New Lowness Results for $\mathrm{ZPP}^{\mathrm{NP}}$ and other Complexity Classes[*]

V. Arvind
Institute of Mathematical Sciences
C.I.T Campus
Chennai, India 600 113
email: arvind@imsc.ernet.in

Johannes Köbler[†]
Humboldt Universität zu Berlin
Unter den Linden 6
D-10099, Berlin, Germany
email: koebler@informatik.hu-berlin.de

Abstract

We show that the class AM $\cap$ coAM is low for $\mathrm{ZPP}^{\mathrm{NP}}$. As a consequence, it follows that Graph Isomorphism and several group-theoretic problems are low for $\mathrm{ZPP}^{\mathrm{NP}}$. We also show that the class IP[P/poly], consisting of sets that have interactive proof systems with honest provers in P/poly, is also low for $\mathrm{ZPP}^{\mathrm{NP}}$. For the nonuniform function classes NPMV/poly, NPSV/poly, and NPMV_t/poly, we show the following lowness results: Sets whose characteristic functions are in NPSV/poly and that have program checkers are low for AM and $\mathrm{ZPP}^{\mathrm{NP}}$. Self-reducible sets with characteristic functions in NPMV_t/poly are low for Σ_2^p. Sets whose characteristic functions are in NPMV/poly and that have program checkers are low for Σ_2^p. We also give applications of these lowness results.

Keywords: Lowness, self-reducible set, interactive proof system, probabilistic class, function class, program checker.

[*] A preliminary version of this paper was presented in the 17th International Symposium on Theoretical Aspects of Computer Science. February 17-19, 2000, Lille.

[†] Part of the work done while at Abteilung für Theoretische Informatik, Universität Ulm, Oberer Eselsberg, D-89069 Ulm, Germany.

1 Introduction

In the recent past the probabilistic class $\mathrm{ZPP}^{\mathrm{NP}}$ has appeared in different results and contexts in complexity theory research. E.g. consider the result $\mathrm{MA} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$ [1, 15] which sharpens and improves Sipser's theorem $\mathrm{BPP} \subseteq \Sigma_2^p$. The proof in [1] uses derandomization techniques based on hardness assumptions [25]. Another example is the result that if $\mathrm{SAT} \in \mathrm{P/poly}$ then $\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}}$ [23, 5], which improves the Karp-Lipton theorem: if $\mathrm{SAT} \in \mathrm{P/poly}$ then PH collapses to Σ_2^p. Actually, Köbler and Watanabe in [23] prove that every self-reducible set A in $(\mathrm{NP} \cap \text{co-NP})/\mathrm{poly}$ is *low* for $\mathrm{ZPP}^{\mathrm{NP}}$, i.e. $\mathrm{ZPP}^{\mathrm{NP}^A} = \mathrm{ZPP}^{\mathrm{NP}}$. (In this paper, by self-reducibility we always mean word-decreasing self-reducibility which is adequate because standard complexity classes contained in EXP have such self-reducible complete problems.) This stronger result of [23] is in a sense natural, since there is usually an underlying lowness result that implies a collapse consequence result like the Karp-Lipton theorem. We may recall here that the lowness result underlying the Karp-Lipton theorem is that self-reducible sets in P/poly are low for Σ_2^p [28]. The notion of lowness was first introduced in complexity theory by Schöning [28]. It has since then been an important conceptual tool in complexity theory, see e.g. the survey paper [19].

In the next two subsections we state the main contributions of this paper.

1.1 $\mathrm{AM} \cap \mathrm{coAM}$ *and* IP[P/poly] *are low for* $\mathrm{ZPP}^{\mathrm{NP}}$

We recall the formal definition of lowness [28]. For a relativizable complexity class $\mathcal{C}$ such that for all sets A, $A \in \mathcal{C}^A$, let $Low(\mathcal{C})$ denote $\{A \mid \mathcal{C}^A = \mathcal{C}\}$. Clearly, $Low(\mathcal{C})$ is contained in $\mathcal{C}$ and consists of languages that are powerless as oracle for $\mathcal{C}$.

Few complexity classes have their low sets exactly characterized. These are the well-known examples: $Low(\mathrm{NP}) = \mathrm{NP} \cap \text{co-NP}$, $Low(\mathrm{AM}) = \mathrm{AM} \cap \mathrm{coAM}$ [28]. For most complexity classes however, a complete characterization of low sets appears to be a challenging open question. Regarding $Low(\Sigma_2^p)$, Schöning proved [29] that $\mathrm{AM} \cap \mathrm{coAM}$ is contained in $Low(\Sigma_2^p)$, implying that $Low(\mathrm{AM}) \subseteq Low(\Sigma_2^p)$. This containment is anomalous because $\mathrm{AM} \not\subseteq \Sigma_2^p$ in some relativized worlds [27]. Indeed, lowness appears to have other anomalous properties: it is not known to preserve containment of complexity classes, for example $\mathrm{NP} \subseteq \mathrm{PP}$ but $\mathrm{NP} \cap \text{co-NP}$ is not known to be contained in $Low(\mathrm{PP})$. Similarly, $\mathrm{NP} \subseteq \mathrm{MA}$ but $Low(\mathrm{NP}) = \mathrm{NP} \cap \text{co-NP}$ is not known to be contained in $Low(\mathrm{MA})$. Little is known about $Low(\mathrm{MA})$ except that it contains BPP and is contained in $\mathrm{MA} \cap \text{co-MA}$ [21].

Regarding $\mathrm{ZPP}^{\mathrm{NP}}$, it is shown in [23] that $Low(\mathrm{ZPP}^{\mathrm{NP}}) \subseteq Low(\Sigma_2^p)$. No characterization of $Low(\mathrm{ZPP}^{\mathrm{NP}})$ is known. In this paper, we show some new inclusions in $Low(\mathrm{ZPP}^{\mathrm{NP}})$.

- We show that $\mathrm{AM} \cap \mathrm{coAM}$ is low for $\mathrm{ZPP}^{\mathrm{NP}}$, i.e. $\mathrm{AM} \cap \mathrm{coAM} \subseteq Low(\mathrm{ZPP}^{\mathrm{NP}})$. Hence we have the inclusion chain: $Low(\mathrm{MA}) \subseteq Low(\mathrm{AM}) \subseteq Low(\mathrm{ZPP}^{\mathrm{NP}}) \subseteq Low(\Sigma_2^p)$. It follows that Graph Isomorphism and other group-theoretic problems in $\mathrm{AM} \cap \mathrm{coAM}$ [3] are low for $\mathrm{ZPP}^{\mathrm{NP}}$.

- Let IP[P/poly] denote the class of languages that have interactive proof systems with honest prover in P/poly. We show that $\mathrm{IP[P/poly]} \subseteq Low(\mathrm{ZPP}^{\mathrm{NP}})$, improving the containment $\mathrm{IP[P/poly]} \subseteq Low(\Sigma_2^p)$ shown in [2]. Our proof has a derandomization component in which the Nisan-Wigderson pseudorandom generator [25] is used to derandomize the verifier in the IP[P/poly] protocol. The rest of the proof is based on the random sampling technique as applied in [5, 20].

1.2 NP/poly $\cap$ co-NP/poly *and subclasses*

As shown in [23], self-reducible sets in (NP $\cap$ co-NP)/poly are *low* for $\mathrm{ZPP}^{\mathrm{NP}}$. However, there are technical difficulties due to which this result does not seem to carry over to NP/poly$\cap$co-NP/poly. The best known collapse consequence of NP $\subseteq$ NP/poly$\cap$co-NP/poly (equivalently, NP $\subseteq$ co-NP/poly) is PH $\subseteq$ $\mathrm{ZPP}(\Sigma_2^p)$ [23].

In order to better understand this aspect of NP/poly $\cap$ co-NP/poly the authors of [12] introduce two interesting subclasses of NP/poly $\cap$ co-NP/poly which we discuss in Section 5.

In this paper, we notice, firstly, that NP/poly$\cap$co-NP/poly and the above-mentioned subclasses are intimately related to the function classes NPMV/poly, NPSV/poly, NPMV_t/poly, and NPSV_t/poly, which are nonuniform analogues of the function classes NPMV, NPSV, NPMV_t, and NPSV_t introduced and studied by Selman and other researchers [30, 13]. More precisely, we note that $A \in$ (NP $\cap$ co-NP)/poly if and only if $\chi_A \in \mathrm{NPSV}_t$/poly, where χ_A denotes the characteristic function of a language A. Similarly, $A \in$ NP/poly $\cap$ co-NP/poly if and only if $\chi_A \in$ NPMV/poly. Likewise, NPSV/poly and NPMV_t/poly capture the two new subclasses of NP/poly $\cap$ co-NP/poly defined in [12].

We prove the following new lowness results for these classes:

- We show that all self-checkable sets — in the program-checking sense of Blum and Kannan [9] — whose characteristic functions are in NPSV/poly are low for AM, and hence for $\mathrm{ZPP}^{\mathrm{NP}}$.
- We show that self-reducible sets whose characteristic functions are in the function class NPMV_t/poly are low for Σ_2^p (this result is essentially the lowness result underlying the collapse consequence derived in [12, Theorem 5.2]).
- Similarly, we show that all self-checkable sets whose characteristic functions are in NPMV/poly are low for Σ_2^p.

