Learning Weak Reductions to Sparse Sets

Harry Buhrman* Lance Fortnow† John M. Hitchcock‡
Bruno Loff§

Abstract

We study the consequences of NP having non-uniform polynomial size circuits of various types. We continue the work of Agrawal and Arvind [1] who study the consequences of SAT being many-one reducible to functions computable by non-uniform circuits consisting of a single weighted threshold gate. (SAT $\leq_m^p$ LT_1). They claim that as a consequence P = NP follows, but unfortunately their proof was incorrect.

We take up this question and use results from computational learning theory to show that if SAT $\leq_m^p$ LT_1 then PH = P^{NP}.

We furthermore show that if SAT disjunctive truth-table (or majority truth-table) reduces to a sparse set then SAT $\leq_m^p$ LT_1 and hence a collapse of PH to P^{NP} also follows. Lastly we show several interesting consequences of SAT $\leq_{dtt}^p$ SPARSE.

1 Introduction

In this paper we study consequences of NP having non-uniform polynomial size circuits of various types. This question is intimately related to the existence of sparse hard sets for SAT under different types of reductions, and has played a central role in complexity theory starting with the work of Berman, Hartmanis, Karp, Lipton and Mahaney [13, 23, 26].

Karp and Lipton showed that if NP is Turing reducible to a sparse set then the polynomial time hierarchy collapse to its second level. This was later improved to a collapse of PH = ZPP^{NP} [24, 14], and finally PH = S_2^p [15]. Improvement of this result to a deeper collapse is a challenging open question that implies unconditional circuit lower bounds for classes in the exponential time hierarchy.

*CWI and U of Amsterdam, buhrman@cwi.nl. Supported by a Vici grant from NWO, and EU-grant QCS.

†Northwestern, fortnow@eecs.northwestern.edu. Supported in part by NSF grants CCF-0829754 and DMS-0652521.

‡University of Wyoming, jhitchco@cs.uwyo.edu. Supported in part by an NWO visiting scholar grant and by NSF grants 0652601 and 0917417. Research done while visiting CWI.

§CWI, bruno.loff@cwi.nl. Supported by FCT grant SFRH/BD/43169/2008.

Mahaney [26] showed that if SAT reduces many-one to a sparse set then in fact P = NP. This implication was subsequently improved by Ogiwara and Watanabe [29] to bounded truth-table reductions, and later work extended this result to other weak reductions [6, 7, 8, 30, 5, 9, 10]. Notoriously open is to show a similar result for disjunctive truth-table reductions. The best known consequence of this is a collapse of PH to P^{NP} [11].

Agarwal and Arvind [1] took a geometric view of this question and studied the consequences of SAT many-one reducing to LT_1, the class of languages accepted by non-uniform circuits consisting of a single weighted threshold gate. They claimed that SAT $\leq_m^p$ LT_1 implies P = NP — unfortunately, the proof in that paper was flawed, as it relied essentially on their incorrect *Splitting Lemma* (p. 203).[1]

We take a fresh look at this approach and connect it with results in learning theory. We use an efficient deterministic algorithm from Maass and Turán [25] for learning half spaces, to obtain a collapse of the polynomial-time hierarchy to P^{NP} from the assumption that SAT $\leq_m^p$ LT_1. Interestingly the main ingredient in the learning algorithm is the use of linear programming, which also featured prominently in the work of Agrawal and Arvind.

The use of learning theory in this area of complexity theory is not new and was used before by [24, 14, 18, 19], however the use of *deterministic* learning algorithms in relationship with the polynomial time hierarchy is new.

Next we examine the consequences of SAT $\leq_{dtt}^p$ SPARSE and make a link with the geometric approach above. Using the leftset technique from [29] it is easy to show for conjunctive truth-table reductions that if SAT $\leq_{ctt}^p$ SPARSE then P = NP. Frustratingly, for disjunctive truth table reductions the best known consequence is PH = P^{NP}, a result due to Arvind et al.[11], who use a complicated argument. We use error-correcting codes to show that $SAT \leq_{dtt}^p$ SPARSE implies that SAT $\leq_m^p$ LT_1, which with our previous result gives a new and more modular proof of the collapse to P^{NP}. Our new approach enables us to obtain the same collapse for majority reductions.

We finish with a handful of new consequences of SAT $\leq_{dtt}^p$ SPARSE and SAT $\leq_{maj}^p$ SPARSE. Interestingly it turns out that in the case of disjunctive reductions, improvement of the above results to a collapse of PH = $P_{\|}^{NP}$ is sufficient to obtain the full collapse to P = NP.

