show the amplification of $\mathrm{ZPP}^{\mathrm{SAT}[1]}$. In Section 4, we prove that PH collapses to $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ if $\mathrm{ZPP}^{\mathrm{SAT}[1]} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]}$. Then in Section 5, we show the ramifications of amplifying $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ to the two queries problem in the deterministic setting. Finally, we discuss the limits of amplification in Section 6 and pose some open problems in Section 7.

2 Preliminaries

Definition 1 Let $q(n)$ be a polynomial-time computable function and X be any language. We use $\mathrm{P}^{X[q(n)]}$ to denote the class of languages recognized by deterministic polynomial-time Turing machines which make at most $q(n)$ serial queries (a.k.a. adaptive queries) to the oracle X on inputs of length n. When the queries are made in parallel (non-adaptively), we use the notation $\mathrm{P}^{X\|[q(n)]}$. Also, when the machines are allowed any polynomial number of queries, we simply use $\mathrm{P}^{\mathrm{SAT}}$ and $\mathrm{P}^{\mathrm{SAT}\|}$.

Definition 2 We use $\mathrm{ZPP}^{X[q(n)]}_{\alpha}$ to denote the class of languages recognized by a ZPP machine with success probability α which makes at most $q(n)$ serial queries to the oracle X on inputs of length n. Note that ZPP machines can output `acc`, `rej` or `dk` (for "don't know") but are never allowed to give an incorrect output. Thus, if M is a $\mathrm{ZPP}^{X[q(n)]}_{\alpha}$ machine and $L = L(M)$ then

$x \in L \Longrightarrow M(x)$ outputs `acc` or `dk` on all paths

$x \notin L \Longrightarrow M(x)$ outputs `rej` or `dk` on all paths.

Furthermore,

$x \in L \Longrightarrow \mathrm{Prob}[M(x)$ outputs `acc`$] \geq \alpha$

$x \notin L \Longrightarrow \mathrm{Prob}[M(x)$ outputs `rej`$] \geq \alpha$.

We use $\mathrm{ZPP}_{\alpha}^{X\|[q(n)]}$ to denote the analogous class of languages recognized by ZPP machines that make parallel queries to X. When the ZPP machines are allowed any polynomial number of queries, we drop the query bound $q(n)$ from our notation. We also drop the probability bound α when it is clear from context or does not matter.

We assume that the reader is familiar with the usual probabilistic complexity classes such as RP and BPP. We also assume familiarity with the use of Chernoff bounds to amplify BPP computations.

Since $\mathrm{ZPP}^{\mathrm{SAT}}$ and $\mathrm{ZPP}^{\mathrm{SAT}\|}$ are allowed polynomially many queries to SAT, these computations can be amplified in the usual way. Thus,

$$\mathrm{ZPP}^{\mathrm{SAT}}_{1/poly} = \mathrm{ZPP}^{\mathrm{SAT}}_{1-1/exp} \text{ and}$$
$$\mathrm{ZPP}^{\mathrm{SAT}\|}_{1/poly} = \mathrm{ZPP}^{\mathrm{SAT}\|}_{1-1/exp}.$$

Here we use $1/poly$ to denote $1/p(n)$ for some polynomial $p(n)$ and $1/exp$ to denote 2^{-n^k} for some constant $k \geq 1$. Using the usual census trick [12], one can show that for any α

$$\mathrm{ZPP}^{\mathrm{SAT}\|}_{\alpha} = \mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}_{\alpha}.$$

Thus, $\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}$ can also be amplified:

$$\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}_{1/poly} = \mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}_{1-1/exp}.$$

We think of a $\mathrm{ZPP}^{\mathrm{SAT}[q(n)]}$ computation as proceeding in two stages. First the ZPP machine makes all of its random moves. Then, at the end of each random path, the ZPP machine asks $q(n)$ queries to SAT forming an *oracle query tree*. At each node of the tree, the machine asks SAT whether some formula $\phi \in \mathrm{SAT}$. If the oracle answers no, the computation proceeds to the left subtree of the oracle query tree. If the oracle answers yes, the right subtree is taken.

For $\mathrm{ZPP}^{\mathrm{SAT}[1]}$, the oracle query tree at the end of each random path makes only 1 query. We classify these 1-query trees into 6 types. (See Figure 1.) The three types of 1-query trees not shown produce the same output regardless of the outcome of the oracle query (e.g., the machine outputs `acc` when the oracle says no and when the oracle says yes). Such trees can be converted to one of the 6 types without changing the behavior of the $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ machine. For convenience, we will also say that a random path in the $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ computation has Type X if the 1-query tree at the end of the random path has Type X.

Definition 3

$$\mathrm{SAT}\wedge\overline{\mathrm{SAT}} = \{(F,G) \mid F \in \mathrm{SAT} \text{ and } G \in \overline{\mathrm{SAT}}\}.$$
$$\overline{\mathrm{SAT}}\vee\mathrm{SAT} = \{(F,G) \mid F \in \overline{\mathrm{SAT}} \text{ or } G \in \mathrm{SAT}\}.$$

Clearly, $\mathrm{SAT}\wedge\overline{\mathrm{SAT}}$ and $\overline{\mathrm{SAT}}\vee\mathrm{SAT}$ are both languages in $\mathrm{P}^{\mathrm{SAT}\|[2]}$. These two languages also have a special role in the analysis of bounded query classes.

Given an oracle query tree T for a $\mathrm{P}^{\mathrm{SAT}[q(n)]}$ computation, the *true path* in the tree is the path taken using the replies from the SAT oracle. Given a path π and an oracle query tree T, the function $\textsc{IsTruePath}(T,\pi)$ returns a pair of formulas (F,G) such that π is the true path in T if and only if $(F,G) \in \mathrm{SAT}\wedge\overline{\mathrm{SAT}}$. Here, F is simply the conjunction of all the queries on the path π that π assumes is satisfiable (the *positive* queries) and G is the disjunction of all the queries that π assumes is unsatisfiable (the *negative* queries). If π is indeed the true path, then all of its positive queries must be satisfiable and all of its negative queries must be unsatisfiable.

Definition 4 Let L be any language. We define $\mathrm{OR}(L)$ and $\mathrm{AND}(L)$ as follows:

$$\mathrm{OR}(L) = \{\ \langle x_1,\ldots,x_t\rangle \mid \text{for some } i,\ x_i \in L\ \}.$$
$$\mathrm{AND}(L) = \{\ \langle x_1,\ldots,x_t\rangle \mid \text{for all } i,\ x_i \in L\ \}.$$

If $\mathrm{OR}(L) \leq^{\mathrm{P}}_{\mathrm{m}} L$ via a polynomial-time function f, we say that L has ORs and call f the OR function. Similarly, if $\mathrm{AND}(L) \leq^{\mathrm{P}}_{\mathrm{m}} L$ via f, we say that L has ANDs and call f the AND function.

Clearly, both SAT and $\overline{\mathrm{SAT}}$ have ORs and ANDs. Chang and Kadin [7] observed that $\mathrm{SAT}\wedge\overline{\mathrm{SAT}}$ has ANDs but does not have ORs unless PH collapses. Similarly, $\overline{\mathrm{SAT}}\vee\mathrm{SAT}$ has ORs but not ANDs unless PH collapses.

Now, suppose that $\mathrm{SAT}\wedge\overline{\mathrm{SAT}} \leq^{\mathrm{P}}_{\mathrm{m}} \overline{\mathrm{SAT}}\vee\mathrm{SAT}$. Using the standard hard/easy argument, it follows that PH collapses [14]. Chang, Rohatgi and Kadin [9] showed that the hard/easy argument can be generalized to work for $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reductions with probability $1/poly$ (defined below).

$$\mathrm{SAT}\wedge\overline{\mathrm{SAT}} \leq^{\mathrm{rp}}_{\mathrm{m}} \overline{\mathrm{SAT}}\vee\mathrm{SAT} \text{ with prob. } 1/poly$$
$$\Longrightarrow \mathrm{PH} \subseteq \Sigma^{\mathrm{P}}_3.$$

The proof exploits the fact that $\overline{\mathrm{SAT}}\vee\mathrm{SAT}$ has ORs since the OR function can be used to combine the output of polynomially many outputs of the $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction. Thus, the success probability of the $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction can be amplified from $1/poly$ to $1-1/exp$.