We give applications of these lowness results in the paper.

2 Preliminaries

Let $\Sigma = \{0, 1\}$. Σ^* denotes the set of all words over Σ and $\Sigma^{\leq n}$ denotes the set of all words of length at most n. We denote the cardinality of a set X by $\|X\|$ and the length of a string $x \in \Sigma^*$ by $|x|$. The characteristic function of a language $L \subseteq \Sigma^*$ is denoted by χ_L. $L^{\leq n}$ denotes the set $L \cap \Sigma^{\leq n}$. The restriction of χ_L to strings of length n can be considered as an n-ary boolean function that we denote by $L^{=n}$. Let $\mathcal{N}$ denote the set of natural numbers and let $\mathcal{R}$ denote the set of real numbers.

The definitions of standard complexity classes like P, NP, E, EXP etc. can be found in standard books [8, 26]. A relativized complexity class $\mathcal{C}$ with oracle A is denoted by either $\mathcal{C}^A$ or $\mathcal{C}(A)$. Likewise, we denote an oracle Turing machine M with oracle A by M^A or $M(A)$.

For a class $\mathcal{C}$ of sets and a class $\mathcal{F}$ of functions from 1^* to Σ^*, let $\mathcal{C}/\mathcal{F}$ [17] be the class of sets A such that there is a set $B \in \mathcal{C}$ and a function $h \in \mathcal{F}$ such that for all $x \in \Sigma^*$,

$$x \in A \Leftrightarrow \langle x, h(1^{|x|}) \rangle \in B.$$

The function h is called an *advice function* for A.

The self-reducibility considered in this paper is word-decreasing self-reducibility. We first recall the definition of word-decreasing self-reducible sets (and define its obvious extension to total single-valued functions).

Definition 1 [7] *For strings $x, y \in \Sigma^*$, $x \prec y$ if $|x| < |y|$ or $|x| = |y|$ and x is lexicographically smaller than y. A set A is* word-decreasing self-reducible *if there is a polynomial-time oracle machine M such that $A = L(M^A)$, where on any input x the machine M queries the oracle only about strings y such that $y \prec x$. Similarly, a total single-valued function f on Σ^* is* word-decreasing self-reducible *if there is a polynomial-time oracle transducer T such that T^f computes f, where on any input x, transducer T can query the oracle only about strings y such that $y \prec x$.*

We next recall definitions of AM and MA. A language L is in AM if there exist a polynomial p and a set $B \in$ P such that for all x, $|x| = n$,

$$\begin{aligned} x \in A &\Rightarrow \mathrm{Prob}_{r \in_R \{0,1\}^{p(n)}}[\exists y, |y| = p(n) : \langle x, y, r \rangle \in B] = 1, \\ x \notin A &\Rightarrow \mathrm{Prob}_{r \in_R \{0,1\}^{p(n)}}[\forall y, |y| = p(n) : \langle x, y, r \rangle \in B] \leq 1/4. \end{aligned}$$

A language L is in MA if there exist a polynomial p and a set $B \in$ P such that for all x, $|x| = n$,

$$\begin{aligned} x \in A &\Rightarrow \exists y, |y| = p(n) : \mathrm{Prob}_{r \in_R \{0,1\}^{p(n)}}[\langle x, y, r \rangle \in B] \geq 3/4, \\ x \notin A &\Rightarrow \forall y, |y| = p(n) : \mathrm{Prob}_{r \in_R \{0,1\}^{p(n)}}[\langle x, y, r \rangle \in B] \leq 1/4. \end{aligned}$$

Notice that we have taken the definition of AM with 1-sided error, known to be equivalent to AM with 2-sided error. Definitions for single and multiprover interactive proof systems can be found in standard texts, e.g. [26]. Let MIP denote the class of languages with multiprover interactive protocols and IP denote the class of languages with single-prover interactive protocols. We denote by MIP[$\mathcal{C}$] and IP[$\mathcal{C}$] the respective language classes where the prover complexity, i.e. the complexity of the honest prover for the interactive protocol, is bounded by FP($\mathcal{C}$), which is the class of functions that can be computed by a polynomial-time oracle transducer with oracle in $\mathcal{C}$. We define MIP[A] analogously: in this case the prover complexity is bounded by FP^A.

3 $\mathrm{AM} \cap \mathrm{coAM}$ is low for $\mathrm{ZPP}^{\mathrm{NP}}$

In this section we show that $\mathrm{AM}\cap\mathrm{coAM}$ is low for $\mathrm{ZPP}^{\mathrm{NP}}$. It follows that Graph Isomorphism and a host of group-theoretic problems known to be in $\mathrm{AM}\cap\mathrm{coAM}$ [3] are all low for $\mathrm{ZPP}^{\mathrm{NP}}$. We recall here that it is already known that $\mathrm{AM}\cap\mathrm{coAM}$ is low for Σ_2^p [29] and also for AM [21].

We notice first that although $\mathrm{AM} \cap \mathrm{coAM} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$ (because $\mathrm{AM} \subseteq \mathrm{coR}^{\mathrm{NP}}$ and the equality $\mathrm{ZPP} = \mathrm{R} \cap \mathrm{coR}$ relativizes) and $\mathrm{AM} \cap \mathrm{coAM}$ is low for itself, it doesn't follow that $\mathrm{AM} \cap \mathrm{coAM}$ is low for $\mathrm{ZPP}^{\mathrm{NP}}$. As mentioned before, $\mathrm{NP} \cap \text{co-NP}$ is trivially low for NP but is not known to be low for PP or MA.

Theorem 2 $\mathrm{AM} \cap \mathrm{coAM}$ *is low for* $\mathrm{ZPP}^{\mathrm{NP}}$.

Proof. Let L be any set in $\mathrm{AM} \cap \mathrm{coAM}$. We need to show that a given $\mathrm{ZPP}^{\mathrm{NP}^L}$ machine M can be simulated in $\mathrm{ZPP}^{\mathrm{NP}}$. Consider an input x of length bounded by n to the machine M. Suppose the lengths of all the queries made to L during the computation are bounded by m. Since $L \in \mathrm{AM} \cap \mathrm{coAM}$, it follows from standard probability amplification techniques (cf. [29]) that there are NP sets A and B, a polynomial p, and subsets $S_m \subseteq \{0,1\}^{p(m)}$ of size $\|S_m\| \geq 2^{p(m)-1}$ such that for all m and all strings y of length at most m,

$$y \in L \quad \Rightarrow \quad \forall w : \langle y, w\rangle \in A \text{ and } \forall w \in S_m : \langle y, w\rangle \notin B$$

$$y \notin L \quad \Rightarrow \quad \forall w : \langle y, w\rangle \in B \text{ and } \forall w \in S_m : \langle y, w\rangle \notin A.$$

In other words, a candidate advice $w \in \Sigma^{p(m)}$ is *not* in S_m iff it satisfies the following NP predicate:

$$\exists y \in \Sigma^{\leq m} : \langle y, w\rangle \in A \cap B.$$

Notice that in the above we are using the fact that AM protocols can be assumed to have one-sided error. It is clear that the w's in S_m act as advice strings using which membership in L for strings of length m can be decided with an $\mathrm{NP}\cap\text{co-NP}$ computation. Notice, however, that it would be incorrect for us to claim from here that $L \in (\mathrm{NP} \cap \text{co-NP})/\mathrm{poly}$, because if we use a string from $\{0,1\}^{p(m)} - S_m$ as advice, the resulting combination of machines for A and B may not yield an $\mathrm{NP} \cap \text{co-NP}$ computation for some input $y \in \Sigma^{\leq m}$.

We now describe the $\mathrm{ZPP}^{\mathrm{NP}}$ machine N that simulates the given $\mathrm{ZPP}^{\mathrm{NP}^L}$ machine M on some input x. Machine N first randomly guesses an advice string in $w \in \Sigma^{p(m)}$ which, by assumption, is in S_m with probability $1/2$. A single NP query using the above NP predicate is now used to certify that $w \in S_m$. Using such a w as advice, N can replace the oracle L with an $\mathrm{NP} \cap \text{co-NP}$ computation when it simulates M. ∎

Corollary 3 *Graph Isomorphism is low for* $\mathrm{ZPP}^{\mathrm{NP}}$.

The above corollary follows since Graph Isomorphism is in $\mathrm{AM}\cap\mathrm{coAM}$ [14]. The lowness result also holds for various group-theoretic problems known to be in $\mathrm{AM} \cap \mathrm{coAM}$ [3].

Notice that the previous theorem essentially shows that we can simulate $\mathrm{AM} \cap \mathrm{coAM}$ with an $\mathrm{NP} \cap \text{co-NP}$ computation using a random string in a co-NP set as advice for the computation. This observation combined with the result of [23] (that self-reducible sets in $(\mathrm{NP} \cap \text{co-NP})/\mathrm{poly}$ are low for $\mathrm{ZPP}^{\mathrm{NP}}$) immediately yields the following corollary.

Corollary 4 *Word-decreasing self-reducible sets in* $(\mathrm{AM} \cap \mathrm{coAM})/\mathrm{poly}$ *are low for* $\mathrm{ZPP}^{\mathrm{NP}}$.