[1] The mistake in this *Splitting Lemma* was not seen by any of the paper's referees, but instead was accidentally discovered years later. For an anecdotal account of the episode, please consult `http://blog.computationalcomplexity.org/2009/10/thanks-for-fuzzy-memories.html`.

2 Preliminaries

We assume that the reader is familiar with computational complexity, as expounded, for instance, in [4]. In particular, we make use of

$$A \in \mathrm{P}/poly \iff A \leq_T^p \mathrm{SPARSE},$$

so a reduction to a sparse set can be seen as a polynomial-time circuit. The weaker the reduction, the weaker the access to non-uniformity.

The least common notation we use, is: $\mathrm{P}_{\|}^{\mathrm{NP}}$ and $\mathrm{FP}_{\|}^{\mathrm{NP}}$, which are the classes of sets and functions, respectively, that are polynomial-time computable with non-adaptive queries to an NP oracle; $\mathrm{P}^{\mathrm{NP}[q]}$, and $\mathrm{FP}^{\mathrm{NP}[q]}$, the classes of sets and functions that are polynomial-time computable by asking no more than $q(n)$ (possibly adaptive) queries to an NP oracle.

A linear threshold function $L : \{0,1\}^m \to \{0,1\}$ is defined by a vector of m real numbers $w \in \mathbb{R}^m$, called weights, a threshold $\theta \in \mathbb{R}$, and the equation

$$L(z) = \begin{cases} 1 & \text{if } z \cdot w > \theta, \text{ and} \\ 0 & \text{if } z \cdot w \leq \theta. \end{cases}$$

Here $z \cdot w$ denotes the inner product $\sum_{i=1}^m z_i w_i$.

We let $\mathrm{LT}_1(m)$ denote the class of linear threshold functions with m-bit binary inputs. We may freely assume, for functions in $\mathrm{LT}_1(m)$, that the weights and threshold are integers of bit-length $m \log m$ [27, Thm. 16].

In this paper we are concerned with three kinds of reductions:

Definition 1. (*dtt* **reductions)** *A set A disjunctive truth-table reduces to a set S, written $A \leq_{dtt}^p S$, if there exists a polytime computable function Q, outputting a set of queries, such that*

$$x \in A \iff Q(x) \cap S \neq \varnothing.$$

(majority reductions) *A set A majority truth-table reduces to a set S, written $A \leq_{maj}^p S$, if there exists a function Q, as above, such that*

$$x \in A \iff |Q(x) \cap S| > \frac{|Q(x)|}{2}$$

(LT_1 reductions) *A set A reduces to linear-threshold functions, written $A \leq_m^p \mathrm{LT}_1$, if there exists a polytime computable function f, and a family $\{L_n\}_{n\in\mathbb{N}}$ of linear threshold functions, such that*[2]

$$x \in A^{=n} \iff L_n(f(x)) = 1.$$

[2]Notice that the length of $f(x)$ must be a function of the length of x.

3 If $\text{SAT} \leq_m^p \text{LT}_1$ ...

Attempting to derive P = NP should prove difficult, since by the next section this would imply the same collapse for *dtt* and majority reductions to sparse sets. Since $A \leq_m^p \text{LT}_1$ implies $A \in \text{P}/poly$, then [15] gives us $\text{PH} = \text{S}_2^p$. This collapse can be improved in the following way:

Theorem 1. *If* $\text{SAT} \leq_m^p \text{LT}_1$*, then* $\text{PH} = \text{P}^{\text{NP}}$.

We take a similar approach as [14]: the existence of a suitable learning algorithm will, under the assumption that $\text{SAT} \leq_m^p \text{LT}_1$, collapse the polynomial-time hierarchy. The difference being that we have a *deterministic* learning algorithm for linear threshold functions, but only (zero-error) probabilistic algorithms with access to an NP oracle are known that can learn general circuits.

Our learning model is the on-line learning model of Angluin [3] for learning with counter-examples. In our case, the learner wishes to identify an unknown linear threshold function, say $L \in \text{LT}_1(m)$. At each learning step, the algorithm proposes some hypothesis $H \in \text{LT}_1(m)$. If $H \neq L$, then the algorithm is given a counter-example x such that $H(x) \neq L(x)$. The algorithm is not allowed to make any assumptions on the counter-example, which could very well be adversarially chosen. Based on the previous counter-examples and hypotheses, the algorithm suggests a new hypothesis which is correct on the inputs seen so far, and the process is repeated until $H = L$. The learning complexity of such an algorithm is the maximum number of these steps that it will need in order to learn any function in $\text{LT}_1(m)$.

