A Moment of Perfect Clarity II: Consequences of Sparse Sets Hard for NP with Respect to Weak Reductions*

Christian Glaßer†
Institut für Informatik
Julius-Maximilians-Universität Würzburg
97074 Würzburg Germany

Lane A. Hemaspaandra‡
Department of Computer Science
University of Rochester
Rochester, NY 14627 USA

November 16, 2000

Abstract

This paper discusses advances, due to the work of Cai, Naik, and Sivakumar [CNS95] and Glaßer [Gla00], in the complexity class collapses that follow if NP has sparse hard sets under reductions weaker than (full) truth-table reductions.

1 Quick Hits

Most of this article will be devoted to presenting the work of Glaßer [Gla00]. However, even before presenting the background and definitions for that, let us briefly note some improvements that follow from the work of Cai, Naik, and Sivakumar due to

 Supported in part by grants NSF-CCR-9322513 and NSF-INT-9815095/DAAD-315-PPP-gü-ab, and the Studienstiftung des Deutschen Volkes. Written in part while Lane A. Hemaspaandra was visiting Julius-Maximilians-Universität Würzburg.

†E-mail: glasser@informatik.uni-wuerzburg.de.

‡E-mail: lane@cs.rochester.edu.

the results discussed in the first part of this article [GH00]. (See [GH00] for definitions of the terms and classes used here: USAT_Q, FewP, Few, etc.)

Theorem 1.1 *(follows from the techniques of [CNS95], as noted by [vM97,BFT97, Siv00]) If* SAT *disjunctively reduces to a sparse set, then* $(\exists Q)[\mathrm{USAT}_Q \in \mathrm{P}]$.

A proof of Theorem 1.1 is sketched in Section 5 below. This advance of Cai, Naik, and Sivakumar establishes immediately the following corollary in the light of two results discussed in the first part of this article ([GH00], see there for a discussion of attribution of the first of these results), namely,

1. If $(\exists Q)[\mathrm{USAT}_Q \in \mathrm{P}]$ then $\mathrm{P} = \mathrm{Few}$ (and thus $\mathrm{P} = \mathrm{UP}$ and $\mathrm{P} = \mathrm{FewP}$).
2. [VV86] If $(\exists Q)[\mathrm{USAT}_Q \in \mathrm{P}]$ then $\mathrm{R} = \mathrm{NP}$.

Corollary 1.2 *If* SAT *disjunctively reduces to a sparse set, then* $\mathrm{P} = \mathrm{Few}$ *and* $\mathrm{R} = \mathrm{NP}$.

Furthermore, Arvind, Köbler, and Mundhenk [AKM96] prove that if SAT disjunctively reduces to a sparse set, then $\mathrm{PH} = \mathrm{P}^{\mathrm{NP}}$. However, in light of Corollary 1.2, clearly the following can be claimed.

Theorem 1.3 *If* SAT *disjunctively reduces to a sparse set, then* $\mathrm{PH} = \mathrm{P}^{\mathrm{R}}$.

2 Background and Motivation

The study of the consequences of NP having sparse hard sets under various types of (polynomial-time) reductions makes one of the most interesting tales in complexity theory. However, we will not repeat that tale here, as many good surveys of (parts of) that story are available [Mah86,You92,HOW92,CO97]. Instead, let us cut right to the chase.

In particular, Table 1 shows, for the most widely studied reductions, the strongest currently known consequences of NP having sparse hard sets with respect to that reduction (we use the definitions of [LLS75], and we will, below, define some additional reductions).

Table 1 brings immediately to mind the key issue: For those reduction types for which a $\mathrm{P} = \mathrm{NP}$ conclusion is not yet known, can one achieve such a conclusion or, failing that, what is the strongest conclusion that one can achieve?

Reduction	Consequence of the existence of sparse sets hard for NP	Reference
$\leq^{\mathrm{p}}_{\mathrm{m}}$	P = NP	[Mah82]
$\leq^{\mathrm{p}}_{\mathrm{btt}}$	P = NP	[OW91]
$\leq^{\mathrm{p}}_{\mathrm{c}}$	P = NP	[AHH$^+$93]
$\leq^{\mathrm{p}}_{\mathrm{btt(c)}}$	P = NP	[AHH$^+$93]
$\leq^{\mathrm{p}}_{\mathrm{d}}$	P^{R} = PH, P = Few, and R = NP	See Section 1
$\leq^{\mathrm{p}}_{\mathrm{tt}}$	$\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}}$	[KW99]
$\leq^{\mathrm{p}}_{\mathrm{T}}$	$\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}}$	[KW99]
$\leq^{\mathrm{RS}}_{\mathrm{T}}$	$\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}}$	[KW99], see [CHW99]
$\leq^{\mathrm{O}}_{\mathrm{T}}$	$\mathrm{PH} = \mathrm{NP}^{\mathrm{NP}}$	[CHW99]
$\leq^{\mathrm{SN}}_{\mathrm{T}}$	$\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}^{\mathrm{NP}}}$	Implicit in [KW99], see [CHW99]