Additionally, we also have the following corollary in the average-case complexity setting. We recall the definitions of polynomial-time computable distributions and the class $\mathcal{AP}$ (see, e.g. [22] for a detailed treatment): Let μ be a probability distribution on Σ^*. More precisely, μ denotes the distribution function for a density function μ', i.e., $\sum_x \mu'(x) = 1$ and $\mu(y) = \sum_{x \leq y} \mu'(x)$. We say that μ is a polynomial-time computable distribution if there is a polynomial time bounded transducer M such that $|M(x, 1^k) - \mu(x)| \leq 2^{-k}$, where $M(x, 1^k)$ is a rational number output as a pair of integers. $\mathcal{AP}$ is the class of decision problems A such that for every polynomial-time computable distribution there is an algorithm that decides A and is polynomial-time on the average for that distribution.

Corollary 5 *If* $\mathrm{NP} \subseteq \mathcal{AP}$ *then* $\mathrm{AM} \cap \mathrm{coAM} = \mathrm{NP} \cap \text{co-NP}$.

The proof follows from the assumption $\mathrm{NP} \subseteq \mathcal{AP}$ combined with the fact that for any set in $\mathrm{AM} \cap \mathrm{coAM}$ a large fraction of strings satisfying a co-NP predicate are good advice strings, as we have already seen in the proof of Theorem 2. Thus, a ZPP computation can randomly guess such an advice string and use an $\mathcal{AP}$ algorithm for the *uniform* distribution to decide the co-NP predicate. This $\mathcal{AP}$ algorithm, with its running time truncated to a suitable polynomial bound, will still accept many of the randomly picked good advice strings. This is an application of ideas from [22].

4 IP[P/poly] is low for $\mathrm{ZPP}^{\mathrm{NP}}$

The class IP[P/poly] already figures, though implicitly, in the proof of the result in [6] that if $\mathrm{EXP} \subseteq \mathrm{P/poly}$ then $\mathrm{EXP} = \mathrm{MA}$. We quickly recall the proof: Suppose $\mathrm{EXP} \subseteq \mathrm{P/poly}$. Note that each language in EXP has a multiprover interactive protocol with the prover in EXP [4]. By assumption, therefore, the honest provers can be simulated by polynomial size circuits. Thus the (MIP) protocol can be simulated by an MA protocol where Merlin simply sends the circuits for the provers to Arthur in the first round. In other words, the proof shows the inclusion chain $\mathrm{EXP} \subseteq \mathrm{MIP[P/poly]} \subseteq \mathrm{MA}$ as a consequence of $\mathrm{EXP} \in \mathrm{P/poly}$. Since the MA protocol is a single prover interactive protocol, we also have $\mathrm{MIP[P/poly]} = \mathrm{IP[P/poly]} \subseteq \mathrm{MA}$. Notice that, as observed in [2], $\mathrm{EXP} \subseteq \mathrm{P/poly}$ actually implies $\mathrm{EXP} = \mathrm{IP[P/poly]}$.

The above collapse consequence result of [6] motivates the study of lowness properties of IP[P/poly]. We show next that $\mathrm{IP[P/poly]} \subseteq \mathit{Low}(\mathrm{ZPP}^{\mathrm{NP}})$, improving the containment $\mathrm{IP[P/poly]} \subseteq \mathit{Low}(\Sigma_2^p)$ shown in [2]. Our result strengthens the result of [20] that NP sets in P/poly with self-computable witnesses are low for $\mathrm{ZPP}^{\mathrm{NP}}$. IP[P/poly] contains such NP sets, but IP[P/poly] may not even be contained in NP. Although $\mathrm{IP[P/poly]} \subseteq \mathrm{MA} \subseteq \mathrm{AM}$, IP[P/poly] is not known to be closed under complement, and it is not known if IP[P/poly] is contained in coAM. Thus, $\mathrm{IP[P/poly]} \subseteq \mathit{Low}(\mathrm{ZPP}^{\mathrm{NP}})$ appears incomparable to $\mathrm{AM} \cap \mathrm{coAM} \subseteq \mathit{Low}(\mathrm{ZPP}^{\mathrm{NP}})$ shown in Theorem 2 in the previous section. Our result is also incomparable to the result in [23] that self-reducible sets in P/poly are low for $\mathrm{ZPP}^{\mathrm{NP}}$. An interesting aspect of our proof is that it combines derandomization and almost uniform random sampling.

As preparation we recall some properties of universal hashing: let $\mathcal{L}(m, k)$ denote the class of all linear functions from Σ^m to Σ^k, where Σ^m and Σ^k are interpreted as m and k-dimensional vector spaces over GF[2], respectively. We recall a useful folklore lemma (as stated in [20]) that lower bounds the probability that a random $h \in \mathcal{L}(m, k)$ *isolates* some x in a given set S of appropriate size (meaning that x is the only element in S such that $h(x) = 0^k$). The lemma also upper bounds the probability that such an x belongs to a given small subset S' of S.

Lemma 6 [20] *Let $S \subseteq \Sigma^m - \{0^m\}$ be a nonempty set of size s, let $S' \subseteq S$ be of size at most $s/6$, and let $k \in \mathcal{N}$ such that $2^k < 3s \leq 2^{k+1}$. Then, for $h \in \mathcal{L}(m, k)$ chosen uniformly at random,*

- *with probability at least $2/9$, there is a unique $x \in S$ such that $h(x) = 0^k$, and*
- *with probability at most $1/9$, there exists some $x \in S'$ such that $h(x) = 0^k$.*

We recall definitions and results on derandomization [25]. For a positive real number $s \in \mathcal{R}^+$, $\mathcal{CIR}(n, s)$ denotes the class of all boolean functions $f : \{0,1\}^n \to \{0,1\}$ that can be computed by deterministic circuits of size at most s. Furthermore, for a function $s : \mathcal{N} \to \mathcal{R}^+$ let $\mathcal{CIR}(s) = \bigcup_{n \geq 0} \mathcal{CIR}(n, s(n))$.

Definition 7 *(cf.* [25]*) A boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be $\mathcal{CIR}(n, r(n))$-hard if*

$$\frac{1}{2} - \frac{1}{r(n)} < \frac{\|\{x \in \{0,1\}^n \mid f(x) = g(x)\}\|}{2^n} < \frac{1}{2} + \frac{1}{r(n)}$$

holds for all functions $g \in \mathcal{CIR}(n, r(n))$.
Let $r : \mathcal{N} \to \mathcal{R}^+$ and let L be any language. L is said to be $\mathcal{CIR}(r)$-hard *if for all but finitely many n, the n-ary boolean function $L^{=n}$ is $\mathcal{CIR}(n, r(n))$-hard.*

Let p, l, m, k be positive integers. A collection $D = (D_1, \ldots, D_p)$ of sets $D_i \subseteq \{1, \ldots, l\}$ is called a (p, l, m, k)*-design* if $\|D_i\| = m$ for all i, and for all $i \neq j$, $\|D_i \cap D_j\| \leq k$. Using D we get from a boolean function $g : \{0,1\}^m \to \{0,1\}$ a sequence of boolean functions $g_i : \{0,1\}^l \to \{0,1\}$, $i = 1, \ldots, p$, defined as $g_i(s_1, \ldots, s_l) = g(s_{i_1}, \ldots, s_{i_m})$ where $D_i = \{i_1, \ldots, i_m\}$. By concatenating the values of these functions we get a function $g_D : \{0,1\}^l \to \{0,1\}^p$ where $g_D(s) = g_1(s) \ldots g_p(s)$. Nisan and Wigderson show [25, Lemma 2.4] that the output of g_D looks random to a small deterministic circuit, provided g is hard to approximate by deterministic circuits of a certain size (in other words, the hardness of g implies that the pseudorandom generator g_D is secure against small circuits). The following makes this more precise.

Lemma 8 [25] *Let D be a (p, l, m, k)*-design *and let $g : \{0,1\}^m \to \{0,1\}$ be a $\mathcal{CIR}(m, p^2 + p2^k)$-hard function. Then the function $g_D : \{0,1\}^l \to \{0,1\}^p$ has the property that for every p-input circuit c of size at most p^2,*

$$\left| \mathrm{Prob}_{y \in_R \{0,1\}^p}[c(y) = 1] - \mathrm{Prob}_{s \in_R \{0,1\}^l}[c(g_D(s)) = 1] \right| \leq 1/p.$$

Next, we recall the main theorem of [25]:

Theorem 9 [25] *For all $\alpha > 0$, if* E *has a language that is $\mathcal{CIR}(2^{\alpha n})$-hard, then* BPP = P.

We also need the following folklore lemma which states that most boolean functions are hard on the average (see e.g. [24]).

Lemma 10 *For each α such that $0 < \alpha < 1/3$, there is a constant n_0 such that for all $n \geq n_0$ the number of n-ary boolean functions that are not $\mathcal{CIR}(n, 2^{\alpha n})$-hard is at most $2^{2^n} \cdot e^{-2^{n/4}}$.*

We now present the proof of lowness of the class IP[P/poly] for $\mathrm{ZPP}^{\mathrm{NP}}$.

Theorem 11 IP[P/poly] *is low for* $\mathrm{ZPP}^{\mathrm{NP}}$.