Theorem 2 ([25])**.** *There is a deterministic polynomial-time algorithm for learning* $\text{LT}_1(m)$ *functions in* $O(m^3 \log m)$ *steps.*

As a corollary, we can now prove Theorem 1.

Proof of Theorem 1. Suppose $\text{SAT} \leq_m^p \text{LT}_1$, and let L_n be a family of linear threshold functions, and f a polytime reduction, such that

$$\psi \in \text{SAT}^{=n} \iff L_n(f(\psi)) = 1. \tag{1}$$

For a given formula of length n, we use the algorithm of Theorem 2 in order to uncover a linear threshold function H with the same property (1) as L_n, in polynomial time with the help of an NP oracle.

Let $m = |f(\psi)|$ on inputs ψ of length n. We proceed as follows: we start with an initial hypothesis H for L_n, given by the learning algorithm for $\text{LT}_1(m)$. Then at each step in the learning process we ask the NP oracle, if there exists some formula ψ of length n such that:

1. ψ has no free variables and evaluates to true, but $H(f(\psi)) = 0$, or

2. ψ has no free variables and evaluates to false, but $H(f(\psi)) = 1$, or

3. $H(f(\psi)) = 1$ but both $H(f(\psi_0)) = 0$ and $H(f(\psi_1)) = 0$, or

4. $H(f(\psi)) = 0$, but $H(f(\psi_0)) = 1$ or $H(f(\psi_1)) = 1$.

Above, ψ_0 and ψ_1 are obtained by replacing the first variable of ψ respectively with 0 or 1. Essentially, we are asking whether the set $\textsc{Sat}(H) = \{\psi | H(f(\psi)) = 1\}$ violates the self-reducibility of Sat. If this is not the case, then necessarily $\textsc{Sat}(H) = \textsc{Sat}^{=n}$, and we are done.

But if the self-reducibility is violated, then for at least one $\phi \in \{\psi, \psi_0, \psi_1\}$, we must have $H(f(\phi)) \neq L_n(f(\phi))$, and so $f(\phi)$ gives us a counter-example to update the hypothesis H. We use prefix-search to obtain such a formula ψ. Then by equation (1) we can use the Sat oracle again in order to know which $\phi \in \{\psi, \psi_0, \psi_1\}$ will provide the counter-example $H(f(\phi)) \neq L_n(f(\phi))$.

After $O(m^3 \log m) = \mathit{poly}(n)$ many iterations, we will either have learnt L_n, or otherwise obtained an hypothesis H suitable for the purpose of querying $\textsc{Sat}^{=n}$.

By feeding the NP oracle the suitable linear-threshold functions, it now becomes possible to simulate a Σ_2^p computation. So Σ_2^p, and consequently all of PH, collapses to P^{NP}. □

The algorithm above is non-adaptive, and in order to solve $\textsc{Sat}^{=n}$, it potentially asks $\Omega(nm^3 \log m)$-many queries to Sat. We can be a bit more clever, and actually reduce this number to n. This will essentially give us the following:

Theorem 3. *If* $\textsc{Sat} \leq_m^p \mathrm{LT}_1$, *then* $\mathrm{NP}^{\textsc{Sat}^{=n}} \subseteq \mathrm{P}^{\textsc{Sat}[n]} \cap \mathrm{NP}/\mathit{lin}$.

Proof. The idea is to use the self-reducibility of Sat once again, in order to learn $\textsc{Sat}^{=n}$ first for formulas with no free variables, that evaluate to true and ones that evaluate to false, then for formulas with 1 free variable, then 2 free variables, and so on. Let $\textsc{Sat}_k^{=n}$ be the set of satisfiable formulas having exactly k free variables. Starting with the initial hypothesis H, we set out to learn $\textsc{Sat}_0^{=n}$. What is the largest number of mistakes that we can make, i.e., how many times might we need to change our hypothesis H until we have properly learned $\textsc{Sat}_0^{=n}$?

Using a Sat oracle, we can ask: *is there a sequence* $\psi_1, \ldots, \psi_\ell$ *of* ℓ *formulas, having* 0 *vars, such that* ψ_{i+1} *is always a counter-example to the hypothesis constructed by our learning algorithm after seeing* $\psi_1, \ldots, \psi_i$*?*[3]

[3]Formalizing the question as an NP-set gives us:

$$A = \{\langle 0^n, 0^\ell \rangle \mid \exists \vec{\psi}, \vec{H} \forall i \; H_i = \mathrm{Learner}(\psi_1, \ldots, \psi_i) \wedge H_{i-1}(f(\psi_i)) \neq \textsc{Sat}(\psi_i)\},$$

where $\vec{\psi}$ is a sequence of ℓ-many formulas with 0 vars, $\vec{H}$ is a sequence of ℓ-many threshold functions, and $i \in \{1, \ldots, \ell\}$. Notice that $H_{i-1}(f(\psi_i)) \neq \textsc{Sat}(\psi_i)$ is decidable in polynomial time because the formulas ψ_i have no variables.