Definition 5 We say that a language A $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduces to B with probability α via a randomized polynomial-time function f if

$$x \in A \Longrightarrow \mathrm{Prob}[f(x) \in B] \geq \alpha$$
$$x \notin A \Longrightarrow f(x) \notin B.$$

For the results in this paper, it is the hypothesis that $\mathrm{ZPP}^{\mathrm{SAT}[1]} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]}$ which gives us an $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction

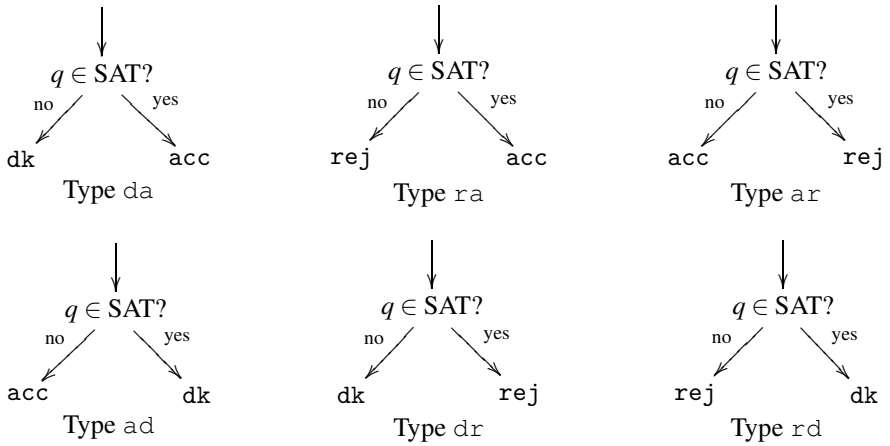

Figure 1. Six types of $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ 1-query trees.

from $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ to $\overline{\mathrm{SAT}} \vee \mathrm{SAT}$. (N.B.: we get an $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reduction and *not* a $\leq_{\mathrm{m}}^{\mathrm{zpp}}$-reduction.) This $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reduction will allow us to show in Section 4 that $\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]} \subseteq \mathrm{ZPP}^{\mathrm{SAT}\|[2]}_{1/2+1/poly}$ if for ZPP two queries to SAT is equivalent to one.

3 Amplifying $\mathrm{ZPP}^{\mathrm{SAT}[1]}$

Theorem 6 $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/poly} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/4-1/exp}$.

Proof: Let $L \in \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/poly}$ via machine M. We say that $M(x)$ produces a definitive output on a random path if it outputs either `acc` or `rej` on that path. Thus, for some polynomial $p(n)$, $M(x)$ produces definitive output on $1/p(n)$ of the random paths, where $n = |x|$. Now, choose $t(n)$ to be a polynomial large enough so that $(1-1/p(n))^{t(n)} \leq 2^{-n}$. Consider a set S of $t(n)$ random paths of $M(x)$ chosen randomly. Then, with probability at least $1-2^{-n}$, $M(x)$ produces definitive output on at least one of the paths in S. Our new machine M' produces definitive output on at least $1/4-1/exp$ of its paths. We will have a special case when S contains a Type `ra` or Type `ar` path, because M always produces a definitive output on these paths.

$M'(x)$:

1. Randomly sample $t(n)$ random paths of $M(x)$. Call this set of paths S.

2. If S contains a Type `ra` or Type `ar` path, simulate $M(x)$ along the first such path and produce the same output as M.

3. Let $Q_{\mathtt{da}}$, $Q_{\mathtt{ad}}$, $Q_{\mathtt{dr}}$ and $Q_{\mathtt{rd}}$ be respectively the queries asked on Type `da`, Type `ad`, Type `dr` and Type `rd` paths in S. Construct the formulas:

$$\phi_{\mathtt{da}} = \bigvee_{q \in Q_{\mathtt{da}}} q, \quad \phi_{\mathtt{ad}} = \bigwedge_{q \in Q_{\mathtt{ad}}} q,$$

$$\phi_{\mathtt{dr}} = \bigvee_{q \in Q_{\mathtt{dr}}} q, \quad \phi_{\mathtt{rd}} = \bigwedge_{q \in Q_{\mathtt{rd}}} q.$$

   As usual we assume that none of the formulas share variables.

4. Choose one of the following 4 steps with equal probability.

   (a) Ask the SAT oracle if $\phi_{\mathtt{da}} \in \mathrm{SAT}$.
   If yes, output `acc`, otherwise output `dk`.

   (b) Ask the SAT oracle if $\phi_{\mathtt{ad}} \in \mathrm{SAT}$.
   If not, output `acc`, otherwise output `dk`.

   (c) Ask the SAT oracle if $\phi_{\mathtt{dr}} \in \mathrm{SAT}$.
   If yes, output `rej`, otherwise output `dk`.

   (d) Ask the SAT oracle if $\phi_{\mathtt{rd}} \in \mathrm{SAT}$.
   If not, output `rej`, otherwise output `dk`.

Clearly, $M'(x)$ uses only 1 query to SAT. Also, $M'(x)$ only outputs `acc` or `rej` when it has confirmed that $M(x)$ has done the same. For example, if $M'(x)$ outputs `acc` in Step 4b, then it has confirmed that one of the queries on a Type `ad` path is unsatisfiable, which means $M(x)$ output `acc` on that path. Thus, $M'(x)$ never produces incorrect output. Finally, if S contains a path where $M(x)$ produces definitive output but S does not have any Type `ra` or Type `ar` paths, then $M(x)$ must do

so on a Type da, ad, dr or rd path. So, $M'(x)$ has a 1/4 probability of picking the right formula in Step 4. Thus, $M'(x)$ outputs acc or rej with probability at least $1/4 \cdot (1-2^{-n}) = 1/4 - 1/exp$. □

In the proof above, we can bump up the probability of $M'(x)$ slightly, by guessing a satisfying assignment for the queries in Type da and Type dr paths before proceeding to Step 4. Although the probability of finding a satisfying assignment can be as low as 2^{-m} where m is the number of variables in the queries of $M(x)$, it allows $M'(x)$ to output acc or rej without asking any queries. Thus, we can lower the probability of choosing $\phi_{\mathtt{da}}$ and $\phi_{\mathtt{dr}}$ to $1/4 - 1/exp$ and raise the probability of choosing $\phi_{\mathtt{ad}}$ and $\phi_{\mathtt{rd}}$ to $1/4 + 1/exp$, which pushes the overall probability of success above $1/4$. Intuitively, in the cases where the only paths of S where $M(x)$ produces definitive output are of Type da or Type dr, $M'(x)$ has a $2^{-m} + 1/4 - 1/exp > 1/4$ probability of success. We omit the formal calculations here, but note that $t(n)$ must be chosen carefully.

Corollary 7 $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/poly} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/4}$.

Theorem 8 $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp}$.

Proof: Let M be a $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$ machine for some language L. For some polynomial $p(n)$, M succeeds with probability $1/2 + 1/p(n)$ on inputs of length n. We construct a new $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ machine M' that succeeds with probability $1 - 1/exp$. Steps 1 to 3 of M' is identical to those in Theorem 6 except for the choice of $t(n)$. In the new Step 4, we choose $\phi_{\mathtt{da}}$, $\phi_{\mathtt{ad}}$, $\phi_{\mathtt{dr}}$ or $\phi_{\mathtt{rd}}$ as follows:

4. Let $t_{\mathtt{da}}$, $t_{\mathtt{ad}}$, $t_{\mathtt{dr}}$ and $t_{\mathtt{rd}}$ be respectively the number of Type da, ad, dr and rd paths in S.
   (a) If $t_{\mathtt{da}} + t_{\mathtt{ad}} \geq t_{\mathtt{dr}} + t_{\mathtt{rd}}$ and $t_{\mathtt{da}} \geq t_{\mathtt{ad}}$ then ask if $\phi_{\mathtt{da}} \in$ SAT. If yes, output acc, otherwise output dk.
   (b) If $t_{\mathtt{da}} + t_{\mathtt{ad}} \geq t_{\mathtt{dr}} + t_{\mathtt{rd}}$ and $t_{\mathtt{da}} < t_{\mathtt{ad}}$ then ask if $\phi_{\mathtt{ad}} \in$ SAT. If not, output acc, otherwise output dk.
   (c) If $t_{\mathtt{da}} + t_{\mathtt{ad}} < t_{\mathtt{dr}} + t_{\mathtt{rd}}$ and $t_{\mathtt{dr}} \geq t_{\mathtt{rd}}$ then ask if $\phi_{\mathtt{dr}} \in$ SAT. If yes, output rej, otherwise output dk.
   (d) If $t_{\mathtt{da}} + t_{\mathtt{ad}} < t_{\mathtt{dr}} + t_{\mathtt{rd}}$ and $t_{\mathtt{dr}} < t_{\mathtt{rd}}$ then ask if $\phi_{\mathtt{rd}} \in$ SAT. If not, output rej, otherwise output dk.

