Electronic Colloquium on Computational Complexity, Report No. 144 (2013)

The two queries assumption and Arthur-Merlin classes

Vyas Ram Selvam
C-VEST, IIIT-H
vyasram.s@research.iiit.ac.in

Abstract

We explore the implications of the two queries assumption, $P^{NP[1]} = P_{||}^{NP[2]}$, with respect to the polynomial hierarchy and the classes AM and MA. We prove the following results:

1. $P^{NP[1]} = P_{||}^{NP[2]} \implies AM = MA$
2. $P^{NP[1]} = P_{||}^{NP[2]} \implies PH \subset MA_{/1}$
3. $\exists\, B\ P^{NP[1]^B} = P^{NP[2]^B}$ and $NP^B \not\subseteq coMA^B$.
4. $P^{NP[1]} = P_{||}^{NP[2]} \implies PH = P_{||}^{NP[1],MA[1]} = P_{||}^{NP[2],pr_{coNP}(MA \cap coMA)[1]}$ (here $pr_{coNP}(MA \cap coMA)$ refers to the promise class $pr(MA \cap coMA)$ where the promise is restricted to a $coNP$ set)

Under the two queries assumption, the best containment of PH known so far in a non-uniform class is $NP_{/poly}$ as shown by Kadin. Our result showing the containment of PH in $MA_{/1}$ betters this.

We also show that the question of whether PH collapses to $AM = MA$ under the two queries assumption is non-relativizable by constructing an oracle B such that $P^{NP[1]^B} = P^{NP[2]^B}$ and $NP^B \not\subseteq coMA^B$. This improves upon the result by Buhrman and Fortnow where they showed that the question of whether PH collapses to NP under the two queries assumption is not relativizable.

Two incomparable results are known regarding the collapse of PH under the two queries assumption: the collapse to $ZPP_{1/2-1/exp}^{NP[1]}$ shown by Chang and Purini and the collapse to $YO_2^p \cap NO_2^p$ shown by Chakaravarthy and Roy. Our results showing the collapse of PH to $P_{||}^{NP[1],MA[1]}$ and $P_{||}^{NP[2],pr_{coNP}(MA \cap coMA)[1]}$ are incomparable to these. These results also imply that a collapse of PH to $P^{NP[1]}$ is possible if and only if $MA \subseteq P^{NP[1]}$ or $pr_{coNP}(MA \cap coMA) \subseteq pr_{coNP}P^{NP[1]}$ under the two queries assumption.

1 Introduction

1.1 Background

The assumption that one query to a SAT oracle is as powerful as two queries to a SAT oracle leads to some interesting implications, with the most profound one being the collapse of the polynomial hierarchy. The question has a long history in the field of structural complexity theory with several results bringing down the collapse to smaller and smaller classes. Krentel showed that $FP^{NP[1]} = FP_{||}^{NP[2]} \implies P = NP$ [14]. While the assumption in the case of functions resulted in a straightforward collapse of the polynomial hierarchy to P; the implications of the corresponding assumption in the case of language classes, $P^{NP[1]} = P_{||}^{NP[2]}$, are not so direct as we do not know of any direct collapse to P or even NP. Throughout this paper, we shall use the term 'two queries assumption' to refer to the assumption: $P^{NP[1]} = P_{||}^{NP[2]}$.

ISSN 1433-8092

Using the hard/easy argument, Kadin showed the the two queries assumption implies that $coNP$ is contained in $NP_{/poly}$ [12]. This also means the containment of PH in the non-uniform class $NP_{/poly}$. This result along with Yap's theorem [17] implies the collapse of PH to Σ_3^p. Wagner improved the collapse to $P^{\Sigma_2^p}$ [16]. Further results brought down the collapse to $P^{\Sigma_2^p[1],NP[1]}$ [2] [7]. While a downward collapse to NP seemed elusive for the assumption $P^{NP[1]} = P_{||}^{NP[2]}$, Hemaspaandra et. al. showed a downward collapse for a general case of the assumption by showing that $P^{\Sigma_k^p[1]} = P_{||}^{\Sigma_k^p[2]} \implies PH \subseteq \Sigma_k^p$ (for $k > 2$) [11].

Buhrman and Fortnow extended the hard/easy argument of Kadin and arrived at three new results under the two queries assumption [4]:

1. $P^{NP[1]} = P^{NP}$
2. locally (at every input length n), either $NP = coNP$ or NP has polynomial sized circuits
3. there exists a relativized world where $P^{NP[1]} = P_{||}^{NP[2]}$,but $coNP \neq NP$.

Making use of the result that locally, either $NP = coNP$ or $NP \subset P_{/poly}$; Fortnow, Pavan and Sengupta showed that PH collapses to S_2^p under the two queries assumption [10]. Chakaravarthy and Roy improved this collapse to their newly defined class $YO_2^p \cap NO_2^p$ [6, Theorem 16]. Later, Chang and Purini showed a collapse of the polynomial hierarchy to $ZPP_{1/2-1/exp}^{NP[1]}$ [9, Theorem 18]. The last two results are incomparable since we do not know about the relation between the two classes independent of the two queries assumption.

Since a relativizable collapse to NP is not possible, let alone P, a collapse to $P^{NP} = P^{NP[1]}$ is much sought after and the collapse to $ZPP_{1/2-1/exp}^{NP[1]}$ comes very close to achieving that. However, the question of achieving a flat collapse to $P^{NP[1]}$ still remains open.

Some research work had been done related to the original problem. Tripathi studied the implication of a weaker assumption ($P_{||}^{NP[2]} \subseteq ZPP_{1/2+1/poly}^{NP[1]}$) and showed that PH collapses to S_2^p under this weaker assumption as well [15]. Chang and Purini studied the implication of the two queries assumption and the additional assumption that the NP machine hypothesis holds. They showed a collapse of PH to NP under the combination of these two assumptions [8].

1.2 Our results

In this paper, we venture on a slightly different track and arrive at some new results establishing some relationships between PH and Arthur-Merlin classes under the two queries assumption. Then we explore the difficulty of bringing down the collapse to $P^{NP[1]}$ and arrive at some necessary and sufficient conditions to achieve such a collapse.. We briefly discuss our results based on the two queries assumption and their significance below.

Firstly, we show the equivalence of AM and MA. We notice that the implications of the two queries assumption and the implications of the Karp-Lipton assumption ($NP \subset P_{/poly}$) are quite similar in the sense that they both result in the collapse of the polynomial hierarchy below S_2^p [6, Theorem 10]. With our result, we pronounce the similarity between these two assumptions even more as $NP \subset P_{/poly} \implies AM = MA$ [1].

Our second result is the containment of the polynomial hierarchy in the non uniform class

$MA_{/1}$. Our result betters the best known containment of PH in a non-uniform class under the two queries assumption - in $NP_{/poly}$, shown by Kadin long back [12].