We know that such a sequence will have at most $poly(n)$ formulas, and so using binary search, then by making $O(\log n)$ such queries, we can find the length of the largest sequence of counter-examples which can be given to our learning algorithm before it necessarily learns $\text{SAT}_0^{=n}$. Let this length be ℓ_0.

Then because ℓ_0 is maximal, at this point we know that if the learning algorithm is given *any* sequence of ℓ_0-many counter-examples having no variables, the constructed hypothesis H will be correct on $\text{SAT}_0^{=n}$, in the sense that $\psi \in \text{SAT}_0^{=n} \iff H(f(\psi)) = 1$.

Now that we know ℓ_0, we set out to learn $\text{SAT}_1^{=n}$. Using SAT as an oracle, we may ask: *Is there a sequence of ℓ_0 counter-examples with 0 vars, followed by ℓ counter-examples with 1 var?* Thus we may obtain ℓ_1, the length of the largest sequence of counter-examples with 1 var, that can be given to the learning algorithm *after it has already learned every possible formula with 0 vars.*

In general we know $\ell_0, \ldots, \ell_{k-1}$, and we set out to learn $\text{SAT}_k^{=n}$. Using SAT as an oracle, we ask: *Is there a sequence of ℓ_0 counter-examples with 0-vars, followed by ℓ_1 counter-examples with 1-var, ..., followed by ℓ_{k-1} counter-examples with $k-1$ vars, followed by ℓ counter-examples with k vars?*

The key observation is that in order for the SAT oracle to be able to tell whether a formula ψ with k variables is a counter-example to hypothesis H, i.e., whether $H(f(\psi)) \neq \text{SAT}(\psi)$, it will need to know whether ψ is or is not satisfiable. In order to know this, the SAT oracle uses H itself, which at this point is known to be correct for formulas with $k-1$ variables, and thus $\psi \in \text{SAT} \iff H(f(\psi_0)) = 1$ or $H(f(\psi_1)) = 1$.

In the end we have $n+1$ numbers $\ell_0, \ldots, \ell_n$, and we know that if the learning algorithm is given *any* sequence of ℓ_0-many counter-examples having no variables, followed by ℓ_1 counter-examples having 1 variable, ..., followed by ℓ_n counter-examples having n variables, then the constructed hypothesis H will be correct on all of $\text{SAT}^{=n}$. Furthermore, such a sequence must exist by construction.

These numbers take up at most $O(n \log n)$ many bits, and each bit is the outcome of one (much larger, adaptive) query to SAT. Having access to $\ell_0, \ldots, \ell_n$, an NP machine can guess a proper sequence of counter-examples, and it will thus obtain an hypothesis H which it can use to answer any query to $\text{SAT}^{=n}$. Thus $\text{NP}^{\text{SAT}^{=n}} \subseteq \text{P}^{\text{SAT}[n \log n]}$, and $\text{NP}^{\text{SAT}^{=n}} \subseteq \text{NP}/n \log n$.

In order to improve $n \log n$ into n bits, or even $\frac{n}{c \log n}$ bits, the proof is similar, but instead of learning how to decide $\text{SAT}^{=n}$ for one extra variable at a time, we learn $O(\log n)$ many extra variables at a time — this requires us to unfold the self-reduction tree $O(\log n)$-deep. □

Under the assumption that SAT has polynomial-size circuits, we may decide, in coNP, wether a given string $\alpha(n)$ encodes a circuit correct for

$\text{SAT}^{=n}$. However, there will possibly be many strings with this property — the following theorem gives us a way to single out, in coNP, a *unique* advice string $\alpha(n)$ suitable to decide $\text{SAT}^{=n}$. The proof will be put in appendix.

Theorem 4. *If* $\text{NP} \subseteq \text{P/}poly$*, and* $\text{PH} \subseteq \text{P}^{\text{NP}}$*, then* $\text{PH} \subseteq \text{P/}\alpha$ *for some polynomial advice function* $0^n \mapsto \alpha(n)$ *whose graph* $G_\alpha = \{\langle 0^n, \alpha(n)\rangle | n \in \mathbb{N}\} \in \text{coNP}$.