Note that the four cases in Step 4 partition all possibilities. As before, M' uses only one query to SAT and never produces incorrect output. Thus, we only need to check that M' succeeds with probability $1 - 1/exp$.

Suppose that $x \in L$. Then, M' might output acc or dk. We need to show that M' outputs dk on at most $1/exp$ of the random paths. So, let A be the set of random paths where $M(x)$ outputs acc and let B be the paths where $M(x)$ outputs dk. (Since $x \in L$, $M(x)$ cannot output rej.)

Also, since $x \in L$, at least $1/2 + 1/poly$ of the paths of $M(x)$ are in A. Using Chernoff bounds, for any constant k, we can choose $t(n)$ to be a polynomial large enough such that

$$\mathrm{Prob}_S[\ |S \cap A| \leq |S \cap B|\ ] \leq 2^{-n^k}.$$

When $|S \cap A| \leq |S \cap B|$ we say that S is *bad*. It suffices to show that S is bad whenever $M'(x)$ outputs dk.

So, suppose $M'(x)$ outputs dk for a fixed S. Since $M'(x)$ never outputs dk in Step 2, it can only do so in Step 4. Thus, S does not have any Type ra or Type ar paths. In general, A contains only Type da, ra, ar and ad paths and B contains only Type da, ad, dr and rd paths. So, $t_{\mathtt{dr}} + t_{\mathtt{rd}} \leq |S \cap B|$. Also, since there are no Type ra and Type ar paths in S, $|S \cap A| \leq t_{\mathtt{da}} + t_{\mathtt{ad}}$.

Now, suppose that $M'(x)$ outputs dk in Step 4c or in Step 4d. Then, $t_{\mathtt{da}} + t_{\mathtt{ad}} < t_{\mathtt{dr}} + t_{\mathtt{rd}}$. Therefore,

$$|S \cap A| \leq t_{\mathtt{da}} + t_{\mathtt{ad}} < t_{\mathtt{dr}} + t_{\mathtt{rd}} \leq |S \cap B|$$

and S is bad.

Next, suppose that $M'(x)$ outputs dk in Step 4b. Then, $\phi_{\mathtt{ad}} \in$ SAT. That means every query $q \in Q_{\mathtt{ad}}$ is satisfiable, so none of the Type ad paths in S are in A. Hence, all of the Type ad paths in S are in B, so $t_{\mathtt{ad}} \leq |S \cap B|$. Furthermore, since $S \cap A$ does not have any Type ad paths, the only remaining type in $S \cap A$ is Type da. Thus, $|S \cap A| \leq t_{\mathtt{da}}$. For Step 4b to be executed, $t_{\mathtt{da}} < t_{\mathtt{ad}}$. Putting it all together, we have $|S \cap A| \leq t_{\mathtt{da}} < t_{\mathtt{ad}} \leq |S \cap B|$ and S is bad.

Finally, suppose that $M'(x)$ outputs dk in Step 4a. Then, $\phi_{\mathtt{da}} \notin$ SAT. That means none of the Type da paths in S output acc. So, $S \cap A$ has no Type da paths. Then all of the Type da paths in S are in B, which implies $t_{\mathtt{da}} \leq |S \cap B|$. Since $S \cap A$ has no Type da paths, the only remaining type in $S \cap A$ is Type ad. Then, $|S \cap A| \leq t_{\mathtt{ad}}$. For Step 4a to be executed, $t_{\mathtt{ad}} \leq t_{\mathtt{da}}$. Therefore, we have $|S \cap A| \leq t_{\mathtt{ad}} \leq t_{\mathtt{da}} \leq |S \cap B|$ and S is bad.

We have checked that when $x \in L$, in every case where $M'(x)$ outputs dk, S is bad. The proof for $x \notin L$ is symmetrical. Since S being bad occurs with $1/exp$ probability, $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp}$. □

The statement $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp}$ only requires that for every polynomial $p(n)$, that $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/p(n)}$

is contained in $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-2^{-n^k}}$ for *some* constant k. The proof above actually shows containment for *all* constants k.

Corollary 9 For any polynomial $p(n)$ and for all constants k, $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/p(n)} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-2^{-n^k}}$.

4 Two Queries for ZPP

Our main theorem, Theorem 16, shows that for ZPP machines with success probability $1/2+1/poly$, if $\mathrm{ZPP}^{\mathrm{SAT}[1]} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]}$ then we have a complete upward collapse of PH down to $\mathrm{ZPP}^{\mathrm{SAT}[1]}$. We will prove this upward collapse in two steps. First, we show in Theorem 11 that $\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}$ collapses to $\mathrm{ZPP}^{\mathrm{SAT}\|[2]}$. Then in Theorem 15 we show that $\mathrm{ZPP}^{\mathrm{SAT}}$ collapses to $\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}$.

Lemma 10 If $\mathrm{ZPP}^{\mathrm{SAT}\|[2]}_{1/2+1/poly} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$, then

$$\mathrm{SAT}\wedge\overline{\mathrm{SAT}} \leq^{\mathrm{rp}}_{\mathrm{m}} \overline{\mathrm{SAT}}\vee\mathrm{SAT}$$

with probability $1-1/exp$.

Proof: Since $\mathrm{SAT}\wedge\overline{\mathrm{SAT}}$ can be recognized with two parallel queries to SAT, the hypothesis of the lemma gives us $\mathrm{SAT}\wedge\overline{\mathrm{SAT}} \in \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$ via some machine M. Our first $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction h_1 guesses a random path of $M(F,G)$ and reduces the 1-query tree at the end of the path to $\overline{\mathrm{SAT}}\vee\mathrm{SAT}$:

$h_1(F,G)$:

1. Guess a random path y of $M(x)$. Let q be the query to SAT asked on y.
2. Consider the type of the random path y:

   Types `da` and `ra`: output $(\mathtt{true}, q)$.

   Types `ar` and `ad`: output $(q, \mathtt{false})$.

   Types `dr` and `rd`: output $(\mathtt{true}, \mathtt{false})$.

It is easy to check that h_1 is an $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction from $\mathrm{SAT}\wedge\overline{\mathrm{SAT}}$ to $\overline{\mathrm{SAT}}\vee\mathrm{SAT}$ with probability $1/2+1/poly$. We can amplify the probability of h_1 by combining the output of h_1 on n^k random paths using the OR function for $\overline{\mathrm{SAT}}\vee\mathrm{SAT}$. The resulting $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction h succeeds with probability at least $1-2^{-n^k}$. □

Theorem 11 If $\mathrm{ZPP}^{\mathrm{SAT}\|[2]}_{1/2+1/poly} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$ then

$$\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]} \subseteq \mathrm{ZPP}^{\mathrm{SAT}\|[2]}_{1-1/exp}.$$

Proof: Since $\overline{\mathrm{SAT}}\vee\mathrm{SAT}$ can be recognized using two queries to a SAT oracle, the hypothesis of this theorem places $\overline{\mathrm{SAT}}\vee\mathrm{SAT}$ in $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$. Furthermore, since $\overline{\mathrm{SAT}}\vee\mathrm{SAT}$ has ORs, the language $\mathrm{OR}(\overline{\mathrm{SAT}}\vee\mathrm{SAT})$ is also in $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$. Recall that $\mathrm{OR}(\overline{\mathrm{SAT}}\vee\mathrm{SAT})$ is defined as the set of tuples $\langle(\phi_1,\psi_1),\ldots,(\phi_m,\psi_m)\rangle$ such that some pair $(\phi_i,\psi_i) \in \overline{\mathrm{SAT}}\vee\mathrm{SAT}$.