In our third result, we construct an oracle relative to which the two queries assumption holds but PH does not collapse to $AM = MA$. In other words, we construct an oracle B relative to which $P^{NP[1]^B} = P^{NP[2]^B}$ but $NP^B \not\subseteq coMA^B$. Combining this with our first result($AM = MA$, which is relativizable as well), we show that collapsing PH to AM is not relativizable. This is an improvement on a similar result shown by Buhrman and Fortnow where they gave an oracle relative to which the two queries assumption holds, but PH does not collapse to NP [4, Theorem 4.1].

Our fourth result is a collapse of the polynomial hierarchy to the class $P_{||}^{NP[1],MA[1]}$. We also achieve a collapse to the class $P_{||}^{NP[2],pr_{coNP}(MA \cap coMA)[1]}$. Both of these collapses are incomparable with the other best known results: collapses to $ZPP_{1/2-1/exp}^{NP[1]}$ and $YO_2^p \cap NO_2^p$. Our characterization turns out to be really useful since our base class is just P and all the randomness required is passed over to the oracle; which, in turn, helps us analyze the possibility of achieving a flat collapse to $P^{NP[1]}$. With this result, it is trivial to show that PH collapses to $P^{NP[1]} = P^{NP}$ if and only if we could decide problems in MA with a P^{NP} machine under the two queries assumption. We also show that such a collapse is possible iff $pr_{coNP}(MA \cap coMA) \subseteq pr_{coNP}P^{NP}$.

Organization: In section 2, we brief on the necessary definitions and notations used. In section 3, we present our result showing that $AM = MA$ and our result showing the containment of PH in $MA_{/1}$ under the two queries assumption. In section 4, we construct an oracle B relative to which $P^{NP[1]} = P^{NP[2]}$ but $PH \neq AM$. In section 5, we present our result that $P^{NP[1]} = P_{||}^{NP[2]} \implies PH = P_{||}^{NP[1],MA[1]} = P_{||}^{NP[2],pr_{coNP}(MA \cap coMA)[1]}$ under the two queries assumption and discuss the possibility of a flat collapse to $P^{NP[1]}$. In section 6, we conclude by mentioning some related open questions.

2 Preliminaries

We assume that the reader is familiar with the standard notations and complexity classes. Throughout this paper, $(\exists^{f(n),g(n)} y, z)$ denotes $(\exists y, z : |y| = f(n)\ \&\ |z| = g(n)$ where n is the input length). Similar definitions apply for the universal operator $\forall^{f(n)} y$ and the probabilistic operator $Pr^{f(n)} y$.

Definition 1 (h)**:** *The assumption $P^{NP[1]} = P_{||}^{NP[2]}$ implies the existence of a polynomial-time function h that reduces the language $\{(a,b) \mid a \in \overline{SAT}\ \&\ b \in SAT\ \&\ |a| = |b|\}$ to the language $\{(c,+)|c \in SAT\} \cup \{(c,-)|c \in \overline{SAT}\}$ where c is of length $t1(|a| + |b|)$ for some polynomial $t1$.*

Definition 2 (SAT, Easy, Hard Classification)**:**

1. $SAT = \{x|x$ *is a satisfiable boolean formula*$\}$
2. $Easy = \{x|\exists^{|x|} y\ h(x,y) = (c,+)\ \&\ c \in SAT\}$
3. $Hard = \{x|x \notin SAT\ \&\ x \notin Easy\}$

The hard/easy argument in its various forms is very useful when dealing with the two queries assumption. We describe the classification first used by Kadin [12]. The function h can be used

to classify strings into three different types: SAT, $Easy$ and $Hard$. $Easy$ strings are unsatisfiable boolean formulae with a short proof of non-satisfiability. $Hard$ strings are unsatisfiable strings not in $Easy$. Trivially, $\overline{SAT} = Hard \cup Easy$. We make use of this classification while proving Theorem 15.

Theorem 3: $P^{NP[1]} = P_{||}^{NP[2]} \implies coNP \subset NP_{/poly}$ *[12].*

If there is a $Hard$ string at length n, then we can use that string as advice to have an $NP_{/poly}$ algorithm to solve $\overline{SAT}$. Otherwise, we have $NP = coNP$ at that length. Combining the two cases, we have $coNP \subset NP_{/poly}$. The first bit of the advice mentions whether there is a hard string and the remaining bits provide a hard string if there is one. This also implies the containment of PH in $NP_{/poly}$.

Lemma 4: $x \in Hard \implies (\forall y \text{ s.t. } |x| = |y| \ (y \in SAT \iff h(x,y) = (z,-) \ \& \ z \in \overline{SAT}))$ *[4, Lemma 5.6].*

Definition 5 (SAT, Easy4, Hard4 Classification)**:**

1. $SAT = \{x | x \text{ is a satisfiable boolean formula}\}$
2. $Easy4 = \{x | \exists^{t2(|x|)} y \ VerifyEasy4(x,y)\}$ *(where* $t2$ *is some polynomial and* $VerifyEasy4$ *is the polynomial-time verifier for* $Easy4$*)*
3. $Hard = \{x | x \notin SAT \ \& \ x \notin Easy4\}$

Buhrman and Fortnow extended the arguments of Kadin to classify strings in a way that there are fewer hard strings under the two queries assumption [4, Definition 5.4]. They called Kadin's classification as $Hard1/Easy1$ and defined newer classifications $Hard2/Easy2$, $Hard3/Easy3$ and $Hard4/Easy4$; successively decreasing the number of hard strings. The basic idea is the same; easy strings have short proof of non-satisfiability. We are interested only in the $Hard4/Easy4$ definition. However, these definitions are quite elaborate; so we just encapsulate the essence of the argument used (i.e. $Easy4$ strings have short proofs) and assume that $VerifyEasy4$ is the polynomial time verifier for $Easy4$ strings. Trivially, $\overline{SAT} = Hard4 \cup Easy4$. We make use of this classification while proving theorem 13.

Theorem 6: $P^{NP[1]} = P_{||}^{NP[2]} \implies$ *locally, either* $NP = coNP$ *or* $NP \subset P_{/poly}$ *[4, Theorem 5.3(1)].*

If there are no $Hard4$ strings at length n, then $NP = coNP$ locally. Otherwise, $NP \subset P_{/poly}$ at that length.