Corollary 5. *If* $\text{SAT} \leq_m^p \text{LT}_1$*, then* $\text{PH} \subseteq \text{P/}\alpha$ *for some polynomial advice function* $0^n \mapsto \alpha(n)$ *whose graph* $G_\alpha \in \text{coNP}$.

4 LT_1 versus *dtt* and *maj* reductions

In this section we show that LT_1 reductions can simulate *dtt* and majority reductions to sparse sets. Thus, effectively, the collapses we have proven for LT_1 reductions imply similar collapses for *dtt* and majority reductions.

Theorem 6. *If* $A \leq_{dtt}^p \text{SPARSE}$ *or* $A \leq_{maj}^p \text{SPARSE}$*, then* $A \leq_m^p \text{LT}_1$.

Proof. We will use a Reed-Solomon code to construct the LT_1 reduction. Suppose $A \leq_{dtt}^p S \in \text{SPARSE}$, and assume w.l.o.g. that the *dtt* reduction is given by a polytime computable function Q, such that

$$x \in A^{=n} \iff S^{=m} \cap Q(x) \neq \varnothing, \tag{2}$$

$$|S^{=m}| = m, \text{ and}$$

$$|Q(x)| = d.$$

That is, for every input x of length n, $Q(x) = \{y_1, \ldots, y_d\}$ always queries the same number of $d = d(n)$ strings of the same length $m = m(n)$, and that there will be exactly m many such strings in $S^{=m}$.

We will be working over the field $\mathbb{F}_{2^\ell}$, for $\ell \geq \lceil \log dm^2 \rceil$. For any given binary string s of length m, we define the polynomial $p_s(z) = \sum_{i=1}^m s_i z^{i-1}$. Now let $C(s)$ be the encoding of s as a $2^\ell \times 2^\ell$-long binary string: this string is the concatenation of $p_s(a)$, as a goes through all the 2^ℓ elements of $\mathbb{F}_{2^\ell}$; each $p_s(a)$ is in turn encoded by a binary string of length 2^ℓ, having a 1 at position $p_s(a)$ (for some fixed enumeration of $\mathbb{F}_{2^\ell}$), and 0s elsewhere.

Note that $|C(s)| = O(d^2m^4) = poly(n)$. Then vitally note that by encoding strings this way, the number of bit positions where $C(s)$ and $C(y)$ are equal, given by the inner product $C(s) \cdot C(y)$,[4] is exactly the number of elements $a \in \mathbb{F}_{2^\ell}$ where $p_s(a) = p_y(a)$. So for any two words $s, y \in \{0,1\}^m$,

[4]Note that the binary strings $C(s)$ and $C(y)$ are seen as 0-1 vectors, and that the inner product is a natural number $\sum_{j=1}^{2^\ell \times 2^\ell} C(s)_j C(y)_j$.

using the fact that $p_s - p_y$ is either identically zero, or has at most $m-1$ roots,

$$\begin{cases} C(y) \cdot C(s) \leq m-1 & \text{if } y \neq s, \text{ and} \\ C(y) \cdot C(s) \geq dm^2 & \text{if } y = s. \end{cases}$$

Define $g(x) = \bigvee_{i=1}^{d} C(y_i)$, where $Q(x) = \{y_1, \ldots, y_d\}$, and by $\bigvee$ we mean bitwise-OR. Then

$$\begin{cases} g(x) \cdot C(s) \leq \sum_{i=1}^{d} C(y_i) \cdot C(s) \leq d(m-1) & \text{if } s \notin Q(x), \text{ and} \\ g(x) \cdot C(s) \geq dm^2 & \text{if } s \in Q(x). \end{cases}$$

Finally, let $w_n = \oplus_{s \in S^{=m}} C(s)$, and $f(x) = (g(x))^{\oplus m}$, where by $\oplus$ we mean the direct sum of vectors / concatenation of strings. Then $f(x) \cdot w_n = \sum_{s \in S^{=m}} g(x) \cdot C(s)$, and we come to

$$\begin{cases} f(x) \cdot w_n \leq md(m-1) & \text{if } S^{=m} \cap Q(x) = \varnothing, \text{ and} \\ f(x) \cdot w_n \geq dm^2 & \text{if } S^{=m} \cap Q(x) \neq \varnothing. \end{cases} \tag{3}$$

So $x \in A \iff f(x) \cdot w_n > dm(m-1)$, showing that $A \leq_m^p \mathrm{LT}_1$.