Proof. For $L \in$ IP[P/poly] we need to show that given a $\mathrm{ZPP}^{\mathrm{NP}^L}$ machine M there is a $\mathrm{ZPP}^{\mathrm{NP}}$ machine N that accepts the same language. There is a polynomial q such that for all inputs x, $q(|x|)$ bounds the length of the queries made by $M(x)$ to L. Let x be a length n_0 input to M. We fix n_0 and let n denote $q(n_0)$. In the design of N, we will have two preprocessing steps which are both $\mathrm{ZPP}^{\mathrm{NP}}$ computations. The preprocessing steps will correctly compute a polynomial-size circuit for $L^{\leq n}$ which can be used to replace the oracle in machine M to complete the proof. For the rest of the proof we fix the input x to machine M.

As observed before, each language in IP[P/poly] is in MA via the following protocol: Merlin (the prover) first sends to Arthur (the verifier) a polynomial-size circuit for the honest prover. Arthur uses this circuit to simulate the IP[P/poly] interactive protocol for the given language. This is simply a randomized BPP-like computation. More precisely, for $L \in$ IP[P/poly] there are a polynomial p and a set $A \in$ P such that for all n,

$$\exists w \in \{0,1\}^{p(n)} \forall y \in L^{\leq n} \ : \ \mathrm{Prob}_{r \in_R \{0,1\}^{p(n)}}[\langle y, w, r\rangle \in A] \geq 3/4$$

and

$$\forall w \in \{0,1\}^{p(n)} \forall y \in \Sigma^{\leq n} - L^{\leq n} \ : \ \mathrm{Prob}_{r \in_R \{0,1\}^{p(n)}}[\langle y, w, r\rangle \in A] \leq 1/4.$$

To proceed further, we use the above MA protocol for L. For a pair y, w, the decision procedure for A can be seen as a circuit $C_{y,w}$ that takes r as input. We can assume w.l.o.g that $C_{y,w}$ has size bounded by $p^2(n)$. Using Lemma 8 we have for any (p, l, m, k)-design D and any $\mathcal{CIR}(m, p^2 + p2^k)$-hard boolean function $g : \{0,1\}^m \to \{0,1\}$ that

$$\left|\mathrm{Prob}_{r \in_R \{0,1\}^p}[c(r) = 1] - \mathrm{Prob}_{s \in_R \{0,1\}^l}[c(g_D(s)) = 1]\right| \leq 1/p$$

holds for every p-input circuit c of size at most p^2. Now let $m(n) = 12 \log p(n)$, $l(n) = 288 \log p(n)$, and $k(n) = \log p(n)$. By Lemma 10 we know that for all sufficiently large n, a randomly chosen boolean function $g : \{0,1\}^{m(n)} \to \{0,1\}$ is $\mathcal{CIR}(m(n), 2^{m(n)/4})$-hard (and thus $\mathcal{CIR}(m(n), p(n)^2 + p(n)2^{k(n)})$-hard) with probability at least $1 - e^{-2^{m(n)/4}}$.

The machine N performs the following preprocessing step:

input x, $|x| = n_0$;
compute a $(p(n), l(n), m(n), k(n))$-design D;
repeat
  choose randomly $g : \{0,1\}^{m(n)} \to \{0,1\}$
until g is $\mathcal{CIR}(m(n), 2^{m(n)/4})$-hard {this can be decided by an NP oracle}
compute the pseudorandom strings $r_1, \ldots, r_{2^{l(n)}}$ of g_D on all seeds;

By the property of these pseudorandom strings $r_1, \ldots, r_{2^{l(n)}}$ with respect to circuits $C_{y,w}$, we have for all n that

$$\exists w \in \{0,1\}^{p(n)} \forall y \in L^{\leq n} \ : \ \|\{i | \langle y, w, r_i\rangle \in A\}\| \geq 2^{l(n)-1}$$

and

$$\forall w \in \{0,1\}^{p(n)} \forall y \in \Sigma^{\leq n} - L^{\leq n} \ : \ \|\{i | \langle y, w, r_i\rangle \in A\}\| < 2^{l(n)-1}.$$

For each n, therefore, we can now efficiently build a polynomial-size circuit C_n (in which the pseudorandom strings $r_1, \ldots, r_{2^{l(n)}}$ are hard-wired) such that

$$\exists w \in \{0,1\}^{p(n)} \forall y \in L^{\leq n} \ : \ C_n(y, w) = 1$$

and

$$\forall w \in \{0,1\}^{p(n)} \forall y \in \Sigma^{\leq n} - L^{\leq n} \ : \ C_n(y, w) = 0.$$

Notice that $\{C_n\}_{n>0}$ is a *uniform* circuit family, where each C_n takes an input y and advice string w to decide y's membership in L. The above property guarantees that the advice strings w have only 1-sided error.

We now proceed to the next step of machine N in which it performs a $\mathrm{ZPP}^{\mathrm{NP}}$ computation to compute with high probability a polynomial-size deterministic circuit $\hat{c}$ that decides L correctly for inputs of length upto n. In fact, each output circuit $\hat{c}_W$ will be constructed from a set W of polynomially many advice strings in $\Sigma^{p(n)}$. Stated formally, for all $y \in \Sigma^{\leq n}$,

$$\hat{c}_W(y) = 1 \iff \exists w \in W \ : \ C_n(w, y) = 1.$$

By virtue of the 1-sided correctness of C_n, $\hat{c}_W$ rejects all $y \in \Sigma^{\leq n} - L^{\leq n}$ for any W.

We need one more notation: For $S \subseteq L^{\leq n}$, define the active advice set $W(S)$ to be

$$W(S) := \{w \in \Sigma^{p(n)} \mid \forall y \in S : C_n(w, y) = 1\}.$$

Notice that $W(S)$ consists of all correct advice strings for S.

During the second preprocessing step, machine N iteratively includes strings $y \in L^{\leq n}$ into S, until it finds a circuit $\hat{c}_W$ for $L^{\leq n}$. N aims to extend S with an y such that $\|W(S)\|$ decreases by a constant factor. To ensure this, N randomly picks $9n$ hash functions h_i and computes the set W of (at most $9n$) advice strings that are isolated in $W(S)$ by some h_i. Then N includes in S the lexicographically least $y \in L^{\leq n}$ such that $\forall w \in W : \ C_n(w, y) = 0$. We now formally describe the second part of N:

$S := \emptyset$;
loop
  choose randomly $k \in \{1, \ldots, p(n)\}$;
  choose randomly $h_1, \ldots, h_{9n}$ from $\mathcal{L}(p(n), k)$;
  $W := \{w \in \Sigma^{p(n)} \mid$ some h_i isolates w within $W(S)\}$
  if $\hat{c}_W(y) = 0$ for some $y \in L^{\leq n}$ **then** $S := S \cup \{y\}$
  else exit(**loop**) **end**
end loop;
Use $\hat{c}_W$ to answer the oracle queries to L when simulating $M(x)$

Clearly, N can be implemented with access to an NP oracle. Moreover, since $\hat{c}_W$ never accepts an $y \notin L^{\leq n}$ and since the loop only terminates outputting $\hat{c}_W$ when $\hat{c}_W$ accepts all $y \in L^{\leq n}$, the algorithm is correct. It only remains to show that the expected running time of N is polynomially bounded.

Consider a specific stage of the loop iteration. Call $y \in L^{\leq n}$ a *good* extension of S if $\|W(S)\|$ decreases by a constant factor, say, $\|W(S \cup \{y\})\| < (5/6)\|W(S)\|$. Let A denote the event that $2^k < 3\|W(S)\| \leq 2^{k+1}$ holds for the randomly picked k. Clearly, A holds with probability $\frac{1}{p(n)}$. We claim that $p_S = \mathrm{Prob}_{h_1,\ldots,h_{9n}}[$ a good extension of S is obtained $\mid A\ ] \geq 1/2$.

To see this, let $BAD = \{y \in L^{\leq n} \mid \|W(S) - W(S \cup \{y\})\| \leq \|W(S)\|/6\}$ denote the set of bad extensions of S. For $y \in BAD$, let p_y be the probability that S is extended by y conditioned on event A. Notice that $1 - p_S \leq \sum_{y \in BAD} p_y$. Now, note that y is added to S only if $\hat{c}_W(y) = 0$. Thus, p_y is bounded by the probability that $\hat{c}_W(y) = 0$. Note that $\hat{c}_W(y) = 0$ if none of $h_1, h_2, \ldots, h_{9n}$ isolates within $W(S)$ an advice string $w \in W(S \cup \{y\})$. By Lemma 6, a single h_i isolates some advice string in $W(S)$ with probability greater than $2/9$, and, moreover, h_i isolates an advice string in $W(S) - W(S \cup \{y\})$ with probability at most $1/9$. Thus, with probability at least $1/9$, each h_i isolates an advice string in $W(S \cup \{y\})$. So, the probability that none of $h_1, \ldots, h_{9n}$ isolates an advice string in $W(S \cup \{y\})$ is at most $(8/9)^{9n} < e^{-n}$. Hence, $p_y \leq e^{-n}$ for each $y \in BAD$. Since $||L^{\leq n}|| \leq 2^{n+1}$ we get $1 - p_S \leq \sum_{y \in BAD} p_y \leq 2(2/e)^n$. Thus, $p_S \geq 1/2$ for large enough n. Therefore, the probability that a single extension of S is good is at least $\frac{1}{2p(n)}$ (since $\mathrm{Prob}[A] = \frac{1}{p(n)}$).