Definition 7 ($ZPP_{f(n)}^{NP[1]}$)**:** $ZPP_{f(n)}^{NP[1]}$ *defines a* ZPP *machine which can query an NP oracle once and outputs the correct answer with* $f(n)$ *probability (where* n *is the input length). The term poly is used to denote polynomial functions when used anywhere in* $f(n)$ *and exp is used to denote exponential functions on the input length. For example* $ZPP_{1/2+1/poly}^{NP[1]}$ *is used to denote a ZPP machine that can query an NP oracle once and succeed with probability* $1/2 + 1/p(n)$ *for some polynomial* p*. Chang and Purini have shown that* $ZPP_{1/poly}^{NP[1]} = ZPP_{1/4-1/exp}^{NP[1]}$ *and that* $ZPP_{1/2+1/poly}^{NP[1]} = ZPP_{1-1/exp}^{NP[1]}$ *[9, Theorems 6,8].*

Definition 8 ($YES(P)$)**:** $YES(P)$ *denotes the* YES *set of a promise problem* P*. Inputs to a problem in a promise class has 3 sets:* YES *(inputs which must be accepted),* NO*(inputs which must be rejected) and* $OUTSIDEPROMISE$*(inputs which can have arbitrary outputs).*

Definition 9 ($pr_C D$)**:** *$pr_C D$ denotes the promise analogue of the language class D where the promise itself is restricted to be in the class C. For example $pr_{coNP}(MA \cap coMA)$ is the promise version of the class $(MA \cap coMA)$ where the promise is a set in $coNP$.*

Theorem 10: $P^{NP[1]} = P_{||}^{NP[2]} \implies P^{NP} = P^{NP[1]}$ *[4, Theorem 5.3(2)].*

We observe that this result extends over promise variations as well. For example, $P^{NP[1]} = P_{||}^{NP[2]} \implies pr_{coNP}P^{NP} = pr_{coNP}P^{NP[1]}$.

Theorem 11: $P^{NP[1]} = P_{||}^{NP[2]} \implies PH = S_2^p = \Sigma_2^p = \Pi_2^p$ *[10].*

Theorem 12: $P^{NP[1]} = P_{||}^{NP[2]} \implies PH = ZPP_{1/2-1/exp}^{NP[1]}$ *[9, Theorem 18].*

A note on self-reducibility of SAT and encoding: The language SAT is self-reducible. If y, the input given, is on n variables $(x1, x2....xn)$; we can self-reduce it using the variable $x1$ to the new formulae $y\&(x1 == 1)$ and $y\&(x1 == 0)$. Now, $y \in SAT \iff ((y\&(x1 == 1)) \in SAT)\ OR\ ((y\&(x1 == 0)) \in SAT)$. We can self-reduce y with any set of free variables remaining. One very useful benefit we obtain from the self-reducible property of SAT is that we can verify membership of SAT in polynomial time by finding a witness when given access to a circuit that solves SAT in polynomial time. So, no circuit can falsely claim a formula to be satisfiable and get away with it.

Throughout this paper, we assume that the encoding used for the inputs to decide membership in SAT is such that all self-reduced formulae of an given formula y have length $|y|$. Moreover, we assume that the encoding used enables padding so that an input of a smaller length can be blown-up to a larger desired length as needed. Such an encoding is trivial and can have at most a polynomial blow-up in length corresponding to an encoding without such restrictions(features).

A note on promise oracles: A promise oracle is an oracle based on a promise problem rather than a decision problem. The promise oracle is supposed to provide the correct answer whenever the query asked by the base machine fulfills the promise condition(reply YES when the query belongs to the YES set, reply NO when the query belongs to the NO set, reply anything when the input is in $OUTSIDEPROMISE$) . For example, a P^{prAM} machine M on input x can ask any query to the $prAM$ oracle. The oracle is supposed to give the correct answer to the query as long as the query input fits the promise. The machine M in this example, however, solves a decision problem and must succeed regardless of the answers provided by the oracle to the queries asked outside the promise. We observe that a promise oracle PO can be replaced with the decision oracle $YES(PO)$ without any loss in computation power.

3 MA with one bit of advice

In this section, we show that PH is contained in $MA_{/1}$ under the two queries assumption. We do this in two parts stated as two theorems.

In the first part, we show the equivalence of MA and AM under the two queries assumption. We note that our result also implies that $coMA = coAM$ and is true for promise classes as well (for example, $pr_{coNP}MA = pr_{coNP}AM$). We also note that our proof is relativizable (i.e. for any oracle A, $P^{NP[1]^A} = P^{NP[2]^A} \implies MA^A = AM^A$). The intuition behind the proof is quite simple: if $P^{NP[1]} = P_{||}^{NP[2]}$; locally, either $NP \subset P_{/poly}$ (Case1) or $NP = coNP$

(Case2). If Case1 occurs locally, Merlin just sends the local circuit for SAT for the corresponding length. Arthur converts the AM problem to a problem where the non-deterministic part is a SAT formula and then uses self-reducibility to verify membership. If a majority of the coin tosses result in a formula satisfiable by the circuit, he accepts. If Case2 occurs locally, Merlin successively reduces the problem in AM to a problem in Π_2^p (trivial) and then to a problem in Σ_2^p (since $P^{NP[1]} = P_{||}^{NP[2]} \implies \Sigma_2^p = \Pi_2^p$ [10]) and then to a problem in NP (since locally $NP = coNP$) and then provides the corresponding certificate for it to Arthur. We now provide the formal proof.

Theorem 13: $P^{NP[1]} = P_{||}^{NP[2]} \implies AM = MA$

Proof. Assume that $P^{NP[1]} = P_{||}^{NP[2]}$ and consider any language L in AM. Let the input be x of length n. Let $p1$,$p2$,$p3$,$p4$ and $p5$ denote polynomials. If $P^{NP[1]} = P_{||}^{NP[2]}$, then for a fixed input length, either SAT has a circuit c of polynomial size at that length (in Case1) or $\overline{SAT}$ has a polynomial time verifier $VerifyEasy4$ (in Case2).

Without loss of generality, we can characterize the language as follows:
$x \in L \iff Pr_r\,[Q(x,r) \in SAT] > 2/3$ and $x \notin L \iff Pr_r\,[Q(x,r) \in SAT] < 1/3$
where Q is a polynomial-time function, $|r| = p1(n)$.

If Case1 holds at length $|Q(x,r)|$,
$x \in L \implies \exists\, c\, (Pr_r[V(Q(x,r),c)] > 2/3)$
where $|c| = p2(n)$ and V is the verifier that uses the self-reducibility property of SAT to check whether a formula claimed to be satisfiable by a circuit is indeed satisfiable (the same verifier used in Karp-Lipton theorems [13]).

Since, $AM \subseteq \Pi_2^p$ and $P^{NP[1]} = P_{||}^{NP[2]} \implies \Pi_2^p = \Sigma_2^p$, we have $AM \subseteq \Sigma_2^p$. So, L can be characterized as follows:
$x \in L \iff \exists\, y\, S(x,y) \in \overline{SAT}$; where $|y| = p3(n)$ and S is a polynomial-time function.

If Case2 holds at length $|S(x,y)|$,
$x \in L \implies \exists\, y,z\, VerifyEasy4(S(x,y),z).$; where $|z| = p4(n)$.