The transformation for *maj* reductions is similar. We begin with a *dtt* reduction function Q, which is like before, except that now Equation (2) is replaced with

$$x \in A^{=n} \iff |S^{=m} \cap Q(x)| > \frac{d}{2}.$$

Then both the LT_1 reduction function f, and the set of weights w_n are constructed exactly in the same way, but over a slightly larger field. Working through the proof, if 2^ℓ is the size of our chosen field, and $K = |S^{=m} \cap Q(x)|$, then Equation (3) becomes:

$$2^\ell K \leq f(x) \cdot w_n \leq 2^\ell K + d(m-1)(m-K).$$

Now choose $\ell \geq \lceil \log 4dm^2 \rceil$ as the size of our field. Using the defining property of the *maj* reduction, a small computation will show us that

$$x \in A^{=n} \iff K > \frac{d}{2} \iff f(x) \cdot w_n > 2^\ell \left(\frac{d}{2} + \frac{1}{4} \right)$$

— this defines our LT_1 reduction. □

5 If SAT $\leq_{dtt}^p$ SPARSE ...

Disjunctive truth-table reductions to sparse sets are powerful enough to simulate bounded truth-table reductions to sparse sets [2]. But the collapses that are known, under the assumption that SAT $\leq_{dtt}^p$ SPARSE, are not as strong as those for btt reductions. We can summarize what was known about SAT $\leq_{dtt}^p$ SPARSE, in the following two theorems:

Consequense 1 ([16, 12])**.** *... then* $\mathrm{FP}_{\|}^{\mathrm{NP}} = \mathrm{FP}^{\mathrm{NP[log]}}$, $\mathrm{UP} \subseteq \mathrm{P}$, *and* $\mathrm{NP} = \mathrm{RP}$. □

Consequense 2 ([11])**.** *... then* $\mathrm{PH} = \mathrm{P}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{RP}} = \mathrm{BPP}$. □

To these consequences, we append our own observations, which follow from the results in the previous sections.

Consequense 3. *... then* $\mathrm{NP}^{\mathrm{SAT}^{=n}} \subseteq \mathrm{P}^{\mathrm{SAT}[n]}$, $\mathrm{NP}^{\mathrm{SAT}^{=n}} \subseteq \mathrm{NP}/\mathit{lin}$. □

Consequense 4. *... then* $\mathrm{PH} \subseteq \mathrm{P}/\alpha$ *for some function* $0^n \mapsto \alpha(n)$ *whose graph* $G_\alpha \in \mathrm{coNP}$. □

Finally, we note that we are not far away from obtaining the final consequence $\mathrm{P} = \mathrm{NP}$.

Consequense 5. *... then* $\mathrm{E} \not\subseteq \mathrm{NP}/\log$.

Consequense 6. *... then* $\mathrm{E}^{\mathrm{NP}} \not\subseteq \mathrm{SIZE}(2^{\varepsilon n})$ *for some* $\varepsilon > 0$.

Consequense 7. *... then the following statements are all equivalent:*

1. $\mathrm{P} = \mathrm{NP}$.
2. $\mathrm{P}^{\mathrm{NP}} = \mathrm{P}_{\|}^{\mathrm{NP}}$.
3. $\mathrm{coNP} \cap \mathrm{SPARSE} \subseteq \mathrm{NP}$.
4. $\mathrm{E}^{\mathrm{NP}} = \mathrm{E}_{\|}^{\mathrm{NP}}$.

Proof of Consequence 5. [17] show that

$$\mathrm{EXP} \subseteq \mathrm{P}_{\|}^{\mathrm{NP}} \iff \mathrm{EXP} \subseteq \mathrm{NP}/\log.$$

But if we had $\mathrm{EXP} \subseteq \mathrm{P}_{\|}^{\mathrm{NP}}$, then we could compute the lexicographically least satisfying assignment of a given formula in $\mathrm{FP}_{\|}^{\mathrm{NP}}$, and thus in $\mathrm{FP}^{\mathrm{NP[log]}}$, by Consequence 1. But then we could also do it in FP alone, simply by trying every possible answer to the queries made by the $\mathrm{FP}^{\mathrm{NP[log]}}$ computation. But then $\mathrm{P} = \mathrm{NP}$, and the necessary conclusion $\mathrm{EXP} \subseteq \mathrm{PH} \subseteq \mathrm{P}$ would contradict the time-hierarchy theorem. □

Proof of Consequence 6. By counting there is a function $f : \{0,1\}^{\log n} \to \{0,1\} \notin \mathrm{SIZE}(n^\varepsilon)$ which can be found in P^{Σ_2} [cf. 22], and thus, by Consequence 2, in P^{NP}. Translating this upwards we get a set in E^{NP} with no circuits of size $2^{\varepsilon n}$. □

Proof of Consequence 7. As in the proof of Consequence 5, $\mathrm{P} = \mathrm{NP}$ follows if we are able to compute the least satisfying assignment of a given formula in $\mathrm{FP}_{\|}^{\mathrm{NP}}$. This is trivially the case when $\mathrm{P}^{\mathrm{NP}} = \mathrm{P}_{\|}^{\mathrm{NP}}$.