The size of $W(S)$ is $2^{p(n)}$ at the start of $N(x)$'s computation. Since $W(S)$ is always nonempty, there can be at most $p(n) \log^{-1}(6/5) < 4p(n)$ successful extensions of S. Hence, it follows that the expected number of loop iterations is at most $8p^2(n)$. ■

The above lowness result easily extends to IP[(NP $\cap$ co-NP)/poly] by observing that the proof relativizes in the following sense: for any oracle set A, $\mathrm{NP}^{\mathrm{IP}[\mathrm{P}^A/\mathrm{poly}]} \subseteq \mathrm{ZPP}^{\mathrm{NP}^A}$.

We conclude this section with another connection to the average-case complexity setting.

Theorem 12 *If* $\mathrm{NP} \subseteq \mathcal{AP}$ *and* $\mathrm{NP} \subseteq \mathrm{P/poly}$ *then PH collapses to* Δ_2^p.

Proof. Recall from [23, 5] that if $\mathrm{NP} \subseteq \mathrm{P/poly}$ then PH collapses to $\mathrm{ZPP}^{\mathrm{NP}}$. At the heart of this collapse result is the following $\mathrm{FZPP}^{\mathrm{NP}}$ computation: on input 0^n it outputs with high probability a polynomial-size circuit for length n instances of SAT. Since NP $\subseteq$ P/poly by assumption, the NP oracle in the above $\mathrm{FZPP}^{\mathrm{NP}}$ computation can be replaced by an appropriate polynomial-size circuit. Thus, given access to a hard boolean function we can use the Nisan-Wigderson generator to derandomize the above $\mathrm{FZPP}^{\mathrm{NP}}$ computation: Observe that derandomization here implies that the output of the pseudorandom generator will include a pseudorandom string that is an accepting computation of the $\mathrm{FZPP}^{\mathrm{NP}}$ computation. Thus, given access to a hard boolean function the $\mathrm{FZPP}^{\mathrm{NP}}$ computation can be derandomized to an $\mathrm{FP}^{\mathrm{NP}}$ computation.

Now, as argued in the proof of the previous theorem, we can use a $\mathrm{ZPP}^{\mathrm{NP}}$ computation to guess a hard boolean function and then verify that it is hard with a single co-NP query. At this point, we can use the assumption that NP $\subseteq \mathcal{AP}$, as in [22] and Corollary 5, to get rid of the NP oracle and replace this $\mathrm{ZPP}^{\mathrm{NP}}$ computation with an ZPP computation. Finally, notice that the lexicographically first output of this ZPP computation can be computed by an $\mathrm{FP}^{\mathrm{NP}}$ computation. Thus it is possible to compute a polynomial-size circuit for $\mathrm{SAT}^{\leq n}$ by an $\mathrm{FP}^{\mathrm{NP}}$ computation and consequently PH collapses to Δ_2^p. ∎

5 Nonuniform function classes and lowness

We now study lowness properties of checkable and self-reducible functions in NPMV/poly, NPSV/poly, NPMV_t/poly, and NPSV_t/poly. These are nonuniform analogs of the function classes NPMV, NPSV, NPMV_t, and NPSV_t studied by Selman [30] and other researchers, e.g. [13]. We find these nonuniform classes interesting because when restricted to characteristic functions of sets, NPMV/poly coincides with NP/poly $\cap$ co-NP/poly and NPSV_t/poly coincides with (NP $\cap$ co-NP)/poly. Likewise, we note that the two subclasses of NP/poly $\cap$ co-NP/poly studied in [12], namely all sets underproductively reducible to sparse sets and all sets overproductively reducible to sparse sets, also coincide with NPSV/poly and NPMV_t/poly, respectively.

Following Selman's notation in [30], a transducer is an NDTM T with a write-only output tape. On input x machine T outputs $y \in \Sigma^*$ if there is an accepting path on input x along which y is output. Hence, the function defined by T on Σ^* could be multivalued and partial. Given a multivalued function f on Σ^* and $x \in \Sigma^*$ we use the notation

$$\textit{set-}f(x) = \{y \mid f : x \mapsto y\}$$

to denote the (possibly empty) set of function values for input x. We recall the basic definitions.

Definition 13 (cf. [10, 17])

1. NPMV *is the class of multivalued, partial functions f for which there is a polynomial-time NDTM N such that*

   (a) *$f(x)$ is defined (i.e., set-$f(x) \neq \emptyset$) if and only if $N(x)$ has an accepting path.*

   (b) *$y \in$ set-$f(x)$ if and only if there is an accepting path of $N(x)$ where y is output.*

2. NPSV *is the class of single-valued partial functions in* NPMV.

3. NPMV_t *is the class of total functions in* NPMV.

4. NPSV_t *is the class of total single-valued functions in* NPMV.

The classes NPMV/poly, NPSV/poly, NPMV_t/poly, *and* NPSV_t/poly *are defined following [17] as the nonuniform analogs of the above function classes: for a class $\mathcal{F}$ of (possibly multivalued and partial) functions, $\mathcal{F}$/poly consists of all functions f such that there is a function $g \in \mathcal{F}$, a polynomial q, and an* advice function $h : 1^* \mapsto \Sigma^*$ *such that for all m, $|h(1^m)| = q(m)$ and for all $x \in \Sigma^{\leq m}$,*

$$\textit{set-}f(x) = \textit{set-}g(\langle x, h(1^m)\rangle).$$

Remark: Notice that single-valued partial functions in NPMV are already in NPSV. On the other hand, if a single-valued partial function f is in NPMV/poly, the nondeterministic machine computing f need be single-valued only for the correct advice. Thus, it does not follow that every single-valued function in NPMV/poly is in NPSV/poly.

Before we connect these classes to NP/poly∩co-NP/poly and its subclasses [12], we recall definitions from [12]: Consider polynomial-time nondeterministic oracle machines N whose computation paths can have three possible outcomes: accept, reject, or **?**. The machine N can also be viewed as a transducer which computes, for given oracle D and input x, a multivalued function. More precisely, if we identify accept with output value 1 and reject with output value 0, and consider the **?** computation paths as paths without output then N^D defines a partial multivalued function: $\textit{set-}N^D(x) \subseteq \{0,1\}$. Machine N^D is said to be *underproductive* if for each x we have $\{0,1\} \not\subseteq \textit{set-}N^D(x)$, and N is said to be *robustly underproductive* if for each oracle D and input x we have $\{0,1\} \not\subseteq \textit{set-}N^D(x)$. Likewise, N^D is *overproductive* if for each x we have $\textit{set-}N^D(x) \neq \emptyset$, and N is said to be *robustly overproductive* if for each oracle D and input x we have $\textit{set-}N^D(x) \neq \emptyset$.

With standard arguments we can convert a sparse set into a polynomial-size advice string and vice-versa (see, e.g. [8]). It follows that $A \in$ NP/poly ∩ co-NP/poly if and only if there is a sparse set S and a nondeterministic machine N such that N^S is both overproductive and underproductive and $A = L(N^S)$. Similarly, $A \in$ (NP ∩ co-NP)/poly if and only if there is a sparse set S and a nondeterministic machine N such that $A = L(N^S)$ and N is both robustly overproductive and robustly underproductive and $A = L(N^S)$.

Consider the characteristic function χ_A of a language A. We formally define $\textit{set-}\chi_A(x) = \{\chi_A(x)\}$, so that it makes sense in accordance with the above definitions to talk of χ_A being in a function class like NPMV/poly. The following result is an immediate consequence of the definitions.

Proposition 14 *Let χ_A denote the characteristic function for a set $A \subseteq \Sigma^*$:*

1. *χ_A is in* NPMV/poly *if and only if A is in* NP/poly ∩ co-NP/poly.

2. *χ_A is in* NPSV_t/poly *if and only if A is in* (NP ∩ co-NP)/poly.

3. χ_A *is in* NPSV/poly *if and only if there are a sparse set* S *and a robustly underproductive machine* N *such that* $A = L(N^S)$.

4. χ_A *is in* NPMV_t/poly *if and only if there are a sparse set* S *and a robustly overproductive machine* N *such that* $A = L(N^S)$.

By abuse of notation, we identify χ_A with A in this section. E.g. we write $A \in$ NPSV/poly when we mean $\chi_A \in$ NPSV/poly. We now turn to lowness questions for the nonuniform function classes. The classes NP/poly $\cap$ co-NP/poly and (NP $\cap$ co-NP)/poly are of interest in the context of deriving strong collapse consequences from the assumption that NP (or other hard complexity classes) is contained in one of these classes. We recall the known collapse consequence [23] result for NP/poly$\cap$co-NP/poly under the assumption that NP is contained therein: If NP $\subseteq$ NP/poly $\cap$ co-NP/poly then PH collapses to $\mathrm{ZPP}^{\Sigma_2^p}$. The open question here is whether the collapse consequence can possibly be improved to $\mathrm{ZPP}^{\mathrm{NP}}$. This is one reason to consider classes that lie between NP/poly $\cap$ co-NP/poly and (NP $\cap$ co-NP)/poly.

5.1 *A lowness result for* NPMV_t/poly

It is shown in [12] that if an NP-complete problem is in NPMV_t/poly then PH collapses to Σ_2^p. (In [12] the authors state this result in terms of overproductive reductions to sparse sets.) We use ideas in their proof to show the underlying lowness result for functions: all word-decreasing self-reducible functions in NPMV_t/poly are low for Σ_2^p.