Table 1: Consequences of the existence of sparse sets hard for NP. $\leq^{\mathrm{SN}}_{\mathrm{T}}$, $\leq^{\mathrm{O}}_{\mathrm{T}}$, and $\leq^{\mathrm{RS}}_{\mathrm{T}}$ are respectively strong nondeterministic reductions; strong and robustly over-productive reductions; and robustly strong reductions (see [CHW99] for definitions and discussion).

Proving a P = NP (or even a collapse of the boolean hierarchy) result for $\leq^{\mathrm{p}}_{\mathrm{tt}}$ or $\leq^{\mathrm{p}}_{\mathrm{T}}$ reductions may be difficult, or at least it will require nonrelativizable techniques, due to the following results.

Theorem 2.1 *[AHH$^+$93] There is an oracle world in which* NP *has a* $\leq^{\mathrm{p}}_{\mathrm{tt}}$*-complete tally set yet the boolean hierarchy does not collapse.*

Theorem 2.2 *[Kad89] For any* $f(n) = \omega(\log n)$ *there is an oracle world in which* NP *has sparse Turing-complete sets yet* $\mathrm{PH} \neq \mathrm{P}^{\mathrm{NP}[f(n)]}$.

Regarding Theorem 2.1, one should keep in mind that it is well-known that the following are all equivalent:

1. NP has tally $\leq^{\mathrm{p}}_{\mathrm{tt}}$-hard sets.

2. NP has tally $\leq^{\mathrm{p}}_{\mathrm{T}}$-hard sets.

3. NP has sparse $\leq^{\mathrm{p}}_{\mathrm{tt}}$-hard sets.

4. NP has sparse $\leq^{\mathrm{p}}_{\mathrm{T}}$-hard sets.

Nevertheless, the gap between $\mathrm{ZPP}^{\mathrm{NP}}$ and smaller classes (P^{NP}, P^{R}, NP) seems a wide one, and suggests the importance of carefully investigating whether broad classes of formulas formerly having only a $\mathrm{ZPP}^{\mathrm{NP}}$ consequence (via the $\leq^{\mathrm{p}}_{\mathrm{tt}}$ line of Table 1) can be shown to have stronger consequences. Glaßer [Gla00] has achieved exactly this, and Section 4 will present the key ideas of his work.

3 Definitions

For an arbitrary set A we denote the characteristic function of A by χ_A and the cardinality of A by $||A||$. We fix the alphabet $\Sigma = \{0, 1\}$. We denote the set of all words over Σ by Σ^*, and we denote the length of a word w by $|w|$. We usually use *language* to refer to (possibly nonproper) subsets of Σ^*. We call a set $S \subseteq \Sigma^*$ *sparse* if and only if there exists a polynomial p such that, for all $n \geq 0$, it holds that S contains at most $p(n)$ words whose length is no greater than n. For any sets $S_1, \ldots, S_k \subseteq \Sigma^*$ we call the Cartesian product $S = S_1 \times \cdots \times S_k$ *sparse* if and only if there exists a polynomial p such that, for all $n \geq 0$, it holds that S contains at most $p(n)$ elements $(w_1, \ldots, w_k)$ that satisfy $\max\{|w_1|, \ldots, |w_k|\} \leq n$. When dealing with machines we always talk about the deterministic version unless nondeterminism is stated explicitly. We call an algorithm a Δ^{p}_2 *algorithm* if it works in polynomial time and if it has access to a SAT oracle.