By Corollary 9, we can amplify $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$ and increase the probability of success to $1-1/exp$. So let M_1 be a $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp}$ machine for $\mathrm{OR}(\overline{\mathrm{SAT}}\vee\mathrm{SAT})$ and let h be the $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction from $\mathrm{SAT}\wedge\overline{\mathrm{SAT}}$ to $\overline{\mathrm{SAT}}\vee\mathrm{SAT}$ given by Lemma 10. Then, given m pairs of formulas $(F_1,G_1),(F_2,G_2),\ldots(F_m,G_m)$, we have:

- If there exists i, such that $(F_i,G_i) \in \mathrm{SAT}\wedge\overline{\mathrm{SAT}}$, then $M_1(h(F_1,G_1),\ldots,h(F_m,G_m))$ accepts with probability at least $1-1/exp$.
- If for all i, we have $(F_i,G_i) \notin \mathrm{SAT}\wedge\overline{\mathrm{SAT}}$, then $M_1(h(F_1,G_1),\ldots,h(F_m,G_m))$ rejects with probability at least $1-1/exp$ and outputs `dk` on paths where it does not reject.

The probability here is taken over the joint distribution of the coin tosses of h and of M_1. Note that this is not a $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ computation for $\mathrm{OR}(\overline{\mathrm{SAT}}\vee\mathrm{SAT})$ because there is a small chance that some $(F_i,G_i) \in \mathrm{SAT}\wedge\overline{\mathrm{SAT}}$ but M_1 outputs `rej`. This one-sided error is caused by the one-sided error of the $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction h. This does not matter as long as our final $\mathrm{ZPP}^{\mathrm{SAT}\|[2]}$ machine does not make any errors.

Let L be a language in $\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}$ via some machine M_2. By amplification, we can assume that M_2 succeeds with probability $1-1/exp$. We now construct a $\mathrm{ZPP}^{\mathrm{SAT}\|[2]}_{1-1/exp}$ machine M_3 for L.

$M_3(x)$:

1. Randomly choose a random path y of $M_2(x)$ and let T_y be the $O(\log n)$ query tree on path y.
2. Collect the paths $\pi_1,\ldots,\pi_s$ in T_y where M_2 outputs `acc`. Let $(F_i,G_i) = \textsc{IsTruePath}(T_y,\pi_i)$.
3. Collect the paths $\rho_1,\ldots,\rho_t$ in T_y where M_2 outputs `rej`. Let $(F'_i,G'_i) = \textsc{IsTruePath}(T_y,\rho_i)$.
4. In parallel, simulate the two computations [additional note below]:

   $M_1(h(F_1,G_1),\ldots,h(F_s,G_s))$ and
   $M_1(h(F'_1,G'_1),\ldots,h(F'_t,G'_t))$.

5. If $M_1(h(F_1,G_1),\ldots,h(F_s,G_s))$ outputs `acc`, M_3 outputs `acc`.

6. If $M_1(h(F_1',G_1'),\ldots,h(F_t',G_t'))$ outputs `acc`, M_3 outputs `rej`.

7. Otherwise, M_3 outputs `dk`.

Since the query trees of M_2 have $O(\log n)$ height, Steps 2 and 3 take polynomial time. Also, the only queries to SAT are the two queries asked in Step 4 where M_1 is simulated twice. We need to be careful here and ask the queries from the two simulations in one parallel step. More precisely, pick two random paths, r and r' of M_1, then determine the query q asked by M_1 on path r on input $\langle h(F_1,G_1),\ldots,h(F_s,G_s)\rangle$ and the query q' asked by M_1 on path r' on input $\langle h(F_1',G_1'),\ldots,h(F_t',G_t')\rangle$. Ask in parallel whether q and q' are in SAT. Finally, using the SAT oracle's reply, determine M_1's output on the two inputs.

Now suppose that $x \in L$. Then, the vast majority of the random paths of $M_2(x)$ output `acc`. On these paths, the true path in the oracle query tree must output `acc`. Hence, one of the (F_i,G_i) computed in Step 2 must be in $\mathrm{SAT}\wedge\overline{\mathrm{SAT}}$. Then, $M_3(x)$ will output `acc` in Step 5 with probability at least $1-1/exp$. The probability that $x \in L$ but $M_2(x)$ outputs `dk` is only $1/exp$, so the overall probability that $M_3(x)$ accepts is still $\geq (1-1/exp)\cdot(1-1/exp) = 1-1/exp$.

The case that $x \notin L$ is symmetrical. On random paths where $M_2(x)$ reject, the true path must output `rej` which means that one of the (F_i',G_i') pairs is in $\mathrm{SAT}\wedge\overline{\mathrm{SAT}}$. Then, $M_3(x)$ will output `rej` in Step 6 with probability at least $1-1/exp$.

Finally, because the $\leq_{\mathrm{m}}^{\mathrm{rp}}$-reduction h makes one-sided errors (on the correct side) and because M_1 is a $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ machine that never gives incorrect output, when M_1 outputs `acc` in Step 5 or Step 6, we have a witness for $M_2(x)$ accepting or rejecting. Thus, $M_3(x)$ never outputs `acc` or `rej` incorrectly. Therefore, $L \in \mathrm{ZPP}^{\mathrm{SAT}\|[2]}_{1-1/exp}$. □

Our next theorem will show that if $\mathrm{ZPP}^{\mathrm{SAT}[1]} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]}$ then $\mathrm{ZPP}^{\mathrm{SAT}} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}$. The outline of the proof is similar to the proof in Buhrman and Fortnow [2] that $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$ implies $\mathrm{P}^{\mathrm{SAT}} \subseteq \mathrm{P}^{\mathrm{SAT}[O(\log n)]}$. That is, we define hard and easy strings then use $O(\log n)$ queries and binary search to find the level of the first query on the true path of a $\mathrm{P}^{\mathrm{SAT}}$ computation that is a hard string. Once this level number has been found, it can be incorporated in one more SAT query which allows an NP machine to find this hard string. With a hard string, the NP machine can recognize $\overline{\mathrm{SAT}}$ and thereby simulate the $\mathrm{P}^{\mathrm{SAT}}$ computation to the end.

The complication we have here is that the hard strings we define do not always result in an NP algorithm for $\overline{\mathrm{SAT}}$. We can end up with a BPP algorithm for $\overline{\mathrm{SAT}}$. This BPP algorithm can make two-sided errors, which is problematic because we want to construct a $\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}$ machine that makes no errors. Correcting for the two-sided error of the BPP algorithm is the main innovation in the proof. We now give the formal definition of easy for this proof.

Definition 12 Suppose that $\mathrm{SAT}\wedge\overline{\mathrm{SAT}} \in \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-2^{-n^3}}$ via M_1. For a fixed length n, we say a formula G of length n is *easy* if there exists a formula F, $|F| = n$, and a random path r of $M_1(F,G)$ such that r is Type `da` or Type `ra` and the query q on r is satisfiable.

Note that an NP machine can check whether a given formula G is easy by guessing a formula F, a random path r and a satisfying assignment for the query q on path r of $M_1(F,G)$. Furthermore, since M_1 outputs `acc` in Type `da` and Type `ra` paths when the query is satisfiable, the NP machine has also verified that $(F,G) \in \mathrm{SAT}\wedge\overline{\mathrm{SAT}}$ and, in particular, that $G \in \overline{\mathrm{SAT}}$. Thus, for easy G, unsatisfiability can be verified by an NP machine.

Definition 13 A formula H of length n is *hard* if $H \in \overline{\mathrm{SAT}}$ and H is not easy.

Typically, in hard/easy arguments, a hard string of length n also gives us a way to verify the unsatisfiability of formulas of length n. However, using our definitions of hard and easy, we might only have probabilistic "evidence" of unsatisfiability.

To see this, suppose that $\mathrm{SAT}\wedge\overline{\mathrm{SAT}} \in \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-2^{-n^3}}$ via M_1. Let H be a hard string of length n, let F be some formula we would like to verify is unsatisfiable and let r be a random path in $M_1(F,H)$.

Case 1: Suppose r is a Type `ra` path in $M_1(F,H)$. Since H is hard, the query q on r must be in $\overline{\mathrm{SAT}}$. Thus, M_1 outputs `rej` on r. This verifies that $(F,H) \notin \mathrm{SAT}\wedge\overline{\mathrm{SAT}}$ and we can conclude that $F \in \overline{\mathrm{SAT}}$ (since $H \in \overline{\mathrm{SAT}}$). Thus, a Type `ra` random path in $M_1(F,H)$ witnesses the unsatisfiability of F. (Note that we do not need to check whether $q \in \mathrm{SAT}$ in this case.)