Now if $\mathrm{SPARSE} \cap \mathrm{coNP} \subseteq \mathrm{NP}$, then, from Consequence 4, we get $\mathrm{PH} \subseteq \mathrm{NP}^{G_\alpha} \subseteq \mathrm{NP}^{\mathrm{NP} \cap \mathrm{SPARSE}}$: the non-deterministic machine just guesses the advice α and checks it using the oracle. But $\mathrm{NP}^{\mathrm{NP} \cap \mathrm{SPARSE}} \subseteq \mathrm{NP}$ [cf 21], and thus the least satisfying assignment of a given formula can be obtained in $\mathrm{FP}_{\|}^{\mathrm{NP}}$.

To see the third equivalence, notice that $\mathrm{E}^{\mathrm{NP}} = \mathrm{E}_{\|}^{\mathrm{NP}}$, then Consequence 6 implies we can derandomise BPP in $\mathrm{P}_{\|}^{\mathrm{NP}}$ [cf. 28, 20]; since $\mathrm{PH} \subseteq \mathrm{BPP}$, this implies that the least satisfying assignment can be found in $\mathrm{FP}_{\|}^{\mathrm{NP}}$. □

6 Final remarks

We remark that our paper also draws new conclusions from $\textsc{Sat} \leq_{maj}^{p}$ SPARSE. It was previously known that, under this hypothesis, $\mathrm{NP} = \mathrm{RP}$, but it remains open to show that $\mathrm{FP}_{\|}^{\mathrm{NP}} = \mathrm{FP}^{\mathrm{NP[log]}}$ [cf. 12]. However, the results in this paper imply that Consequences 2, 3 and 4 of the previous section also apply to the $\textsc{Sat} \leq_{maj}^{p}$ SPARSE case, which was previously unknown.

References

[1] M. Agrawal and V. Arvind. Geometric sets of low information content. *Theor. Comput. Sci.*, 158(1-2):193–219, 1996.

[2] E. Allender, L. A. Hemachandra, M. Ogiwara, and O. Watanabe. Relating equivalence and reducibility to sparse sets. *SIAM J. Comput.*, 21(3):521–539, 1992.

[3] D. Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, 1987.

[4] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[5] V. Arvind, Y. Han, L. Hemachandra, J. Köbler, A. Lozano, M. Mundhenk, M. Ogiwara, U. Schöning, R. Silvestri, and T. Thierauf. Reductions to sets of low information content. In K. Ambos-Spies, S. Homer, and U. Schöning, editors, *Complexity Theory: Current Research*, pages 1–45. Cambridge University Press, 1993.

[6] V. Arvind, J. Köbler, and M. Mundhenk. Bounded truth-table and conjunctive reductions to sparse and tally sets. Technical report, University of Ulm, 1992.

[7] V. Arvind, J. Köbler, and M. Mundhenk. Lowness and the complexity of sparse and tally descriptions. In *Proc. 3rd ISAAC*, pages 249–258. Springer, 1992.

[8] V. Arvind, J. Köbler, and M. Mundhenk. On bounded truth-table, conjunctive, and randomized reductions to sparse sets. In *Proc. 12th CFSTTCS*, pages 140–151. Springer-Verlag, 1992.

[9] V. Arvind, J. Köbler, and M. Mundhenk. Hausdorff reductions to sparse sets and to sets of high information content. In *MFCS '93*, pages 232–241, 1993.

[10] V. Arvind, J. Köbler, and M. Mundhenk. Monotonous and randomized reductions to sparse sets. *Theo. Inform. and Appl.*, 30(2):155–179, 1996.

[11] V. Arvind, J. Köbler, and M. Mundhenk. Upper bounds for the complexity of sparse and tally descriptions. *Theor. Comput. Syst.*, 29:63–94, 1996.

[12] V. Arvind and J. Torán. Sparse sets, approximable sets, and parallel queries to NP. In C. Meinel and S. Tison, editors, *STACS 99*, volume 1563 of *Lect. Notes Comp. Sc.*, pages 281–290. Springer, 1999.

[13] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. In *Proc. 8th STOC*, pages 30–40, 1976.

[14] N. H. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Syst. Sci.*, 52(3):421–433, 1996.