We can force the output lengths of Q and S to be equal through padding and let this length be $p5(n)$. Since one of the two cases Case1 or Case2 must hold at length $p5(n)$, we have

$x \in L \implies (\exists\, y,z\, VerifyEasy4(S(x,y),z)\, OR\, \exists\, c\, (Pr_r[V(Q(x,r),c)] > 2/3))$

Also, regardless of which of the two cases holds,
$x \notin L \implies \forall\, c\, Pr_r\,[V(Q(x,r),c)] < 1/3)$ and
$x \notin L \implies \forall\, y,z\, \neg VerifyEasy4(S(x,y),z).$

So, $x \in L \implies \exists\, y,z,c\, (Pr_r\,[V(Q(x,r),c)\, OR\, VerifyEasy4(S(x,y),z)] > 2/3)$
and $x \notin L \implies \forall\, y,z,c\, (Pr_r\,[V(Q(x,r),c)\, OR\, VerifyEasy4(S(x,y),z)] < 1/3)$

By definition, this is an MA characterization for L. So, $AM = MA$. □

Now, in the second part of this section, we prove that $P^{NP[1]} = P_{||}^{NP[2]} \implies PH \subset AM_{/1}$. It

is sufficient to show that $P^{NP[1]} = P_{||}^{NP[2]} \implies coNP \subset AM_{/1}$. We now describe the intuition behind the proof. Our proof considers the language $\overline{SAT}$ which is complete for $coNP$. Let the input be x. Under the two queries assumption, we can introduce a language called $HARDBITS$ similar to the one used by Buhrman et. al. [3, Theorem 2].

Definition 14 ($HARDBITS$)**:** $HARDBITS = \{0^m | \textit{there is a Hard strings of length } m\} \cup \{(0^{m-i}1^i) | \textit{ the i'th bit of the lexicographically smallest Hard strings for length m is 1}\}$. *(where the bits are numbered from* 1 *to* n*)*

Trivially, the language $HARDBITS \in PH$ which means that $HARDBITS \in ZPP^{SAT[1]}_{1/2-1/exp}$ under the two queries assumption. So it can be decided by a ZPP machine M with one query to a SAT oracle with $1/2 - 1/exp$ probability. On input y; M takes a random path , generates the query through a polynomial time function and then uses the answer to that query to output 1(accept), 0(reject) or $'?'$(don't know). For a given input y, there may exist a random path s.t. the query asked by M is either in SAT or $Easy$ and the final output on that path is either 0 or 1. Such inputs to M have easily verifiable proofs (the random path chosen and the short proof showing that the query generated is SAT or $Easy$).

Now, we provide an Arthur-Merlin protocol(with one bit of advice) for deciding $\overline{SAT}$. Consider the case where either there are no $Hard$ strings at length n or the case where $\forall_{0<=i<=n} i$ there exists an easily verifiable proof showing that M accepts or rejects $0^{n-i}1^i$. Here, the advice bit makes Arthur aware of this case and thus Arthur expects Merlin to provide either a short proof of non-satisfiability for the input x or provide the lexicographically smallest $Hard$ string $hlex$ at length n along with the necessary witnesses proving that and then a short proof showing that x in unsatisfiable assuming $hlex$ to be the advice (as in Theorem 3).

If the advice bits tells that neither of these two cases hold, then Arthur has a very simple way of sampling $Hard$ strings. In this case, $\exists_{0<=i<=n} i$ s.t. $0^{n-i}1^i$ does not have an easily verifiable witness to prove membership in $HARDBITS$. And yet, M succeeds on such an input with $1/2-1/exp$ probability; so, on that particular input, the query generated by M is in $Hard$ with $1/2 - 1/exp$ probability. For each $i(0 <= i <= n)$, Arthur uses $(n+1)^2$ random paths on input $0^{n-i}1^i$ and collects the queries asked by each of those random paths; thus generating a set of $(n+1)^3$ queries. The probability that there is a $Hard$ string in this set is very high. Arthur then asks Merlin to classify this set of strings into three sets: $Hard$, $Easy$ and SAT (as defined in def.2). Arthur also expects the corresponding witnesses for the strings in $Easy$ and SAT sets. For all pairs of strings (a, b) in $Hard$, Arthur asks Merlin to show that $h(a, b) = (c, +)$ OR $h(a, b) = (c, -)$ & $c \in SAT$ (the function h is as defined in Def.1). Since there is a hard string in this set with very high probability, Merlin cannot falsely claim a satisfiable string to be a $Hard$ string (by Lemma.4). So, with very high probability, Arthur knows which of those queries are satisfiable and which are not. Using this information, with very high probability, Arthur can compute every bit of the lexicographically smallest $Hard$ string $hlex$ at length n. Arthur proceeds to use this as the advice and asks Merlin for the proof to show that x is unsatisfiable (as in Theorem 3).

We now formalize our proof

Theorem 15: $P^{NP[1]} = P_{||}^{NP[2]} \implies coNP \subset AM_{/1}$

Proof. Assume that $P^{NP[1]} = P_{||}^{NP[2]}$ and consider $\overline{SAT}$ which is a complete language for $coNP$. We provide an $AM_{/1}$ algorithm to decide $\overline{SAT}$. Let the input be x of length n.

Let M be the $ZPP^{SAT[1]}_{1/3}$ machine to decide $HARDBITS$ (this is possible since $P^{NP[1]} = P^{NP[2]}_{||} \implies PH = ZPP^{NP[1]}_{1/2-1/exp}$). Without loss of generality, we can assume that the queries asked by the base ZPP machine of M are of the same length for a given input length. Let this be denoted by the polynomial $p1(n)$. Also, the random strings used by the machine can be considered to be of the same length for a given input length. Let this be denoted by the polynomial $p2(n)$. We now describe how M functions when y is given as input. M chooses a random path r and generates the query $Q(y,r)$. If $Q(y,r)$ is satisfiable, M outputs $P(y,r,1)$. If not, it outputs $P(y,r,0)$. Here Q is a polynomial-time function and P is a polynomial-time function that can output 1,0 or $'?'$.

$y \in HARDBITS$
$\iff Pr_r$ $[(Q(y,r) \in SAT$ AND $P(y,r,1) = 1)$ OR $(Q(y,r) \in Easy$ AND $P(y,r,0) = 1)$ OR $(Q(y,r) \in Hard$ AND $P(y,r,0) = 1)] > 1/3$
$\iff \exists\, r (Q(y,r) \in SAT$ AND $P(y,r,1) = 1)$ OR $(Q(y,r) \in Easy$ AND $P(y,r,0) = 1)$ OR $(Q(y,r) \in Hard$ AND $P(y,r,0) = 1)$

We now describe the $AM_{/1}$ algorithm:

The advice bit b is 1 if and only if one of the following(at least) is true

1. $\forall^n z\ z \notin Hard$
2. $\forall_{0<=i<=n} i\ \exists\ r$ ($(Q(0^{n-i}1^i, r) \in SAT$ AND $P(0^{n-i}1^i, r, 1) \neq' ?')$ OR $(Q(0^{n-i}1^i, r) \in Easy$ AND $P(0^{n-i}1^i, r, 0) \neq' ?')$)

When the advice bit is 1, the following protocol is used

1. Arthur asks Merlin to prove that $x \in Easy$ and accepts if Merlin provides the required certificate.
2. If not, Arthur asks the following: For each i(where $0 <= i <= n$), r_i s.t. $(Q(0^{n-i}1^i, r_i) \in SAT$ AND $P(0^{n-i}1^i, r_i, 1) \neq' ?')$ OR $(Q(0^{n-i}1^i, r_i) \in Easy$ AND $P(0^{n-i}1^i, r_i, 0) \neq' ?')$. Arthur also asks for the corresponding certificates showing that the formulae $Q(0^{n-i}1^i, r_i)$ (where $0 <= i <= n$) are indeed $Easy$ or SAT. Arthur rejects if Merlin doesn't comply.
3. Arthur calculates whether there is a $Hard$ string at length n by simulating M on input 0^n using the random path provided by Merlin along with the corresponding answer (and witness)to the query provided by Merlin. If there is no hard string, Arthur rejects. Otherwise, Arthur again simulates M n times (for each i) in a similar way, using the random paths and answers(with witnesses) provided by Merlin to calculate each of the bits of the lexicographically smallest $Hard$ string $hlex$ at length n.
4. Arthur asks Merlin to provide a proof of non-satisfiability for x assuming $hlex$ as advice (as in Theorem 3) and accepts if Merlin provides. Arthur rejects otherwise.