Case 2: Suppose r is a Type `ar` path in $M_1(F,H)$. Let q be the query on r. If $q \in \mathrm{SAT}$, then $M_1(F,H)$ outputs `rej` and $(F,H) \notin \mathrm{SAT}\wedge\overline{\mathrm{SAT}}$. Since $H \in \overline{\mathrm{SAT}}$, we have $F \in \overline{\mathrm{SAT}}$ as well. Conversely, if $q \notin \mathrm{SAT}$, then $M_1(F,H)$ outputs `acc`, which implies that $(F,H) \in \mathrm{SAT}\wedge\overline{\mathrm{SAT}}$ and in particular, $F \in \mathrm{SAT}$. Thus, when we find a Type `ar` path, we get $F \in \overline{\mathrm{SAT}} \iff q \in \mathrm{SAT}$. So, a satisfying assignment for F witnesses $F \in \mathrm{SAT}$ and a satisfying assignment for q witnesses $F \in \overline{\mathrm{SAT}}$.

Case 3: Suppose that r is a Type dr or Type rd path in $M_1(F,H)$. If $F \in \text{SAT}$, then $(F,H) \in \text{SAT} \wedge \overline{\text{SAT}}$ and $M_1(F,H)$ must output acc with probability $1-2^{-n^3}$. Since M_1 cannot output acc on Type dr and rd paths, it is unlikely that $F \in \text{SAT}$ and a randomly chosen r is Type dr or rd:

$$F \in \text{SAT} \Longrightarrow \text{Prob}_r[\, r \text{ is Type dr or rd in } M_1(F,H)\,] \leq 2^{-n^3}. \quad (1)$$

So, r being Type dr or rd is probabilistic "evidence" that $F \in \overline{\text{SAT}}$.

Case 4: Suppose that r is a Type da or Type ad path in $M_1(F,H)$. Here, we have probabilistic evidence that $F \in \text{SAT}$, since $F \in \overline{\text{SAT}}$ means $(F,H) \notin \text{SAT} \wedge \overline{\text{SAT}}$ and M_1 cannot output rej on Type da and ad paths.

$$F \in \overline{\text{SAT}} \Longrightarrow \text{Prob}_r[\, r \text{ is Type da or ad in } M_1(F,H)\,] \leq 2^{-n^3}. \quad (2)$$

Note that Equations 1 and 2 hold regardless of the existence of Type ra and Type ar random paths in $M_1(F,H)$. We might combine Case 3 and 4 into a BPP algorithm for $\overline{\text{SAT}}$. However, such an algorithm will make two-sided errors and cannot be used directly in the construction of a $\text{ZPP}^{\text{SAT}[O(\log n)]}$ machine that must not make any error. We take two measures to correct these potential errors.

First, we identify the random paths r where we have a satisfiable formula F of length n but r is nevertheless Type dr or rd. We call such r bad:

Definition 14 We call a random path r *bad* with respect to a hard string H and a length n if there exists a formula F, $|F| = n$, such that $F \in \text{SAT}$ but r is Type dr or Type rd in $M_1(F,H)$.

There are indeed very few bad r. By Equation 1 and the fact that there are less than 2^n satisfiable formulas of length n

$$\text{Prob}_r\left[\begin{array}{l} \exists F \in \text{SAT}, |F| = n, \\ \text{and } r \text{ is Type dr or rd} \\ \text{in } M_1(F,H) \end{array}\right] \leq 2^{-n^2}. \quad (3)$$

Thus, the probability that a randomly chosen r is *not* bad is at least $1-2^{-n^2}$.

Checking whether a particular r is bad can be done in NP by guessing a formula F with length n, guessing a satisfying assignment for F and checking that r is Type dr or rd. Otherwise, if r is not bad, then we have a guaranteed witness for the unsatisfiability of F:

$$\forall F,\ |F| = n, \left[r \text{ is Type dr or rd in } M_1(F,H) \Longrightarrow F \in \overline{\text{SAT}}\,\right]. \quad (4)$$

Our strategy is to have our $\text{ZPP}^{\text{SAT}[O(\log n)]}$ machine guess an r and ask SAT if r is bad. If r is indeed bad, the machine outputs dk and gives up. Otherwise, it uses r in subsequent queries as a witness for unsatisfiability.

Our second measure involves Case 4. Here an error might occur if $F \in \overline{\text{SAT}}$ but r is Type da or ad in $M_1(F,H)$ which would cause us to infer that $F \in \text{SAT}$. So, we simply have an NP machine guess a satisfying assignment for F whenever we find that r is Type da or ad. The concern here is that if F is actually in $\overline{\text{SAT}}$, then all branches of the NP machine will terminate and the NP machine will not be able carry out more simulations. However, the probability of this event is again very low. Using Equation 2 we have

$$\text{Prob}_r\left[\begin{array}{l} \exists F \in \overline{\text{SAT}}, |F| = n, \\ \text{and } r \text{ is Type da or ad} \\ \text{in } M_1(F,H) \end{array}\right] \leq 2^{-n^2}. \quad (5)$$

Finally, we still have to contend with the issue of finding a hard string H. We look for a hard string in the P^{SAT} oracle query tree T at the end of a ZPP^{SAT} random path. To do this, we ask the NP question:

> Are the first k queries on the true path of T either satisfiable or easy?

The trick here is that we do not have to provide the true path of T. An NP machine N_1 can guess a path from the root of T to a node in level k. For each positive query on the path (i.e., those queries that the path assumes are satisfiable), N_1 can guess a satisfying assignment for the query. For each negative query, N_1 will verify that it is easy (since the assumption is that none of the first k queries are hard). This simultaneously verifies that the path N_1 guessed is an initial segment of the true path and that each of the first k queries is satisfiable or easy. Using N_1 and binary search, the level of the first hard query on the true path can be found using $O(\log n)$ queries to SAT.

Theorem 15 If $\text{ZPP}^{\text{SAT}\|[2]}_{1/2+1/poly} \subseteq \text{ZPP}^{\text{SAT}[1]}_{1/2+1/poly}$ then $\text{ZPP}^{\text{SAT}} \subseteq \text{ZPP}^{\text{SAT}[O(\log n)]}$.

Proof: Let $L \in \text{ZPP}^{\text{SAT}}$. By amplification, we can assume that there is a ZPP^{SAT} machine M_5 that recognizes L with success probability $1-2^{-n^3}$. Furthermore, for notational convenience, assume that all the queries to SAT made by M_5 on inputs of length n have the same length m. By Corollary 9, we can assume that $\text{SAT} \wedge \overline{\text{SAT}} \in \text{ZPP}^{\text{SAT}[1]}_{1-2^{-n^3}}$ via some machine M_1. Now, we can construct a $\text{ZPP}^{\text{SAT}[O(\log n)]}$ machine M_6:

$M_6(x)$:

1. Guess a random path y of $M_5(x)$ and consider the oracle query tree T_y at the end of the random path y. [Note: The tree T_y can be exponential in size, so we do not want to actually construct it.]

2. Guess a random path r for M_1 long enough for formulas of length m.

3. Use binary search and N_1 (as described above) to find the first query on the true path of T_y that is a hard string. [This uses $O(\log n)$ queries to SAT.]

4. If the true path of T_y does not have any hard queries, use 2 more queries to SAT to determine if $M_5(x)$ on random path y output `acc`, `rej` or `dk` at the end of the true path of T_y, then output the same value and terminate.

5. Otherwise, let k be the level in T_y where the first hard query appears on the true path.

6. Construct an NP machine N_2 that on input (r,k) finds the hard string H on level k of the true path and accepts if r is bad with respect to H and m. Ask SAT whether $N_2(r,k)$ accepts. If yes, output `dk` and terminate (because r is bad).

7. Construct an NP machine N_3 that accepts input (x,r,y,k) if $M_5(x)$ on random path y outputs `acc` at the end of the true path for T_y. Similarly, construct an NP machine N_4 that accepts (x,r,y,k) if $M_5(x)$ outputs `rej` on random path y. [Detailed descriptions of N_3 and N_4 are given below.]