The definition of lowness extends naturally to total, single-valued functions: A functional oracle f returns $f(x)$ on query x. For any relativizable complexity class $\mathcal{C}$ we say that $f \in \mathit{Low}(\mathcal{C})$ if $\mathcal{C}^f = \mathcal{C}$. We show next that self-reducible sets and self-reducible functions in NPMV/poly have identical lowness properties. Hence it suffices to prove lowness of self-reducible sets in NPMV/poly.

Theorem 15 *Let* $\mathcal{F}$ *contain all self-reducible functions in any of the four function classes* $\{\mathrm{NPMV/poly}, \mathrm{NPSV/poly}, \mathrm{NPMV}_t\mathrm{/poly}, \mathrm{NPSV}_t\mathrm{/poly}\}$. *Let* $\mathcal{C}$ *be the subclass of* $\mathcal{F}$ *consisting of characteristic functions (making* $\mathcal{C}$ *a language class, essentially). For every self-reducible function* $f \in \mathcal{F}$ *there is a self-reducible set* $A \in \mathcal{C}$ *such that* f *and* A *are polynomial-time Turing equivalent.*

Proof. Given $f \in \mathcal{F}$, we can define the corresponding set $A \in \mathcal{C}_{\mathcal{F}}$ by suitably encoding, for each x, the bits of $f(x)$ in A. We can easily ensure that the self-reducibility of f carries over to A and that f and A are polynomial-time Turing equivalent. ∎

Theorem 16 *Word-decreasing self-reducible sets in* NPMV_t/poly *are low for* Σ_2^p.

Proof. Let A be a word-decreasing self-reducible set in NPMV_t/poly. Let M_0 be the self-reduction machine for A. Consider a $\Sigma_2^p(A)$ machine M with oracle A. There is a polynomial p such that for inputs x of length n, $p(n)$ bounds the length of the queries made by $M(x)$ to A. We fix n and let m denote $p(n)$. Since $A \in \mathrm{NPMV}_t$/poly there exist a function

$g \in \mathrm{NPMV}_t$ and a polynomial q such that for all m there exists a string $w_m \in \Sigma^{q(m)}$ such that for all $z \in \Sigma^{\leq m}$,

$$set\text{-}\chi_A(z) = set\text{-}g(\langle z, w_m \rangle).$$

How hard is it to test that a candidate advice $w \in \Sigma^{q(m)}$ is good? The conjunction of the following two co-NP predicates does this task:

- We first define the co-NP predicate STRONG(w):

  $$\forall z \in \Sigma^{\leq m}\ \exists b \in \{0,1\}\ \forall y\ :\ T(z,w,y) \in \{\mathbf{?}, b\}$$

  where T is a nondeterministic transducer that computes g and $T(z,w,y)$ is the value output by T on computation path y, given input z and advice w (in case that $T(\langle z,w\rangle)$ rejects on computation path y, we define $T(z,w,y) = \mathbf{?}$). Notice that this co-NP predicate just verifies that T is single-valued for advice w. However, observe that advice w could still be incorrect. The next co-NP predicate checks correctness of w.

- For input $z \in \Sigma^{\leq m}$, let $M_0^w(z)$ denote the computation that results by simulating the self-reduction machine $M_0(z)$, where any query q is answered by simulating $T(q,w)$: the simulation of $T(q,w)$ aborts along a path resulting in $\mathbf{?}$. On other paths the simulation proceeds treating output 1 as accept and output 0 as reject. Notice that this simulation of $M_0^w(z)$ yields a nondeterministic *single-valued* computation if STRONG(w) holds, as STRONG(w) forces each simulation of the kind $T(q,w)$ to be single-valued for all q. When the simulation is complete along some path, $M_0^w(z)$ accepts on that path and outputs the value computed.

  We now define the co-NP predicate CORRECT(w):

  $$\begin{aligned}\mathrm{CORRECT}(w) \ :=\ & \text{for all } z \in \Sigma^{\leq m}, \text{ if } T(z,w) \text{ accepts then } M_0^w(z) \text{ never rejects,}\\ & \text{and if } T(z,w) \text{ rejects then } M_0^w(z) \text{ never accepts.}\end{aligned}$$

Notice that if STRONG(w) holds then CORRECT(w) checks that for all $z \in \Sigma^m$ the advice string w is consistent with the self-reducibility machine M_0. Thus it follows that a string $w \in \Sigma^{q(m)}$ is a correct advice string for $\Sigma^{\leq m}$ if and only if STRONG(w) $\wedge$ CORRECT(w) holds.

Continuing with the proof, recall that the computation tree for $M^A(x)$ has an $\exists$ layer followed by a $\forall$ layer. We denote this by saying that $M^A(x)$ accepts if and only if there exists a y such that $M^A(x,y)$ accepts on all paths, where $M^A(x,y)$ is the remaining computation which defines a co-NP^A predicate. Now it is easy to logically describe the Σ_2^p machine N that simulates M on an input x of length n. N accepts x if and only if the following Σ_2^p predicate holds:

$$\exists w \in \Sigma^{q(m)}\ \exists y\ :\ \mathrm{STRONG}(w)\ \wedge\ \mathrm{CORRECT}(w)\ \wedge\ M^w(x,y) \text{ accepts on all paths,}$$

where $M^w(x,y)$ represents the following computation: simulate $M^A(x,y)$, and for each query q made to A plug in the nondeterministic computation $T(q,w)$. If a computation path of $T(q,w)$ rejects then terminate that computation as accepting (because this path is irrelevant to the overall computation). If a computation path of $T(q,w)$ outputs a value different from 0 and 1, then terminate that computation as rejecting. If a computation path of $T(q,w)$ outputs 1 (interpreted as answering $q \in A$) or 0 (interpreted as answering $q \notin A$), machine $M^w(x,y)$ continues with the computation assuming that the answer is correct. Continuing in this manner $M^w(x,y)$ finally accepts or rejects on each computation path.

To see correctness, notice first that N accepts $x \in \Sigma^{\leq n}$ if M^A accepts x, because for the good advice string w and for y such that $M^A(x,y)$ accepts on all paths, $M^w(x,y)$ correctly simulates $M^A(x,y)$ and therefore accepts on all paths. Next, for a string $x \in \Sigma^{\leq n}$ that is rejected by M^A, notice that for the good advice string w there is no y such that $M^w(x,y)$ accepts on all paths. And for a bad advice string w, the co-NP predicate $\mathrm{STRONG}(w) \wedge \mathrm{CORRECT}(w)$ is false. This completes the proof. ∎

Since Σ_k^p, Π_k^p, PP, $\mathrm{C_=P}$, $\mathrm{Mod}_m\mathrm{P}$, PSPACE, and EXP have many-one complete word-decreasing self-reducible sets [7], we get the following corollary.

Corollary 17 *If $\mathcal{C} \in \{\Sigma_k^p, \Pi_k^p, \mathrm{PP}, \mathrm{C_=P}, \mathrm{Mod}_m\mathrm{P}, \mathrm{PSPACE}, \mathrm{EXP}\}$, for $k \geq 1$ and $m \geq 2$, has a hard set in $\mathrm{NPMV}_t/\mathrm{poly}$ then $\mathcal{C} \subseteq \Sigma_2^p$ and $\mathrm{PH} = \Sigma_2^p$.*

The proof follows since for each $\mathcal{C} \in \{\Sigma_k^p, \Pi_k^p, \mathrm{PP}, \mathrm{C_=P}, \mathrm{Mod}_m\mathrm{P}, \mathrm{PSPACE}, \mathrm{EXP}\}$ and any set A complete for $\mathcal{C}$ w.r.t. polynomial-time Turing reductions we have $\Sigma_3^p \subseteq \Sigma_2^A$.

We end this section with the observation that $\mathrm{AM} \cap \mathrm{coAM}$ is contained in $\mathrm{NPMV}_t/\mathrm{poly}$. It is interesting to now compare the lowness results (Theorems 2 and 16) for these classes.

Proposition 18 $\mathrm{AM} \cap \mathrm{coAM} \subseteq \mathrm{NPMV}_t/\mathrm{poly}$.

Proof. Given $L \in \mathrm{AM} \cap \mathrm{coAM}$, as already observed in the proof of Theorem 2, there are NP sets A and B, a polynomial p, and subsets $S_m \subseteq \{0,1\}^{p(m)}$ of size $\|S_m\| \geq 2^{p(m)-1}$ such that for all m and all strings x of length at most m we have,

$$\begin{aligned} x \in L \quad &\Rightarrow \quad \forall w : \langle x,w \rangle \in A \text{ and } \forall w \in S_m : \langle x,w \rangle \notin B, \\ x \notin L \quad &\Rightarrow \quad \forall w : \langle x,w \rangle \in B \text{ and } \forall w \in S_m : \langle x,w \rangle \notin A. \end{aligned}$$

We can combine the NP machines M_A and M_B for A and B and build a transducer I that on input $\langle x,w \rangle$ outputs 1 (0) on any accepting simulation of M_A (resp. M_B) on input $\langle x,w \rangle$. Observe that in case $w \in S_m$ transducer I will always yield a single-valued, total computation for all inputs x of length m, outputting either 1 or 0 depending on the membership of x. On the other hand, no matter which $w \in \{0,1\}^{p(m)}$ is used as advice, $\langle x,w \rangle$ is either in A or in B and so the transducer I always outputs at least one of 0 or 1 for any advice string $w \in \{0,1\}^{p(m)}$ and any input x of length m. Hence it follows that L is in $\mathrm{NPMV}_t/\mathrm{poly}$. ∎

5.2 *A lowness result for* NPSV/poly

In [12] it is left as an open problem to discover new lowness (or collapse consequence) results for NPSV/poly. As noted in [12], nothing better is known for NPSV/poly than the collapse consequence result: if SAT is in NPSV/poly then PH collapses to $\mathrm{ZPP}^{\Sigma_2^p}$, which holds even for the larger class NP/poly $\cap$ co-NP/poly [23].