When the advice bit is 0, the following protocol is used

1. Arthur generates uniformly at random $n+1$ sets $R'_i (0 <= i <= n)$ each containing $(n+1)^2$ strings of length $p2(n)$. Arthur also generates $n+1$ sets R_i s.t. $R_i = \{Q(0^n - i1^i, r) | r \in R'_i\}$. Arthur combines these sets to create the set $RCombined = (\cup_i^{0<=i<=n} R_i)$.

2. Arthur asks Merlin to classify the strings in *RCombined* into three sets SAT, *Easy* and $Hard$. Arthur expects Merlin to provide the corresponding certificates for strings Merlin claims to be in *Easy* or SAT. For every pair of strings (a, b) classified as $Hard$ by Merlin, Arthur expects Merlin to show that $h(a, b) = (c, +)$ OR $(h(a, b) = (c, -)$ AND $c \in SAT)$. If Merlin doesn't comply, Arthur rejects.

3. From the answers provided by Merlin to these $(n + 1)^3$ queries, Arthur simulates the actions of M (on the random paths in R'_i for each i (where $0 <= i <= n$) to calculate all the bits of the lexicographically smallest $Hard$ string *hlex* at length n. If Arthur still couldn't decide on some bits of *hlex* from the information provided by Merlin or if different random paths give conflicting answers for a particular bit, Arthur accepts.

4. Otherwise, after calculating *hlex*, Arthur asks Merlin to provide a proof of non-satisfiability for x assuming *hlex* as advice (as in Theorem 3) and accepts if Merlin provides. Arthur rejects otherwise.

If the advice bit is 1 and $x \in L$, Merlin should be able to provide the necessary certificates to prove this; either by directly proving that $x \in Easy$ or by proving that there is a $Hard$ string at length n and then proving the value of each bit of the lexicographically smallest $Hard$ string *hlex* at length n and then providing the short proof for $x \in Hard$ using *hlex* as advice(as in Theorem 3). If $x \notin L$, Merlin has no way of fooling Arthur. So, we actually have an NP algorithm when the advice bit is 1.

If the advice bit is 0, Arthur understands that
$\exists_{0<=i<=n} i$ s.t. $Pr_r[Q(0^{n-i}1^i, r) \in Hard] > 1/3$.
By sampling $(n+1)^2$ random paths for each i, Arthur has a high probability $(> 1-(2/3)^{(n+1)^2})$ of having a $Hard$ string in *RCombined*. Merlin cannot falsely claim that a string is in SAT or *Easy* as he cannot provide the corresponding witness. If there is at least one $Hard$ string in *RCombined*, Merlin cannot falsely claim a string in SAT to be in $Hard$ (by Lemma 4). So, Merlin will be truthful in this classification with high probability. Since Arthur uses $(n+1)^2$ random paths for each i, the probability that there is at least one successful path (not outputting $'?'$ based on the answers given by Merlin to the queries) for every $M(0^{n-i}1^i)$ (where $0 <= i <= n$) is very high $(> (1 - (2/3)^{(n+1)^2})^{(n+1)})$. If Arthur's coin tosses were favorable (which would be with $> (1 - (2/3)^{(n+1)^2})^{(n+1)} - (2/3)^{(n+1)^2}$ probability), he would have the lexicographically smallest $Hard$ string *hlex* at length n and hence can ask Merlin to provide the proof for $x \in L$ assuming *hlex* as advice (as in Theorem 3). So, when $x \in L$, Merlin would be able to convince Arthur with probability 1 (either by helping him calculate *hlex* and then providing the proof for $x \in L$ or if Arthur is unable to calculate *hlex* or if Arthur wrongly calculates *hlex*). If $x \notin L$, Merlin has a small chance of fooling Arthur $(< 1 - ((1 - (2/3)^{(n+1)^2})^{(n+1)} - (2/3)^{(n+1)^2})$, this tends to 0). Thus, we have an AM algorithm when the advice bit is 0.

Combining the two protocols, we get an $AM_{/1}$ algorithm for deciding $\overline{SAT}$. □

Now, we combine the two theorems to get our desired result

Theorem 16: $P^{NP[1]} = P_{||}^{NP[2]} \implies PH \subset MA_{/1} \cap coMA_{/1}$

Proof. It is trivial to show that $coNP \subset AM_{/1} \implies PH \subset AM_{/1}$. Since any AM protocol can be converted into an MA protocol under the two queries assumption (Theorem 7), we have $PH \subset MA_{/1}$. Once again, it is trivial to show that $PH \subset MA_{/1} \implies PH \subset coMA/1$. So, $P^{NP[1]} = P_{||}^{NP[2]} \implies PH \subset MA_{/1} \cap coMA_{/1}$. □

4 PH vs AM does not relativize

In section 3, we showed that PH is contained in $MA_{/1}$ under the two queries assumption. A natural question one may ask is whether we could do away with that one bit of advice and thereby achieve a collapse to MA. Also, Chakaravarthy and Roy have shown a collapse to $YO_2^p \cap NO_2^p$ [6, Theorem 16]. This is somewhat close to achieving a collapse to $MA \cap coMA$ (as NO_2^p is just above MA in the polynomial hierarchy and is closely related to it, YO_2^p has a similar relationship with $coMA$).

Here, we show that it is not possible to bring down the collapse to $MA = AM$ through relativizable methods by providing an oracle relative to which the two queries assumption holds but $PH \neq AM$. This improves upon the result by Buhrman and Fortnow that it is not possible through relativizable techniques to bring the collapse down to NP [4, Theorem 4.1]. In the following theorem, we build an oracle relative to which NP is not a subset of $coMA$ but $PSPACE$ is equal to $P^{NP[1]}$.

Theorem 17: $\exists\ B\ NP^B \not\subseteq coMA^B\ AND\ P^{NP[1]^B} = PSPACE^B$

Proof. The oracle B is defined as $B = A \oplus TQBF$. So, the base machine can ask queries to the oracle A as well as the $PSPACE - Complete$ oracle $TQBF$. We will define A later. First, let us define the language L_A

Definition 18 (L_A)**:** $L_A = \{1^n | \exists x\ s.t.\ |x| = n\ AND\ x \in A\}$

Trivially, $L_A \in NP^A$. Now, we construct A in such a way that $L_A \notin coMA^A$.