8. Ask the SAT oracle whether $N_3(x,r,y,k)$ accepts and whether $N_4(x,r,y,k)$ accepts. If N_3 accepts, output `acc`. If N_4 accepts, output `rej`. Otherwise, when both N_3 and N_4 reject, output `dk`.

It is clear from the description of M_6 that it makes at most $O(\log n)$ queries to SAT. To check that M_6 does not make any errors — i.e., output `acc` when $x \notin L$ or output `rej` when $x \in L$ — we need to examine the NP machines N_3 and N_4 more closely:

$N_3(x,r,y,k)$:

1. Simulate $M_5(x)$ on random path y until the oracle query tree T_y is reached.

2. Assume that the first $k-1$ queries on the true path of T_y are either satisfiable or easy. Guess and verify the first $k-1$ queries of the true path. This recovers the initial $k-1$ segment of the true path.

3. Recover the k-th query on the true path of T_y. Use this query as a hard string H.

4. If more queries remain [on what will turn out to be the true path], let ϕ be the next query.

   (a) If r is Type `ra` in $M_1(\phi,H)$ conclude that $\phi \in \overline{\text{SAT}}$. Repeat Step 4.

   (b) If r is Type `ar` in $M_1(\phi,H)$, let q be the query $M_1(\phi,H)$ asks on r. Conclude that $\phi \in \overline{\text{SAT}} \iff q \in \text{SAT}$. Guess whether $\phi \in \text{SAT}$ or $\phi \in \overline{\text{SAT}}$, then verify by guessing a satisfying assignment for ϕ or for q. Repeat Step 4.

   (c) If r is Type `dr` or `rd` conclude that $\phi \in \overline{\text{SAT}}$. Repeat Step 4.

   (d) If r is Type `da` or `ad`, conclude that $\phi \in \text{SAT}$. Confirm that ϕ is indeed satisfiable by guessing a satisfying assignment for ϕ. Repeat Step 4.

5. Accept if M_5 outputs `acc` on the true path.

The NP machine N_4 does the same thing as N_3 except the last step where N_4 accepts if M_5 outputs `rej`.

Correctness of M_6: We claim that N_3 correctly recovers the true path in T_y as long as r is not bad and k is indeed the level of the first hard query on the true path in T_y. The correctness of the first k steps of the true path is guaranteed by the correctness of k. For queries after the k-th query, if r is Type `ra` or Type `ar`, then correctness is guaranteed by the hardness of H. (See Cases 1 and 2 above.) If N_3 concludes that $\phi \in \overline{\text{SAT}}$ in Step 4c, then ϕ must indeed be in $\overline{\text{SAT}}$ because M_6 has already checked that r is not bad relative to the hard string H and formulas with length m. (See the justification for Equation 4.) If N_3 concludes that $\phi \in \text{SAT}$ in Step 4d, it also finds a satisfying assignment for ϕ.

Thus, if N_3 accepts, then it has verified that M_5 outputs `acc` on random path y. Note that if N_3 fails to find a satisfying assignment for some $\phi \in \overline{\text{SAT}}$ in Step 4d, it could only cause M_6 to output `dk`. Similarly, N_4 accepts only when it has verified that M_5 outputs `rej` on random path y. Thus, M_6 only outputs `acc` or `rej` when it has confirmed that M_5 has done the same. Therefore, M_6 never outputs `acc` when $x \notin L$ and never outputs `rej` when $x \in L$.

Probabilistic Analysis of M_6: Now we can bound the probability that $M_6(x)$ outputs `dk`. First, for all choices of a random path y where $M_5(x)$ outputs `dk`, $M_6(x)$ will also output `dk`. The probability that $M_5(x)$ outputs `dk` is at most 2^{-n^3}. Second, $M_6(x)$ might output `dk` because r is bad, but r is bad with probability at most 2^{-m^2} by Equation 3. Finally, M_6 might output `dk` because all nondeterministic paths of N_3 or N_4 terminated in Step 4d while trying to guess a satisfying assignment for some $\phi \in \overline{\text{SAT}}$. By Equation 5, this event occurs with probability at most 2^{-m^2}. Therefore, the probability that $M_6(x)$ outputs `dk` is no more than $2^{-n^3} + 2^{-m^2} + 2^{-m^2}$. Hence, M_6 succeeds with probability $1 - 1/exp$. □

Theorem 16 If $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]}_{1/2+1/poly}$ then

$$\mathrm{PH} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}.$$

Proof: Tripathi [16] showed that

$$\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]}_{1/2+1/poly} \Longrightarrow \mathrm{PH} \subseteq \mathrm{S}^{\mathrm{P}}_2.$$

By Cai's result [3], $\mathrm{S}^{\mathrm{P}}_2 \subseteq \mathrm{ZPP}^{\mathrm{SAT}}$, so PH collapses to $\mathrm{ZPP}^{\mathrm{SAT}}$. Theorem 15 brings the collapse down $\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}$ which Theorem 11 shows is contained in $\mathrm{ZPP}^{\mathrm{SAT}\|[2]}_{1/2+1/poly}$. Finally, using the hypothesis of the theorem, $\mathrm{PH} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$. □

5 Two Queries for P

Lemma 17 If $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$ then

$$\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2-1/exp}.$$

Proof Sketch: Since $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$ does *not* immediately imply that $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{\alpha} = \mathrm{ZPP}^{\mathrm{SAT}\|[2]}_{\alpha}$, Theorem 16 does not help us. Instead, this proof is similar to the proof of Theorem 11. Here, we exploit the fact that $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$ implies that $\mathrm{SAT} \wedge \overline{\mathrm{SAT}} \equiv^{\mathrm{P}}_{\mathrm{m}} \overline{\mathrm{SAT}} \vee \mathrm{SAT}$. This in turn implies that $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$ has OR's and, more importantly, that there exists a $\mathrm{P}^{\mathrm{SAT}[1]}$ machine M_1 which takes as input $(F_1, G_1), \ldots, (F_n, G_n)$ and accepts if and only if one of the (F_i, G_i) is in $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$.

Let M_2 be a $\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]}$ machine for some language L. By amplification, we can assume that the probability of success for M_2 is $1 - 1/exp$. As in the proof of Theorem 11, we construct an M_3 that simulates M_2 along a random path and collect together paths in the oracle query tree where M_2 outputs `acc` and paths where M_2 output `rej`. Then, we use the IsTruePath function to reduce each path to $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$. Instead of simulating two $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ computations in parallel, we simply guess with 1/2 probability whether M_2 accepts or rejects, and confirm this guess using M_1. Since M_1 is a $\mathrm{P}^{\mathrm{SAT}[1]}$ machine, which is deterministic, there is no error involved in using M_1. The only errors are from guessing whether M_2 accepts or rejects and from M_2 producing output `dk` on $1/exp$ of the paths. Thus, the overall probability of success of M_3 is $(1/2) \cdot (1 - 1/exp) = 1/2 - 1/exp$. □

Note that the "bumping" trick we used in Corollary 7 to prove that $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/poly} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/4}$ does not work here because there is no analog of "guessing a satisfying assignment" for $\mathrm{SAT} \wedge \overline{\mathrm{SAT}}$. As we mentioned in the introduction, the following theorem allows us to interpret the gaps in the collapse of PH when $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$ in terms of gaps in the amplification of $\mathrm{ZPP}^{\mathrm{SAT}[1]}$.