In boolean formulas, $\overline{v}$ denotes the negation of the variable v. An *anti-Horn formula* is a boolean formula in conjunctive normal form such that each conjunct contains at most one negative literal. We will be particularly concerned with *k-anti-Horn formulas*; these are, by definition, anti-Horn formulas having exactly one negative literal and at most k positive literals in each conjunct. A conjunct $\alpha = (\overline{v_0} \vee v_1 \vee v_2 \vee \cdots \vee v_m)$ of some k-anti-Horn formula is called a *k-anti-Horn clause* and can be written as $\alpha = (v_0 \rightarrow (v_1 \vee v_2 \vee \cdots \vee v_m))$. We will always assume that $v_1, \ldots, v_m$ are pairwise distinct. We refer to v_0 as the left-hand side of α and to $\{v_1, \ldots, v_m\}$ as the right-hand side of α (RHS(α) for short). Note that we allow

empty right-hand sides, i.e., k-anti-Horn clauses of the form $(v_0 \rightarrow)$; this is equivalent to $(\overline{v_0})$. We write a k-anti-Horn formula as the set of its clauses.

We use the definitions and notations of Ladner, Lynch, and Selman [LLS75] for polynomial-time reductions. However, in case of $\leq^{\mathrm{p}}_{\mathrm{btt}}$ and $\leq^{\mathrm{p}}_{\mathrm{c}}$ we use the following alternative definitions which are equivalent to those in [LLS75]. (For notational simplicity, henceforward whenever we write reduction we will mean polynomial-time reduction.)

Definition 3.1 *Let $A, B \subseteq \Sigma^*$ be arbitrary languages.*

1. A bounded truth-table reduces to B *(denoted $A \leq^{\mathrm{p}}_{\mathrm{btt}} B$) if and only if there exists a constant $k \geq 1$ and a polynomial-time machine that, given an arbitrary word x, computes a list of words $y_1, \ldots, y_k$ and a k-ary boolean formula Φ_x in conjunctive normal form such that each conjunct contains at most k literals, and $x \in A \iff \Phi_x(\chi_B(y_1), \ldots, \chi_B(y_k))$.*

2. A conjunctive truth-table reduces to B *(denoted $A \leq^{\mathrm{p}}_{\mathrm{c}} B$) if and only if there exists a polynomial-time machine that, given an arbitrary word x, computes a (possibly empty, i.e., $m = 0$) collection $Y_x = \{y_1, \ldots, y_m\}$ such that $x \in A \iff (\forall j : 1 \leq j \leq m)[\chi_B(y_i)]$.*

In addition, we define the following.

Definition 3.2 *Let $A, B \subseteq \Sigma^*$ be arbitrary languages.*

1. *Let $k \geq 1$. We say that A k-anti-Horn reduces to B (denoted $A \leq^{\mathrm{p}}_{\mathrm{k\text{–}ah}} B$) if and only if there exists a polynomial-time machine that, given an arbitrary word x, computes a list of words $y_1, \ldots, y_n$ and an n-ary k-anti-Horn formula Φ_x such that $x \in A \iff \Phi_x(\chi_B(y_1), \ldots, \chi_B(y_n))$.*

2. *$A \leq^{\mathrm{p}}_{\mathrm{btt(c)}} B$ if and only if there exists a language X such that $A \leq^{\mathrm{p}}_{\mathrm{btt}} X$ and $X \leq^{\mathrm{p}}_{\mathrm{c}} B$.*

3. *$A \leq^{\mathrm{p}}_{\mathrm{c(btt)}} B$ if and only if there exists a language X such that $A \leq^{\mathrm{p}}_{\mathrm{c}} X$ and $X \leq^{\mathrm{p}}_{\mathrm{btt}} B$.*

4. *$A \leq^{\mathrm{p}}_{\mathrm{d(btt)}} B$ if and only if there exists a language X such that $A \leq^{\mathrm{p}}_{\mathrm{d}} X$ and $X \leq^{\mathrm{p}}_{\mathrm{btt}} B$.*

Proposition 3.3 *Let $A, B \subseteq \Sigma^*$ be arbitrary languages. $A \leq^{\mathrm{p}}_{\mathrm{c(btt)}} B$ if and only if there exist a constant $k \geq 1$ and a polynomial-time machine that, given an arbitrary word x, computes a list of words $y_1, \ldots, y_n$ and an n-ary boolean formula Φ_x in conjunctive normal form such that each conjunct contains at most k literals, and $x \in A \iff \Phi_x(\chi_B(y_1), \ldots, \chi_B(y_n))$.*