We show that sets in NPSV/poly that are checkable, in the sense of program checking as defined by Blum and Kannan [9], are low for AM and for $\mathrm{ZPP}^{\mathrm{NP}}$. Since $\oplus$P, PP, PSPACE, and EXP have checkable complete problems, it follows that for any of these classes inclusion in NPSV/poly implies its containment in AM $\cap$ coAM. This result is proved on the same lines as the Babai et al result [6]: If EXP is contained in P/poly then EXP $\subseteq$ MA.

Recall the definitions of MIP[$\mathcal{C}$] and IP[$\mathcal{C}$] for a class $\mathcal{C}$ of languages. We prove a technical lemma that immediately yields the lowness result.

Lemma 19 *If $A \in$ NPSV/poly then* MIP[A] $\subseteq$ AM.

Proof. Let $L \in$ MIP[A] for some set $A \in$ NPSV/poly. Let T be the nondeterministic transducer that witnesses that $A \in$ NPSV/poly. We describe an MAM protocol for L:

1. Let x be an input of length n to the MIP protocol. Let $m = p(n)$, where p is a polynomial bounding the size of the queries to A made by the provers during the MIP protocol for inputs of length n.

2. **Merlin** sends advice w of length $q(m)$ to Arthur.

3. **Arthur** sends a polynomial random string r (used for simulating the original MIP protocol) to Merlin.

4. **Merlin** sends back the list of successive queries to set A (generated by simulating the original MIP protocol with random string r), the list of answers to those queries along with the computation paths of transducer T with advice w that certify the answers to the queries.

5. **Arthur** can verify in polynomial time that Merlin's message is all correct and accept if and only if the original MIP protocol accepts.

By the fact that T computes a single-valued partial function for any advice w, although Merlin is simulating the nondeterministic transducer T, it is guaranteed that each accepting computation path has identical output and hence does identical computation. Thus, what makes the above MAM protocol work is the fact that for any advice w and query q all accepting computation paths of $T(q, w)$ output the same value. So, regardless of which computation paths are sent to Arthur by Merlin in Step 4 of the above protocol, Arthur's decision will be the same. In other words, Arthur's acceptance depends only on the random string r, hence exactly preserving the acceptance probability of the original MIP protocol.

Standard techniques (cf. [3]) can be used to convert the MAM protocol to an AM protocol. This completes the proof. ∎

We have as immediate consequence the following lowness result.

Theorem 20 *If L is a checkable set in* NPSV/poly *(i.e. L reduces by a robustly underproductive machine to a sparse set) then $L \in$* AM $\cap$ coAM *and hence low for AM and* $\mathrm{ZPP}^{\mathrm{NP}}$.

Proof. The assumption in the theorem's statement implies that both L and $\overline{L}$ are in MIP$[L]$ by the checker characterization theorem of [9]. Now, applying Lemma 19 yields that both L and $\overline{L}$ are in AM and the result follows. ∎

We can derive new collapse consequences as corollary, since $\oplus$P, PP, PSPACE, and EXP have checkable complete problems [4, 9]. It follows that for any of these classes inclusion in NPSV/poly implies its containment in AM $\cap$ coAM.

Corollary 21 *If* $\mathcal{C} \in \{\oplus\mathrm{P}, \mathrm{PP}, \mathrm{PSPACE}, \mathrm{EXP}\}$ *is contained in* NPSV/poly *then* $\mathcal{C}$ *is contained in* AM $\cap$ coAM*, and hence* PH = AM.

Notice that we also have the same lowness for checkable functions in NPSV/poly.

Theorem 22 *Checkable functions in* NPSV/poly *are low for* AM *and* $\mathrm{ZPP}^{\mathrm{NP}}$*.*

Proof. Let f be a checkable function in NPSV/poly. We can suitably encode, for each x, the bits of $f(x)$ in a language A which is polynomial-time Turing equivalent to f and hence A is also checkable. The lowness result now follows by invoking Theorem 20. ∎

5.3 *A lowness result for* NPMV/poly

Finally, using the same ideas as in Lemma 19 and Theorem 20, we will prove the following lowness result for checkable sets in NPMV/poly.

Theorem 23 *Checkable sets in* NPMV/poly *(i.e. checkable sets in* NP/poly $\cap$ co-NP/poly*) are low for* Σ_2^p*.*

Proofsketch. Let $A \in$ NPMV/poly be a checkable set. It follows that both A and $\overline{A}$ are in MIP$[A]$. Our aim is to show that $A \in Low(\Sigma_2^p)$. Consider a Σ_2^A machine M with input x of length n. There is a polynomial p, such that the queries made by M to A on input x are of length bounded by $m = p(n)$. By definition of NPMV/poly, there is a function $g \in$ NPMV, a polynomial q, and an advice function $h : 1^* \mapsto \Sigma^*$ such that for all m, $|h(1^m)| = q(m)$ and for all $y \in \Sigma^{\leq m}$,

$$\textit{set-}\chi_A(y) = \textit{set-}g(\langle y, h(1^m)\rangle).$$

Let w be an advice string of appropriate length $q(m)$ for $A^{\leq m}$. Call the advice *safe* if for all $y \in \Sigma^{\leq m}$ either $g(\langle y, w\rangle)$ outputs 1 and ? on all computation paths, or $g(\langle y, w\rangle)$ outputs 0 and ? on all computation paths.

Thus, verifying if an advice string w is safe requires a co-NP computation. We now sketch the simulation of $M^A(x)$ by a Σ_2^p computation.

In this Σ_2^p computation, first an advice string w of length $q(m)$ is guessed. Consider the path along which a *safe* advice string w is guessed (at the end of the Σ_2^p computation we will verify that w is safe with a co-NP computation which will not increase the number of alternations).

If we restrict the advice strings to safe advice strings, notice from the definition that NPMV/poly computations will be essentially like NPSV/poly computations. The following claim is proved like Lemma 19, using the fact that both A and $\overline{A}$ are in MIP$[A]$. Crucially, we note that only if w is a safe advice string we get an MAM protocol (more precisely, step 4 which is the second Merlin move of the MAM protocol in Lemma 19, is correct only if w is safe). This MAM protocol (for safe w) has as usual an AM simulation by [3].

Claim. *There exist a polynomial p and sets $B, C \in \mathrm{P}$ such that for all m, all y of length at most m, and all safe advice strings $w \in \Sigma^{q(m)}$,*

$$\begin{array}{rcl} y \in A & \Rightarrow & \mathrm{Prob}_{r \in_R \{0,1\}^{p(m)}}[\exists z, |z| = p(m) : \langle y, w, z, r\rangle \in B] \ = \ 1 \textit{ and} \\ & & \mathrm{Prob}_{r \in_R \{0,1\}^{p(m)}}[\forall z, |z| = p(m) : \langle y, w, z, r\rangle \in C] \leq 1/4, \\ y \notin A & \Rightarrow & \mathrm{Prob}_{r \in_R \{0,1\}^{p(m)}}[\forall z, |z| = p(m) : \langle y, w, z, r\rangle \in B] \leq 1/4 \textit{ and} \\ & & \mathrm{Prob}_{r \in_R \{0,1\}^{p(m)}}[\exists z, |z| = p(m) : \langle y, w, z, r\rangle \in C] \ = \ 1. \end{array}$$

Informally, this means that if $w \in \Sigma^{q(m)}$ is a safe advice string for $A^{\leq m}$, then any oracle query "$y \in A$?" can be decided by an AM $\cap$ coAM computation. Now, since AM $\cap$ coAM is low for Σ_2^p [29], after guessing w the computation by $M^A(x)$ can be replaced by a Σ_2^p computation. (Notice that this replacement is correct only if w is safe). Finally, at the end of this Σ_2^p computation, verifying if the guessed advice w is indeed a safe advice string can be done by a co-NP computation. ∎

As $\oplus$P, PP, PSPACE, and EXP have checkable complete problems [4, 9], for any of these classes inclusion in NPMV/poly implies its lowness for Σ_2^p by Theorem 23. In particular we get some new collapse consequence results stated in the following corollary.

Corollary 24 *If $\mathcal{C} \in \{\mathrm{PSPACE}, \mathrm{EXP}\}$ is contained in* NPMV/poly *then $\mathcal{C} = \Sigma_2^p$.*

6 Connections to NEXP

Impagliazzo, Kabanets and Wigderson in [16] have shown the surprising result that NEXP $\subseteq$ P/poly if and only if NEXP = MA. We state two results from [16] which are important ingredients in their proof of the above result.