Theorem 18 If $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$ then

$$\begin{aligned}
&\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}} \\
&\subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly} \\
&\subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2-1/exp} = \mathrm{ZPP}^{\mathrm{SAT}} = \mathrm{PH}.
\end{aligned}$$

Proof: We know $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}\|[2]}$ implies $\mathrm{P}^{\mathrm{SAT}[1]} = \mathrm{P}^{\mathrm{SAT}}$ from Buhrman and Fortnow [2]. Fortnow, Pavan and Sengupta [11] give us $\mathrm{PH} \subseteq \mathrm{S}^{\mathrm{P}}_2$. Cai showed $\mathrm{S}^{\mathrm{P}}_2 \subseteq \mathrm{ZPP}^{\mathrm{SAT}}$. Since $\mathrm{P}^{\mathrm{SAT}[O(\log n)]} = \mathrm{P}^{\mathrm{SAT}}$ does imply $\mathrm{ZPP}^{\mathrm{SAT}[O(\log n)]} = \mathrm{ZPP}^{\mathrm{SAT}}$, we just need Lemma 17 to bring PH down to $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2-1/exp}$. Finally, $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$ is Theorem 8 and $\mathrm{P}^{\mathrm{SAT}} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp}$ because $\mathrm{P}^{\mathrm{SAT}[1]} \subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp}$ and $\mathrm{P}^{\mathrm{SAT}} = \mathrm{P}^{\mathrm{SAT}[1]}$. □

6 The Limits of Amplification

We know that we can amplify the success probability for ZPP, $\mathrm{ZPP}^{\mathrm{SAT}}$, $\mathrm{ZPP}^{\mathrm{SAT}\|}$ machines from $1/poly$ to $1 - 1/exp$. But what can we say about amplifying $\mathrm{ZPP}^{\mathrm{SAT}[k]}$ and $\mathrm{ZPP}^{\mathrm{SAT}\|[k]}$ machines, when k is a constant? For $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ machines, we have shown that

$$\begin{aligned}
\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1-1/exp} &= \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly} \\
&\subseteq \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/4} = \mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/poly}.
\end{aligned}$$

This leaves us with a gap between the complexity classes $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/2+1/poly}$ and $\mathrm{ZPP}^{\mathrm{SAT}[1]}_{1/4}$. Can these two complexity classes be equal? If we conjecture they are not equal, can we show some supporting evidence? Perhaps PH collapses if these two complexity classes are equal? For $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ machines we do not have an answer in either direction. However for $\mathrm{ZPP}^{\mathrm{SAT}\|[k]}$ machines and $k \geq 2$, we give a partial answer using results by Rohatgi [15] who showed a trade-off between success probability and the number of queries. We start with some definitions.

Definition 19 For constant k, we define the languages BL_k, coBL_k and ODD_k as follows. First, $\mathrm{BL}_1 = \mathrm{SAT}$.

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

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

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

$$\mathrm{ODD}_k = \{\langle x_1, \ldots, x_k \rangle \mid \text{the number of } x_i \in \mathrm{SAT} \text{ is odd}\}.$$

The languages BL_k and ODD_k are $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete languages for the k^{th} level of Boolean Hierarchy [5, 6].

Definition 20 We say that a sequence of Boolean formulas $\langle F_1, \ldots, F_k \rangle$ is *nested* if for all i, $2 \leq i \leq k$, $F_i \in \mathrm{SAT} \Longrightarrow F_{i-1} \in \mathrm{SAT}$.

Given any sequence of formulas $\langle F_1, \ldots, F_k \rangle$ we can construct a sequence of nested formulas $\langle F'_1, \ldots, F'_k \rangle$ in polynomial time such that $\langle F_1, \ldots, F_k \rangle \in \mathrm{ODD}_k \iff \langle F'_1, \ldots, F'_k \rangle \in \mathrm{ODD}_k$. Simply let $F'_j = F_j \vee \cdots \vee F_k$.

Theorem 21 (Rohatgi [15]) For $k \geq 1$,

$$\mathrm{BL}_{2k+2} \leq^{\mathrm{rp}}_{\mathrm{m}} \mathrm{BL}_{2k}, \quad \mathrm{BL}_{2k+1} \leq^{\mathrm{rp}}_{\mathrm{m}} \mathrm{BL}_{2k} \quad \text{and}$$
$$\mathrm{BL}_{2k+2} \leq^{\mathrm{rp}}_{\mathrm{m}} \mathrm{BL}_{2k+1}$$

with success probability $\sigma_k = 1 - 1/(k+1)$. Furthermore, improving the probability of success from σ_k to $\sigma_k + 1/poly$ in any of the $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reductions above implies PH collapses.

Rohatgi's theorem above gives us an indication of the best success probability that can be shown for $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reductions between adjacent levels of Boolean Hierarchy. It also provides insight on the limits of probability amplification for $\mathrm{RP}^{\mathrm{SAT}\|[k]}$ machines. For example, since BL_{2k+1} is contained in $\mathrm{RP}^{\mathrm{SAT}\|[2k]}_{\sigma_k}$, we have

$$\mathrm{RP}^{\mathrm{SAT}\|[2k]}_{\sigma_k} \subseteq \mathrm{RP}^{\mathrm{SAT}\|[2k]}_{\sigma_k+1/poly} \Longrightarrow \text{PH collapses.}$$

Similarly, $\mathrm{BL}_{2k+2} \in \mathrm{RP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k}$, so

$$\mathrm{RP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k} \subseteq \mathrm{RP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k+1/poly} \Longrightarrow \text{PH collapses.}$$

For $\mathrm{RP}^{\mathrm{SAT}\|[2k]}$ and $\mathrm{RP}^{\mathrm{SAT}\|[2k+1]}$ machines, we get

$$\mathrm{RP}^{\mathrm{SAT}\|[2k]}_{\sigma_k+1/poly} \subsetneq \mathrm{RP}^{\mathrm{SAT}\|[2k]}_{\sigma_k} \quad \text{and}$$
$$\mathrm{RP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k+1/poly} \subsetneq \mathrm{RP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k}$$

unless PH collapses. Here, we prove similar results for $\mathrm{ZPP}^{\mathrm{SAT}\|[2k]}$ and $\mathrm{ZPP}^{\mathrm{SAT}\|[2k+1]}$ machines. But for technical reasons we have to work with ODD_k instead of BL_k.

Lemma 22 For $k \geq 1$ and $\sigma_k = 1 - 1/(k+1)$,

$$\mathrm{ODD}_{2k+2} \in \mathrm{ZPP}^{\mathrm{SAT}\|[2k]}_{\sigma_k},$$
$$\mathrm{ODD}_{2k+1} \in \mathrm{ZPP}^{\mathrm{SAT}\|[2k]}_{\sigma_k}, \text{ and}$$
$$\mathrm{ODD}_{2k+2} \in \mathrm{ZPP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k}.$$

Furthermore improving the success probability of any of the above machines from σ_k to $\sigma_k + 1/poly$ implies PH collapses.

Proof: We shall prove that $\mathrm{ODD}_{2k+2} \in \mathrm{ZPP}^{\mathrm{SAT}\|[2k]}_{\sigma_k}$ which implies $\mathrm{ODD}_{2k+2} \in \mathrm{ZPP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k}$. Furthermore, since we know $\mathrm{ODD}_{2k+1} \leq^{\mathrm{P}}_{\mathrm{m}} \mathrm{ODD}_{2k+2}$, we will also have $\mathrm{ODD}_{2k+1} \in \mathrm{ZPP}^{\mathrm{SAT}\|[2k]}_{\sigma_k}$.

Consider $\langle F_1, \ldots, F_{2k+2} \rangle$, a $(2k+2)$-tuple of formulas. W.l.o.g., assume the formulas are nested. Now consider the set $\mathscr{S} = \{ \ (F_{2i-1}, F_{2i}) \mid 1 \leq i \leq k+1 \ \}$. The $\mathrm{ZPP}^{\mathrm{SAT}\|[2k]}$ machine uniformly randomly picks an element from $\mathscr{S}$ and drops the corresponding pair from the $(2k+2)$-tuple $\langle F_1, \ldots, F_{2k+2} \rangle$. Then, the machine uses the SAT oracle to determine the satisfiability of the remaining $2k$ formulas.

Next, let us assume that in the input $(2k+2)$-tuple $\langle F_1, \ldots, F_{2k+2} \rangle$, that F_l is the rightmost satisfiable formula. If l is odd, the machine can figure out that $\langle F_1, \ldots, F_{2k+2} \rangle \in \mathrm{ODD}_{2k+2}$ unless it dropped (F_l, F_{l+1}). Similarly if l is even, the machine can figure out that $\langle F_1, \ldots, F_{2k+2} \rangle \notin \mathrm{ODD}_{2k+2}$ unless it dropped (F_{l-1}, F_l) or (F_{l+1}, F_{l+2}). Therefore the probability that the machine outputs `dk` is $max\{1/|\mathscr{S}|, 2/|\mathscr{S}|\}$ which is equal to $2/(2k+2)$. Therefore the success probability of the $\mathrm{ZPP}^{\mathrm{SAT}\|[2k]}$ machine is $1 - 1/(k+1) = \sigma_k$.