[15] J.-Y. Cai. $S_2^p \subseteq \mathrm{ZPP}^{\mathrm{NP}}$. *J. Comput. Syst. Sci.*, 73(1):25–35, 2002.

[16] J.-Y. Cai, A. Naik, and D. Sivakumar. On the existence of hard sparse sets under weak reductions. In C. Puech and R. Reischuk, editors, *STACS 96*, volume 1046 of *Lect. Notes Comp. Sc.*, pages 307–318. Springer, 1996.

[17] L. Fortnow and A. Klivans. NP with small advice. In *Proc. 20th CCC*, pages 228–234, 2005.

[18] R. Harkins and J. M. Hitchcock. Dimension, halfspaces, and the density of hard sets. *Theor. Comput. Syst.*, 49(3):601–614, 2011.

[19] J. M. Hitchcock. Online learning and resource-bounded dimension: Winnow yields new lower bounds for hard sets. *SIAM J. Comput.*, 36(6):1696–1708, 2007.

[20] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits. In *Proc. 29th STOCS*, pages 220–229, 1997.

[21] J. Kadin. $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ and sparse Turing-complete sets for NP. *J. Comput. Syst. Sci.*, 39(3):282–298, 1989.

[22] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Inform. Comput.*, 55(1-3):40–56, 1982.

[23] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. 12th STOC*, pages 302–309, 1980.

[24] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. In Z. Fülöp and F. Gécseg, editors, *Automata, Languages and Programming*, volume 944 of *Lect. Notes Comp. Sc.*, pages 196–207. Springer, 1995.

[25] W. Maass and G. Turán. How fast can a threshold gate learn? In *Worksh. Comput. Learn. Theor. & Natur. Learn. Syst.*, volume 1, pages 381–414, Cambridge, MA, USA, 1994. MIT Press.

[26] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *J. Comput. Syst. Sci.*, 25(2):130–143, 1982.

[27] S. Muroga, I. Toda, and S. Takasu. Theory of majority decision elements. *J. Franklin. I.*, 271(5):376–418, 1961.

[28] N. Nisan and A. Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[29] M. Ogiwara and O. Watanabe. Polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM J. Comput.*, 20(3):471–483, 1991.

[30] D. Ranjan and P. Rohatgi. On randomized reductions to sparse sets. In *Proc. 7th STOC*, pages 239–242, 1992.

7 Appendix

Proof of Theorem 4. Let A be Δ_2-complete. Then there is a polytime machine $\mathcal{M}$ that decides $A^{=n}$ with polynomially-long advice $\gamma(n)$, where $\gamma(n)$ codes a circuit solving $\text{SAT}^{=m}$, for some $m = poly(n)$. The machine $\mathcal{M}$ uses $\gamma(n)$ to answer the queries needed in the Δ_2 computation of A. Furthermore, the function $0^n \mapsto \tilde{\alpha}(n)$, given by

$$\begin{aligned} \tilde{\alpha}(n) \quad & \text{is the lexicographically smallest string} \\ & \text{such that } x \in A^{=n} \iff \mathcal{M}(x)/\tilde{\alpha}(n) = 1, \end{aligned}$$

is in PH and thus in FP^{SAT}. Then let $\mathcal{N}$ be a polytime machine computing $\tilde{\alpha}$ with a SAT oracle, and let's say it makes k queries to compute $\tilde{\alpha}(n)$. Let $S \in \text{coNP}$ be the set of strings $\langle 0^n, \tilde{\alpha}, a_1, \ldots, a_k, y_1, \ldots, y_k \rangle$ such that

1. $\mathcal{N}^{\vec{a}}(0^n) = \tilde{\alpha}$ (i.e., when $a_1, \ldots, a_k$ are given as answers to the queries of $\mathcal{N}$),
2. if $a_i = 1$ then y_i is the lexicographically smallest satisfying assignment of the i-th formula queried by $\mathcal{N}^{\vec{a}}$, and
3. if $a_i = 0$ then $y_i = \lambda$ (the empty string) and the i-th formula queried by $\mathcal{N}^{\vec{a}}$ is not satisfiable.

Notice that for a given n, the string $\langle 0^n, \tilde{\alpha}, a_1, \ldots, a_k, y_1, \ldots, y_k \rangle \in S$ is uniquely defined, so S is the graph of $\alpha(n) = \langle \tilde{\alpha}, a_1, \ldots, a_k, y_1, \ldots, y_k \rangle$. When given $\alpha(n)$, an algorithm for A can simply check if $\mathcal{M}(x)/\tilde{\alpha} = 1$. □