1. NEXP $\neq$ EXP implies that for every $\epsilon > 0$, AM $\subseteq$ i.o-[NTIME$(2^{n^\epsilon})/n^\epsilon$] [16].

2. From the above result they derive that NEXP $\subseteq$ P/poly implies NEXP = EXP [16].

Here, NTIME$(2^{n^\epsilon})/n^\epsilon$ stands for the advice class, with advice length n^ϵ, corresponding to NTIME(2^{n^ϵ}). Also, AM $\subseteq$ i.o-[NTIME$(2^{n^\epsilon})/n^\epsilon$] means that for each $L \in$ AM there is a language $A \in$ NTIME$(2^{n^\epsilon})/n^\epsilon$ such that for infinitely many n we have $A \cap \Sigma^n = L \cap \Sigma^n$.

In this section we make some observations relating their results to our result on NPSV/poly (i.e., Theorem 20). Recall from Theorem 20 and Corollary 21 that EXP $\subseteq$

NPSV/poly implies EXP = AM. We can combine this with the results of [16] to derive further results. First we generalize one of the results of [16].

Lemma 25 NEXP $\subseteq$ NPSV/poly *implies* NEXP = EXP.

Proof. To see the above, notice that NEXP in NPSV/poly implies EXP is in NPSV/poly. By Theorem 20 we get the collapse EXP = AM.

Now, suppose NEXP $\neq$ EXP. By a result of [16] (stated at the beginning of this section) it follows that AM, and therefore even EXP, has an i.o-[NTIME$(2^{n^\epsilon})/n^\epsilon$] simulation for each $\epsilon > 0$. But, arguing as in [16], we observe that NEXP $\subseteq$ NPSV/poly implies that EXP is *not* contained in i.o-[NTIME$(2^n)/n$] (the proof is by a diagonalization argument as in [16]). This is a contradiction. Thus, NEXP = EXP. ■

The above results yield the following consequence.

Theorem 26 NEXP $\subseteq$ NPSV/poly *implies* NEXP = AM.

The reverse implication in the above statement is open.

Next, recall from Section 4 that EXP $\subseteq$ P/poly implies EXP = IP[P/poly]. Combining with the result of [16] that NEXP $\subseteq$ P/poly implies NEXP = EXP, we immediately get the following.

Theorem 27 NEXP $\subseteq$ P/poly *if and only if* NEXP = IP[P/poly].

Next, combining Theorem 27 with the theorem in [16] that NEXP $\subseteq$ P/poly if and only if NEXP = MA, we get a kind of gap theorem.

Corollary 28 NEXP = MA *if and only if* NEXP = IP[P/poly].

The gap is interesting because MA contains NP whereas IP[P/poly] is low for $\mathrm{ZPP}^{\mathrm{NP}}$ as shown in Theorem 11.

We next derive a result for NPSV/poly, similar to Theorem 27. Firstly, we can easily strengthen Theorem 20 as shown below. This is analogous to replacing MA by IP[P/poly] in the result of [6].

Lemma 29 EXP $\subseteq$ NPSV/poly *implies that* EXP = IP[NPSV/poly] *(where* NPSV/poly *bounds the complexity of honest provers in the considered* IP *protocols).*

The above theorem combined with Lemma 25 yields the following.

Theorem 30 NEXP $\subseteq$ NPSV/poly *if and only if* NEXP = IP[NPSV/poly].

Notice that in the above the reverse implication follows easily as IP[NPSV/poly] is contained in NPSV/poly.

Remark In [11], it is show that MA_{exp} — the exponential-time analog of AM — is not contained in P/poly. This result is proved in [11] as an application of EXP $\subseteq$ P/poly implies EXP = MA. It can be seen that the proof in [11] actually shows that $\mathrm{IP[P/poly]}_{exp}$ is not contained in P/poly, using the fact that EXP $\subseteq$ P/poly actually implies EXP = IP[P/poly]. Here, $\mathrm{IP[P/poly]}_{exp}$ is the exponential time analog of IP[P/poly], defined similarly as MA_{exp}. Notice that MA_{exp} contains NEXP, while it appears unlikely that $\mathrm{IP[P/poly]}_{exp}$ contains NEXP (it contains BPEXP, clearly).

Acknowledgments The first author was partially supported by an Alexander von Humboldt fellowship in the year 1999, and he is grateful to Prof. Uwe Schöning for hosting his visit to Ulm university where this work was carried out. Part of the work was supported by a DST-DAAD grant supporting exchange visits.

References

[1] V. Arvind and J. Köbler, Pseudorandomness and resource-bounded measure, *Proceedings 17th Conference on the Foundations of Software Technology & Theoretical Computer Science,* Springer-Verlag, LNCS **1346** (1997), 235-249.

[2] V. Arvind, J. Köbler, and R. Schuler, On helping and interactive proof systems, *International Journal of Foundations of Computer Science* **6(2)** (1994), 137-153.

[3] L. Babai, Bounded round interactive proofs in finite groups, *SIAM Journal of Discrete Mathematics,* **5** (1992), 88-111.

[4] L. Babai, L. Fortnow, C. Lund, Non-deterministic exponential time has two-prover interactive protocols, *Computational Complexity,* **1** (1991), 1-40.

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

[6] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson, BPP has subexponential simulations unless EXPTIME has publishable proofs, *Computational Complexity,* **3** (1993), 307–318.

[7] J. Balcázar, Self-reducibility, *Journal of Computer and System Sciences,* **41** (1990), 367-388.

[8] J. L. Balcázar, J. Díaz, and J. Gabarró, "Structural Complexity I", EATCS Monographs on Theoretical Computer Science. Springer-Verlag, second edition, 1995.

[9] M. Blum and S. Kannan, Designing programs that check their work, *Journal of the ACM,* **43** (1995), 269–291.

[10] R. Book, T. Long, and A. L. Selman, Quantitative relativizations of complexity classes, *SIAM Journal on Computing,* **13** (1984), 461-487.

[11] H. Buhrman, L. Fortnow, and T. Thierauf, Nonrelativizing separations, *Proceedings of the 13th IEEE Conference on Computational Complexity,* IEEE, New York, (1998), 8-12.

[12] J.Y. Cai, L.A. Hemaspaandra, and G. Wechsung, Robust Reductions, *Proceedings of the 4th Annual International Computing and Combinatorics Conference,* Springer-Verlag, LNCS **1449** (1998), 174-183.

[13] S. Fenner, L. Fortnow, A. Naik, and J. Rogers, Inverting onto functions, *Proceedings of the 11th IEEE Conference on Computational Complexity,* IEEE, New York, (1996), 213-222.

[14] O. Goldreich, S. Micali, and A. Wigderson, Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems, *Journal of the ACM,* **38** (1991), 691-729.

[15] O. Goldreich and D. Zuckerman, Another proof that BPP⊆PH (and more), *Technical Report TR97-045, Electronic Colloquium on Computational Complexity,* 1997.

[16] R. Impagliazzo, V. Kabanets, and A. Wigderson, In search of an easy witness: exponential time versus probabilistic polynomial time, *Proceedings of the 16th IEEE Conference on Computational Complexity,* IEEE, (2001), 2–12.

[17] R. M. Karp and R. J. Lipton, Some connections between nonuniform and uniform complexity classes, *Proceedings of the 12th ACM Symposium on Theory of Computing,* ACM Press, (1980), 302-309.

[18] K. Ko and U. Schöning, On circuit-size complexity and the low hierarchy in NP, *SIAM Journal on Computing,* **14** (1985), 41-51.

[19] J. Köbler, On the structure of low sets, *Proceedings of the 10th Structure in Complexity Theory Conference,* IEEE Computer Society Press, (1995), 246-261.

[20] J. Köbler and U. Schöning, High sets for NP, *in* "Advances in Algorithms, Languages, and Complexity," D. Zu and K. Ko, editors, Kluwer Acad. Publishers, (1997), 139-156.

[21] J. Köbler, U. Schöning, and J. Torán, "The Graph Isomorphism Problem: Its Structural Complexity", Birkhäuser, Boston, 1993.

[22] J. Köbler and R. Schuler, Average-case intractability vs. worst-case intractability, *Proceedings of the conference on Mathematical Foundations of Computer Sciences (MFCS),* Springer-Verlag, LNCS **1450** (1998), 493-502.

[23] J. Köbler and O. Watanabe, New collapse consequences of NP having small circuits, *SIAM Journal on Computing,* **28(1)** (1999), 311–324.

[24] J. H. Lutz and W. J. Schmidt, Circuit size relative to pseudorandom oracles, *Theoretical Computer Science,* **107** (1993), 95-120.

[25] N. Nisan and A. Wigderson, Hardness vs randomness, *Journal of Computer and System Sciences,* **49** (1994), 149-167.

[26] C. Papadimitriou, "Computational Complexity," Addison-Wesley, 1994.

[27] M. Santha, Relativized Arthur-Merlin versus Merlin-Arthur games, *Information and Computation,* **80(1)** (1989), 44-49.

[28] U. Schöning, A low and a high hierarchy within NP, *Journal of Computer and System Sciences,* **27** (1983) 14-28.

[29] U. Schöning, Probabilistic complexity classes and lowness, *Journal of Computer and System Sciences,* **39** (1989), 84-100.

[30] A. Selman, A taxanomy of complexity classes of functions, *Journal of Computer and System Sciences,* **48(2)** (1994), 357-381.