Now we prove that the success probability cannot be increased by any $1/poly$ additive term. Suppose that $\mathrm{ODD}_{2k+2} \in \mathrm{ZPP}^{\mathrm{SAT}\|[2k]}_{\sigma_k+1/poly}$. Then, by converting the `dk`'s in the $\mathrm{ZPP}^{\mathrm{SAT}\|[2k]}$ machine into `rej`'s, we get an $\mathrm{RP}^{\mathrm{SAT}\|[2k]}_{\sigma_k+1/poly}$ machine for ODD_{2k+2}. This gives us an $\leq^{\mathrm{rp}}_{\mathrm{m}}$-reduction from BL_{2k+2} to BL_{2k} with success probability $\sigma_k + 1/poly$, which is not possible unless PH collapses (by Theorem 21). The proofs for the other cases are similar. □

Lemma 23 For $k \geq 1$ and $\sigma_k = 1 - 1/(k+1)$,

- $\mathrm{ZPP}^{\mathrm{SAT}\|[2k]}_{\sigma_k+1/poly} \subsetneq \mathrm{ZPP}^{\mathrm{SAT}\|[2k]}_{\sigma_k}$ unless PH collapses.
- $\mathrm{ZPP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k+1/poly} \subsetneq \mathrm{ZPP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k}$ unless PH collapses.

Proof: Suppose that $\mathrm{ZPP}^{\mathrm{SAT}\|[2k]}_{\sigma_k+1/poly} = \mathrm{ZPP}^{\mathrm{SAT}\|[2k]}_{\sigma_k}$. Then, we have $\mathrm{ODD}_{2k+2} \in \mathrm{ZPP}^{\mathrm{SAT}\|[2k]}_{\sigma_k+1/poly}$. Then, by Lemma 22, PH collapses. Similarly, if we are given that $\mathrm{ZPP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k+1/poly} = \mathrm{ZPP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k}$, then $\mathrm{ODD}_{2k+2} \in \mathrm{ZPP}^{\mathrm{SAT}\|[2k+1]}_{\sigma_k+1/poly}$ which collapses PH by Lemma 22. □

One consequence of these results concerns derandomization. While it might be possible to derandomize $\mathrm{ZPP}^{\mathrm{SAT}}$ so it equals $\mathrm{P}^{\mathrm{SAT}}$, can we derandomize $\mathrm{ZPP}^{\mathrm{SAT}\|[k]}$ so it equals $\mathrm{P}^{\mathrm{SAT}\|[k]}$? The results above show that such derandomization cannot be accomplished for some probability bounds unless PH collapses. For example, if we can derandomize $\mathrm{ZPP}^{\mathrm{SAT}\|[k]}_{\sigma_k}$ and keep the same number of queries, then

$$\mathrm{P}^{\mathrm{SAT}\|[k]} = \mathrm{ZPP}^{\mathrm{SAT}\|[k]}_{\sigma_k+1/poly} = \mathrm{ZPP}^{\mathrm{SAT}\|[k]}_{\sigma_k}$$

which collapses PH.

7 Open Problems

Lemma 23 gives us, for every k, a probability bound $\alpha \in (0,1)$ such that $\mathrm{ZPP}^{\mathrm{SAT}\|[k]}_{\alpha+1/poly} \subsetneq \mathrm{ZPP}^{\mathrm{SAT}\|[k]}_{\alpha}$, assuming PH does not collapse. We obtained α by showing that α is the optimum success probability with which a $\mathrm{ZPP}^{\mathrm{SAT}\|[k]}$ can recognize ODD_{k+2} (or ODD_{k+1} if k is odd). Would it be possible to find a sequence of α_i's, $\alpha_1 > \alpha_2 > \cdots > \alpha_i > \cdots$, such that $\mathrm{ZPP}^{\mathrm{SAT}\|[k]}_{\alpha_i+1/poly} \subsetneq \mathrm{ZPP}^{\mathrm{SAT}\|[k]}_{\alpha_i}$? Perhaps by showing that α_i is the optimal probability with which a $\mathrm{ZPP}^{\mathrm{SAT}\|[k]}$ can recognize ODD_{k+2i} or ODD_{k+2i-1} ? Our current proof techniques based on hard/easy arguments do not seem to help us in this pursuit.

As we mentioned in Section 2, we know that if $\mathrm{SAT} \wedge \overline{\mathrm{SAT}} \leq^{\mathrm{rp}}_{\mathrm{m}} \overline{\mathrm{SAT}} \vee \mathrm{SAT}$ with even $1/poly$ probability, then PH collapses to Σ^{P}_3. This gives us

$$\mathrm{RP}^{\mathrm{SAT}[1]}_{1/poly} = \mathrm{RP}^{\mathrm{SAT}\|[2]}_{1/poly} \Longrightarrow \mathrm{PH} \subseteq \Sigma^{\mathrm{P}}_3.$$

Can we show for some α that $\mathrm{RP}^{\mathrm{SAT}[1]}_{\alpha} = \mathrm{RP}^{\mathrm{SAT}\|[2]}_{\alpha}$ implies PH collapses all the way down to $\mathrm{RP}^{\mathrm{SAT}[1]}_{\alpha}$?

Now, consider the two queries problem for BPP. We know that $\mathrm{SAT} \wedge \overline{\mathrm{SAT}} \leq^{\mathrm{corp}}_{\mathrm{m}} \overline{\mathrm{SAT}} \vee \mathrm{SAT}$ with probability $1/2$. We can convert this $\leq^{\mathrm{corp}}_{\mathrm{m}}$-reduction into a $\leq^{\mathrm{bpp}}_{\mathrm{m}}$-reduction with success probability $2/3$. Rohatgi [15] showed that this is indeed the optimal probability of success for the $\leq^{\mathrm{bpp}}_{\mathrm{m}}$-reduction — i.e., any improvement of the probability of success from $2/3$ to $2/3 + 1/poly$ would collapse PH. Thus,

$$\mathrm{BPP}^{\mathrm{SAT}\|[2]}_{\alpha} \subseteq \mathrm{BPP}^{\mathrm{SAT}[1]}_{2/3+1/poly} \Longrightarrow \mathrm{PH \text{ collapses}}$$

for all $\alpha \geq 1/2 + 1/poly$. But we cannot prove such results when the success probability of the $\mathrm{BPP}^{\mathrm{SAT}[1]}$ machine is less than $2/3$.

References

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

[2] H. Buhrman and L. Fortnow. Two queries. *J. Comput. Syst. Sci.*, 59(2):182–194, 1999.

[3] J. Cai. $\mathrm{S}^{\mathrm{P}}_2 \subseteq \mathrm{ZPP}^{\mathrm{NP}}$. *J. Comput. Syst. Sci.*, 73(1):25–35, February 2007.

[4] J. Cai and V. Chakaravarthy. On zero error algorithms having oracle access to one query. *Journal of Combinatorial Optimization*, 11(2):189–202, 2006.

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

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

[7] R. Chang and J. Kadin. On computing Boolean connectives of characteristic functions. *Mathematical Systems Theory*, 28(3):173–198, May/June 1995.

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

[9] R. Chang, J. Kadin, and P. Rohatgi. On unique satisfiability and the threshold behavior of randomized reductions. *J. Comput. Syst. Sci.*, 50(3):359–373, June 1995.

[10] 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, June 2007.

[11] L. Fortnow, A. Pavan, and S. Sengupta. Proving SAT does not have small circuits with an application to the two queries problem. *J. Comput. Syst. Sci.*, 74(3):358–363, May 2008.

[12] L. Hemachandra. The strong exponential hierarchy collapses. *J. Comput. Syst. Sci.*, 39(3):299–322, 1989.

[13] E. Hemaspaandra, L. A. Hemaspaandra, and H. Hempel. Downward collapse within the polynomial hierarchy. *SIAM J. Comput.*, 28(2):383–393, April 1999.

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

[15] P. Rohatgi. Saving queries with randomness. *J. Comput. Syst. Sci.*, 50(3):476–492, June 1995.

[16] R. Tripathi. The 1-versus-2 queries problem revisited. In *Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC)*, volume 4835 of *Lecture Notes in Computer Science*, pages 137–147, December 2007.

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

[18] K. Wagner. Bounded query classes. *SIAM J. Comput.*, 19(5):833–846, 1990.