Consider the set S of all one round Merlin-Arthur protocols with oracle access to B(where Arthur is restricted to probabilistic polynomial time computation power and Merlin is restricted to polynomial space computation power). The number of such protocols is enumerable and restricting Merlin to use only polynomial space does not affect the definition of MA. Let S_i ($i \in \mathbb{N}$) denote the i'th protocol in this enumeration where Arthur receives a proof string of size $p1_i(n)$ bits from Merlin, uses $p2_i(n)$ bits of randomness and runs in $p3_i(n)$ time ($p1_i$, $p2_i$ and $p3_i$ are all polynomials and n is the input length). Both Arthur and Merlin have oracle access to B. For a given $coMA$ protocol S_i,

S_i is said to accept an input $x \iff$ for every proof y of size $p1_i(n)$, Arthur rejects it with probability $< 1/3$ over the coin tosses r.

S_i is said to reject an input $x \iff$ there exists a proof y of size $p1_i(n)$ such that Arthur rejects with probability $> 2/3$ over the coin tosses r

S_i is valid $\iff$ for all inputs, the protocol either accepts or rejects.

Initially, the oracle A is empty. We build the oracle A in stages. The oracle A is gappy and is restricted to contain at most one string at lengths which are towers of 2 and no strings at other lengths. We define a function $nextLength$ as follows:
$nextLength(i) = n$, where n is the smallest tower of 2 such that $2^n > 6 * p3_i(n)$ and $n > nextLength(i-1)$ (we set $nextLength(0) = 0$)

In the i'th stage, we diagonalize against the i'th protocol(S_i) in S by the following rules.

1. If S_i is not a valid $coMA$ protocol: do nothing.

2. If S_i accepts 1^n ($n = nextLength(i)$): do nothing.

3. If S_i rejects 1^n ($n = nextLength(i)$): pick a proof y of length $p1_i(n)$ such that Arthur rejects with probability $> 2/3$ over random strings of size $p2_i(n)$ when y is provided as proof and add a string a of length n to A such that the string is queried (whether $a \in A$) by at most $p3_i(n) * 2^{p2_i(n)}/2^n$ of these random paths.

If S_i is not a valid $coMA$ protocol, it cannot be an valid protocol for L_A. If S_i accepts 1^n, we don't add anything to A and A remains empty at that length. So, S_i accepts 1^n and $1^n \notin L_A$. So, in both these cases, S_i fails by default and we don't need to add anything to A.

If $_i$ rejects $1^{nextLength(i)}$, we have to add a string a to A such that S_i still rejects or becomes an invalid protocol. Since Arthur runs in time $p3_i(n)$, Arthur can ask at most $p3_i(n)$ queries to oracle B (which means at most $p3_i(n)$ queries to the A part of B) in any random path.

There are $2^{p2_i(n)}$ possible random paths. So, a total of $p3_i(n) * 2^{p2_i(n)}$ queries could be asked in total across all possible random paths. A total of 2^n different strings of length n could be queried.

So, by a simple averaging argument, we can see that there exists a string a of length n which is queried(whether $a \in A$) at most $p3_i(n) * 2^{p2_i(n)}/2^n$ times. We add this string a to A. Since $2^n > 6 * p3_i(n), \quad p3_i(n)/2^n < 1/6$.

So, at most $1/6th$ of the random paths ask this particular query $a \in A$. So, there can be at most a $1/6th$ decrease in the ratio of rejecting paths. So, the ratio of rejecting paths does not go below $1/2$. So, either the protocol becomes invalid (if the new ratio of rejecting paths is between $1/2$ and $2/3$) or the input still remains rejected by the protocol even though $1^n \in L_A$ (since a of length n is in A).

We can diagonalize against the enumeration of all Merlin-Arthur protocols by following the above rules. So, $L_A \in NP^A$ and $L_A \notin coMA^A$. So, $NP^B \not\subseteq coMA^B$.

We now show that $PSPACE^B \subseteq P^{NP[1]^B}$. The proof is the same as the one used by Fortnow and Buhrman to show that $P^{NP[1]} = PSPACE$ relative to $UP - Generics\ G$ [4, Theorem 4.1].

The oracle B is a join of two oracles A and $TQBF$. Here, $TQBF$ is $PSPACE - Complete$ and A is gappy (at most one string at lengths which are towers of 2 and no strings at other lengths). Consider a $P^{NP[1]^B}$ machine, there can be at most one 'interesting' length to query about in A by making use of the one available NP query. The other possible lengths are either too small that strings could be queried by directly querying A(which we assume that the algorithm does implicitly) or too large to query even with the help of non-determinism. The 'interesting' length is the length where making use of the NP query seems to provide additional computation power. We call this length the 'cookie-length' and if a string of this length is present in A, we call that string the 'cookie'.

Let M^B be an alternating Turing machine with oracle access to B. We now devise a $P^{NP[1]^B}$ algorithm which can decide $L(M)$ on input x:

First, ask the $TQBF$ oracle whether M^G accepts x if there is no cookie.

1. If yes, accept if and only if the following NP query (with oracle access to B) is false: $\exists\ y$ s.t. ($|y|$=cookie-length) AND ($y \in A$) AND ($M^B(x)$ rejects when cookie y is assumed to be present)

2. If no, accept if and only if the following NP query (with oracle access to B) is true: $\exists\ y$ s.t. ($|y|$=cookie-length) AND ($y \in A$) AND ($M^B(x)$ accepts when cookie y is assumed to be present)

Since there could only be one possible cookie, the NP query will find it and use the $TQBF$ oracle to proceed with the rest of the computation(whether $M^B(x)$ accepts/rejects when the cookie is assumed to be present). So, we ask only one NP query and accept if and only if M^B accepts. This completes our proof. □

Relative to the oracle B constructed above, the assumption $P^{NP[1]} = P^{NP[2]}$ holds; but $PH \neq MA$ (since relative to B, $NP \not\subseteq coMA$). We've already shown in section 3 that $AM = MA$(with a relativizable proof) under the two queries assumption. So, relative to the oracle B, the assumption $P^{NP[1]} = P^{NP[2]}$ holds, but $PH \neq AM$. So, the question of whether PH collapses to AM under the two queries assumption is not relativizable.

5 Towards a flat collapse to $P^{NP[1]}$

While a relativizable collapse to MA is not possible, we do not know of any such result for the longstanding question of whether PH collapses to $P^{NP[1]}$ under the two queries assumption. In this section, we look at the possibility of achieving a flat collapse of the polynomial hierarchy to $P^{NP[1]}$ under the two queries assumption. A collapse to $P^{NP[1]}$ is highly sought after, and the collapse to $ZPP^{NP[1]}_{1/2-1/exp}$ comes close to achieving that. However, there are two gaps preventing this flat collapse: the gap between $P^{NP[1]}$ and $ZPP^{NP[1]}_{1/2+1/poly}$ and the gap between $ZPP^{NP[1]}_{1/2+1/poly}$ and $ZPP^{NP[1]}_{1/2-1/exp}$.