We introduce the following abbreviated notation for the case when a set A reduces to a set B in such a way that for each word x, a list of words $y_1, \ldots, y_n$ and an n-ary boolean formula $\Phi_x(a_1, \ldots, a_n)$ are computed such that $x \in A \iff \Phi_x(\chi_B(y_1), \ldots, \chi_B(y_n))$. Instead of considering the list of words $y_1, \ldots, y_n$ and the boolean formula $\Phi_x(a_1, \ldots, a_n)$ as separate objects, we combine them in a natural way into a boolean formula over words, i.e., we replace each occurrence of some variable a_i in $\Phi_x(a_1, \ldots, a_n)$ by the word y_i. For instance, if the reduction of some word x produces the words y_1, y_2, y_3 and the boolean formula $\Phi_x(a_1, a_2, a_3) = (a_1 \vee \overline{a_2}) \wedge (a_1 \vee \overline{a_3})$, then as a simplification we assume that the reduction produces the formula $\Phi_x = (y_1 \vee \overline{y_2}) \wedge (y_1 \vee \overline{y_3})$. A boolean formula over words is said to be *satisfied by a set* $S \subseteq \Sigma^*$ if and only if this formula is satisfied when each occurring word y is replaced by the value $\chi_S(y)$.

4 New Collapses to P^{NP} for Subclasses of Truth-Table Reductions

In this section we present the core result of [Gla00], though with what we hope is a somewhat more accessible proof. In particular, if there exists a sparse $\leq^{\mathrm{p}}_{\mathrm{k\text{–}ah}}$-hard set for NP, then the polynomial hierarchy collapses to P^{NP}. From this result, the same collapse of the polynomial hierarchy from the existence of sparse $\leq^{\mathrm{p}}_{\mathrm{c(btt)}}$-hard or sparse $\leq^{\mathrm{p}}_{\mathrm{d(btt)}}$-hard sets for NP can be shown to also hold (see the end of this section).

Throughout this section, we consider only boolean formulas (respectively, boolean clauses) over words, k will always denote the parameter of $\leq^{\mathrm{p}}_{\mathrm{k\text{–}ah}}$ reductions, p, q, r will denote polynomials, v, w, x, y, z will denote words from Σ^*, $\alpha, \beta, \gamma, \delta, \theta$ will denote k-anti-horn clauses, Γ, Δ, Θ will denote k-anti-horn formulas, and $\mathcal{L}, \mathcal{L}_1, \mathcal{L}_2, \ldots$ will denote lists of k-anti-horn formulas.

We introduce the following binary relation on k-anti-Horn clauses and k-anti-Horn formulas.

Definition 4.1 *For k-anti-Horn clauses* $\gamma = (v_0 \rightarrow (v_1 \vee \cdots \vee v_m))$ *and* $\delta = (w_0 \rightarrow (w_1 \vee \cdots \vee w_n))$*, we write* $\gamma \vdash \delta$ *if and only if* $v_0 \neq w_0$ *or* $\{v_1, \ldots, v_m\} \subseteq \{w_1, \ldots, w_n\}$*. For k-anti-Horn formulas* Γ *and* Δ*, we write* $\Gamma \vdash \Delta$ *if and only if for all* $\delta \in \Delta$ *there is some* $\gamma \in \Gamma$ *with* $\gamma \vdash \delta$*.*

Note that $\vdash$ is reflexive, and it is even transitive if all considered clauses have the same left-hand side. It is easy to see that $\gamma \vdash \delta$ and $\Gamma \vdash \Delta$ are decidable in polynomial time.

Theorem 4.2 *For all* $k \geq 1$*, if* $\mathrm{SAT} \leq^{\mathrm{p}}_{\mathrm{k\text{–}ah}}$ *reduces to a sparse set, then* $\mathrm{PH} = \mathrm{P}^{\mathrm{NP}}$*.*

Proof Let $k \geq 1$ and let S be a sparse set such that $\mathrm{SAT} \leq^{\mathrm{p}}_{\mathrm{k\text{–}ah}} S$. Let p be a polynomial such that for all $n \geq 0$ it holds that $p(n) > 1$, and S contains at most $p(n)$ words having length at most n. Let Φ_x denote the k-anti-Horn formula that occurs when reducing the word x to the sparse set S via the $\leq^{\mathrm{p}}_{\mathrm{k\text{–}ah}}$ reduction mentioned above. Moreover, let q be a polynomial such that for all words x it holds that (i) Φ_x does not contain more than $q(|x|)$ k-anti-Horn clauses and (ii) the length of words appearing in Φ_x is bounded by $q(|x|)$.