A direct approach to solving this is trying to find a reduction from $ZPP^{NP[1]}_{1/2-1/exp}$ to P^{NP} under the two queries assumption and thereby closing these two gaps. We show that such a reduction is possible if and only if $MA \subseteq P^{NP}$ by showing that $PH = P_{||}^{NP[1],MA[1]}$. We also show that PH collapses to $P_{||}^{NP[2],pr_{coNP}(MA \cap coMA)}$ and thereby show that a collapse to $P^{NP[1]}$ is possible if and only if $pr_{coNP}(MA \cap coMA) \subseteq pr_{coNP}P^{NP[1]}$.

We now prove two unconditional results for the class $ZPP^{NP[1]}_{1/poly}$. The proofs for the following two theorems are similar to the proof used by Cai and Chakaravarthy to show that $ZPP^{NP[1]}_{1/2+1/poly} \subseteq S_2^p$ [5, Theorem 3.2] (They also implicitly showed that $ZPP^{NP[1]}_{1/2+1/poly} \subseteq P_{||}^{NP[2],pr_{coNP}BPP[1]}$)

Theorem 19: $ZPP^{NP[1]}_{1/poly} \subseteq P^{NP[1],AM[1]}$

Proof. Since $ZPP^{NP[1]}_{1/poly} = ZPP^{NP[1]}_{1/4-1/exp}$ [9, Theorem 6] , we just have to prove our theorem for $ZPP^{NP[1]}_{1/5}$.

Consider any language L in $ZPP^{NP[1]}_{1/5}$. Let x be the input of length n and M be the $ZPP^{NP[1]}_{1/5}$ machine for the language L. M uses $p2(n)$ bits of randomness r and asks the query $Q(x, r)$ to the oracle and outputs $P(x, r, b)$ where b is the answer to the query. Here $p2$ is a polynomial, Q

is a polynomial time function and P is a polynomial time function that can output 1,0 or '?'. By definition,

$x \in L \iff Pr_r\ [(Q(x,r) \in SAT \text{ AND } P(x,r,1) = 1) \text{ OR } (Q(x,r) \in \overline{SAT} \text{ AND } P(x,r,0) = 1)] > 1/5$
$\iff \exists\, r\ (Q(x,r) \in SAT \text{ AND } P(x,r,1) = 1) \text{ OR } (Q(x,r) \in \overline{SAT} \text{ AND } P(x,r,0) = 1).$

$x \notin L \iff Pr_r\ [(Q(x,r) \in SAT \text{ AND } P(x,r,1) = 0) \text{ OR } (Q(x,r) \in \overline{SAT} \text{ AND } P(x,r,0) = 0)] > 1/5$
$\iff \exists\, r\ (Q(x,r) \in SAT \text{ AND } P(x,r,1) = 0) \text{ OR } (Q(x,r) \in \overline{SAT} \text{ AND } P(x,r,0) = 0).$

We show that L can be characterized as $\overline{L1} \cap L2$ where $L1 \in NP$ and $L2 \in AM$. Here,
$L1 = \{x | \exists\, r\ Q(x,r) \in SAT \text{ AND } P(x,r,1) = 0\}$ and $L2 = L \cup L1$.
By definition $L1 \in NP$ and $\overline{L1} \cap L2 = L$. So, we just have to show that $L2 \in AM$, which we do by providing the following AM protocol for the language.

1. Arthur generates a set R' of 100 strings each of length $p2(n)$ uniformly at random and computes the set $R = \{r \in R' | P(x,r,0) = 0\}$.
2. Arthur asks Merlin to prove that $x \in L1$ by providing a witness or to show that $\forall_{r \in R} Q(x,r) \in SAT$ by providing the corresponding witnesses.
3. Arthur accepts if Merlin provides the required witnesses or else rejects.

Assume that $x \in L2$. If $x \in L1$, Merlin would just prove it by providing a verifiable witness. Suppose $x \notin L1$, then $x \in L$ (since $L2 = L \cup L1$). If $x \in L$, there cannot be an r s.t. $P(x,r,0) = 0$ AND $Q(x,r) \in \overline{SAT}$. So, Merlin could just the provide witnesses to show that $\forall_{r \in R} Q(x,r) \in SAT$. This satisfies the completeness requirement.

Assume that $x \notin L2$. Then, $x \in \overline{L} - L1$. So, $Pr_r\ [Q(x,r) \in \overline{SAT} \text{ AND } P(x,r,0) = 0] > 1/5$. So, with high probability($> 1 - (4/5)^{100}$) probability, Arthur would've picked one such r in R and Merlin wouldn't be to prove that $Q(x,r) \in SAT$ for that r. This satisfies the soundness requirement. So, $L2 \in AM$. □

Theorem 20: $ZPP^{NP[1]}_{1/poly} \subseteq P^{NP[2],pr_{coNP}(AM \cap coAM)[1]}_{||}$

Proof. Let the definitions and assumptions used in the previous theorem hold here as well. We now describe the $P^{NP[2],pr_{coNP}(AM \cap coAM)[1]}_{||}$ algorithm for L.

Ask if $x \in LA$ and accept if true where $LA = \{x | \exists\, r\ Q(x,r) \in SAT \text{ AND } P(x,r,1) = 1\}$

Ask if $x \in LR$ and reject if true where $LR = \{x | \exists\, r\ Q(x,r) \in SAT \text{ AND } P(x,r,1) = 0\}$

If one of these two NP queries is true, we can output the answer directly. Under the promise that both these queries to the NP oracle are false, membership in L can be decided by the following AM protocol

1. Arthur generates a set R' of 100 strings uniformly at random and computes the set $R = \{r \in R' | P(x,r,0) = 0\}$.
2. Arthur asks Merlin to show that $\forall_{r \in R}\ Q(x,r) \in SAT$ by providing witnesses.

3. Arthur accepts if Merlin provides the required witnesses or else rejects.

If $x \in L$, there cannot be an r s.t. $Q(x,r) \in \overline{SAT}$ AND $P(x,r,0) = 0$. So, Merlin should succeed every time. If $x \notin L$ and the promise is true, then $Pr_r\ [Q(x,r) \in \overline{SAT}$ AND $P(x,r,0) = 0] > 1/5$. So, with high probability ($> 1 - (4/5)^{100}$), $\exists r1 \in R'$ s.t. $Q(x,r1) \in \overline{SAT}$ AND $P(x,r1,0) = 0$. So, Merlin would not be able to prove that $Q(x,r1) \in SAT$ and Arthur will ultimately reject with high probability.

Under the same promise, we can have an AM protocol to decide membership in $\overline{L}$. So, we have reduced the problem to a promise problem in $pr(AM \cap coAM)$. Since the promise $\overline{LA \cup LR} \in coNP$, we have actually reduced the original problem to a problem in $pr_{coNP}(AM \cap coAM)$.

So, we can solve L using two NP queries and one $pr_{coNP}(AM \cap coAM)$ query and all these can be asked in parallel. This completes our proof. □

Making use of our unconditional results on $ZPP^{NP[1]}_{1/poly}$, our result that $MA = AM$ under the two queries assumption and the collapse of PH to $ZPP^{NP[1]}_{1/2-1/exp}$ under the two queries assumption, we can state the following.

Theorem 21: $P^{NP[1]} = P^{NP[2]}_{||} \implies PH = P^{NP[1],MA[1]}_{||} = P^{NP[2],pr_{coNP}(MA \cap coMA)[1]}_{||}$

Proof. We prove in in two parts.

By theorems 12,13, and 19 we have $PH \subseteq P^{NP[1],MA[1]}_{||}$. Trivially, $P^{NP[1],MA[1]}_{||} \subseteq PH$. So, $P^{NP[1]} = P^{NP[2]}_{||} \implies PH = P^{NP[1],MA[1]}_{||}$.

By theorems 12, 13 and 20, we have $PH \subseteq P^{NP[2],pr_{coNP}(MA \cap coMA)[1]}_{||}$. We can always replace a promise oracle PO with a language oracle L which is the YES set of the promise oracle ($L = YES(PO)$). Here, the promise oracle $PO \in pr_{coNP}(MA \cap coMA)$ and the corresponding YES set is in $coMA$ (a conjunction of the $coNP$ promise and the $coMA$ characterization of the problem). Hence, $P^{NP[2],pr_{coNP}(MA \cap coMA)[1]}_{||} \subseteq P^{NP[1],MA[1]}_{||} = PH$. So, $P^{NP[1]} = P^{NP[2]}_{||} \implies PH = P^{NP[2],pr_{coNP}(MA \cap coMA)[1]}_{||}$. □

We can state the following results concerning the possibility of a collapse to $P^{NP[1]}$ under the two queries assumption.

Corollary 22: *Under the two queries assumption,*

1. $PH \subseteq P^{NP[1]} \iff MA \subseteq P^{NP[1]}$
2. $PH \subseteq P^{NP[1]} \iff pr_{coNP}(MA \cap coMA) \subseteq pr_{coNP}P^{NP[1]}$

Proof. The forward direction of the first result is trivial. The converse is a direct implication of theorems 21 and 10. The forward direction of the second result is true because $YES(pr_{coNP}(MA \cap coMA)) \subseteq PH$. The converse is a direct result of theorems 21,10 and the fact that $P^{NP[1]} = P^{NP[2]}_{||} \implies YES(pr_{coNP}P^{NP[1]}) \subseteq P^{NP[1]}$. □

6 Conclusion and open questions

Whether $P^{NP[1]} = P_{||}^{NP[2]} \implies PH = P^{NP[1]}$ is a long standing open question. Our results enable two different approaches to this problem.

Buhrman, Chang and Fortnow showed that $coNP \subset NP_{/k} \implies PH = P^{NP}$ (for constant k) [3]. While we are not able to show that $P^{NP[1]} = P_{||}^{NP[2]} \implies coNP \subset NP_{/k}$, theorem 15 shows a containment of $coNP$ (and PH) in $MA_{/1}$. An interesting question is the containment of MA in $NP_{/k}$ under the two queries assumption. If the two queries assumption implies such a containment, we would get the desired flat collapse to $P^{NP[1]}$.

We have also shown that a collapse to $P^{NP[1]}$ is possible if and only if we can show that $MA \subseteq P^{NP[1]}$ or $pr_{coNP}(MA \cap coMA) \subseteq pr_{Co_NP}P^{NP[1]}$ under the two queries assumption. Though our results make use of promise classes, even showing that $MA \cap coMA \subseteq P^{NP[1]}$ under the two queries assumption might help us collapse the polynomial hierarchy to $P^{NP[1]}$. Even the question of whether $BPP \subseteq P^{NP[1]}$ under the two queries assumption is open. One may also look into the possibility of a relativized world where the two queries assumption holds, but $BPP \not\subseteq P^{NP[1]}$.

A related open question, asked by Buhrman and Fortnow [4], is whether $\overline{SAT}$ can be described as the union/intersection of an NP set and a $BPP_{/1}$ set under the two queries assumption. Our results make some progress on this front by showing that $coNP \subset MA_{/1}$. Our approach primarily relied on the simple $Hard/Easy$ classification. One may look into the possibility of extending this technique by making use of the $Hard4/Easy4$ classification to answer this question.

Acknowledgements I want to thank Suresh Purini for providing valuable suggestions and proof-reading the paper.

References

[1] V. Arvind, J. Köbler, U. Schöning, and R. Schuler. If NP has polynomial-size circuits, then MA=AM. *Theor. Comput. Sci.*, 137(2):279–282, 1995.

[2] R. Beigel, R. Chang, and M. Ogihara. A relationship between difference hierarchies and relativized polynomial hierarchies. *Mathematical systems theory*, 26(3):293–310, 1993.

[3] H. Buhrman, R. Chang, and L. Fortnow. One Bit of Advice. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '03, pages 547–558, 2003.

[4] H. Buhrman and L. Fortnow. Two Queries. *Journal of Computer and System Sciences*, 59:182–194, 1998.

[5] J. Cai and V. Chakaravarthy. A note on zero error algorithms having oracle access to one NP query. In *Proceedings of the 11th annual international conference on Computing and Combinatorics*, pages 339–348, 2005.

[6] V. Chakaravarthy and S. Roy. Oblivious symmetric alternation. In *Proceedings of the 23rd Annual conference on Theoretical Aspects of Computer Science*, pages 230–241, 2006.

[7] R. Chang and J. Kadin. The Boolean Hierarchy and the Polynomial Hierarchy: A Closer Connection. *SIAM J. Comput.*, 25(2):340–354, 1996.

[8] R. Chang and S. Purini. Bounded queries and the np machine hypothesis. In *Proceedings of the 2007 IEEE 22nd Annual Conference on Computational Complexity*, pages 52–59, 2007.

[9] R. Chang and S. Purini. Amplifying $ZPP^{SAT[1]}$ and the Two Queries Problem. In *Proceedings of the 2008 IEEE 23rd Annual Conference on Computational Complexity*, pages 41–52, 2008.

[10] 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, 2008.

[11] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. A Downward Collapse within the Polynomial Hierarchy. *SIAM J. Comput.*, 28(2):383–393, 1999.

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

[13] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 302–309, 1980.

[14] M. Krentel. The complexity of optimization problems. *J. Comput. Syst. Sci.*, 36(3):490–509, 1988.

[15] R. Tripathi. The 1-versus-2 queries problem revisited. *Theor. Comp. Sys.*, 46(2):193–221, 2010.

[16] K. Wagner. *Number-of-query Hierarchies*. Institut für Informatik Würzburg: Report. 1989.

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

ECCC ISSN 1433-8092

http://eccc.hpi-web.de