Our aim is to show that $\mathrm{NP}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}}$, as that implies that the polynomial hierarchy collapses to P^{NP}. The proof has three parts, and in the first part we show the following claim.

Claim A *There exists a* Δ^{p}_2 *algorithm* `LearnSat` *such that for all* $n \in \mathbb{N}$ *and* $z \in \Sigma^*$ *the computation* `LearnSat`$(0^n, z)$ *returns a k-anti-Horn formula* Γ' *with the following properties:*

(i) each clause $\gamma \in \Gamma'$ *has the left-hand side* z*,*

(ii) Γ' *is satisfied by* S*, and*

(iii) $\Gamma' \vdash \Phi_x$ *for all* $x \in \mathrm{SAT}$ *with* $|x| \leq n$*.*

So the output Γ' of the computation `LearnSat`$(0^n, z)$ allows a forecast concerning queries to SAT of length at most n, in such a way that elements of SAT are treated correctly. Then, in the second part of the proof, we use Claim A to show the following.

Claim B *There exists a* Δ^{p}_2 *algorithm* `LearnAll` *that, on input* 0^n*, returns a list of k-anti-Horn formulas,* $\mathcal{L}_n$*, such that for all words* $x \in \Sigma^*$*,* $|x| \leq n$*, it holds that* $x \in \mathrm{SAT} \iff (\forall \Gamma \in \mathcal{L}_n)[\Gamma \vdash \Phi_x]$*.*

In other words, `LearnAll`(0^n) returns a list of k-anti-Horn formulas, $\mathcal{L}_n$, such that each $x \in$ SAT with $|x| \leq n$ is forecast as "satisfiable" by all elements of $\mathcal{L}$, and for each $x \notin$ SAT with $|x| \leq n$ there is an element of $\mathcal{L}$ giving a negative forecast. So with $\mathcal{L}_n$ we can forecast queries to SAT of length at most n, in such a way that *all* queries are treated correctly. Finally, in the third part of the proof, we use the algorithm `LearnAll` to show that each language from $\mathrm{NP}^{\mathrm{NP}}$ can be accepted by a Δ_2^{p} algorithm. This implies $\mathrm{NP}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}}$.

PART I:

We start with the listing of the algorithm `LearnSat`, which works on inputs of the form $(0^n, z)$ with $n \in \mathbb{N}$ and $z \in \Sigma^*$.

1. Algorithm: `LearnSat`$(0^n, z)$
2. $\Gamma := \{(z \to z)\}$
3. `for` $i := 0$ `to` $\left(p(q(n))^{(k+1)} + 1\right)^k$
4. `if` there exists $x \in$ SAT with $|x| \leq n$ and $\Gamma \nvdash \Phi_x$, `then` determine the smallest such x, call it $\hat{x}$, `else return` Γ and `stop`, `endif`
5. choose a k-anti-Horn clause $\delta \in \Phi_{\hat{x}}$ such that there is no $\gamma \in \Gamma$ with $\gamma \vdash \delta$
6. $\Gamma := (\Gamma \setminus \{\gamma \in \Gamma \mid \delta \vdash \gamma\}) \cup \{\delta\}$
7. `if` $||\Gamma|| = p(q(n))^{k+1}$ `then`
8. choose $\beta, \gamma \in \Gamma$ and a k-anti-Horn clause α such that $\beta \neq \gamma$, α is satisfied by S, $\alpha \vdash \beta$, $\alpha \vdash \gamma$, and α, β, γ have the left-hand side z
9. $\Gamma := (\Gamma \setminus \{\gamma \in \Gamma \mid \alpha \vdash \gamma\}) \cup \{\alpha\}$
10. `endif`

11. `next` i

12. `remark` this step will never be reached

Claim A1 *Let $n \in \mathbb{N}$ and $z \in \Sigma^*$. Then, after the initialization of the variable Γ in step 2, the following holds at the end of each step of the computation* `LearnSat`$(0^n, z)$.

(i) Γ *is a set of k-anti-Horn clauses with* $1 \leq ||\Gamma|| \leq p(q(n))^{k+1}$.

(ii) All words that appear in elements of Γ *are of length at most* $\max\{q(n), |z|\}$.

(iii) All clauses of Γ *have the left-hand side z.*

(iv) If $\gamma_1, \gamma_2 \in \Gamma$ *and* $\gamma_1 \vdash \gamma_2$ *then* $\gamma_1 = \gamma_2$.

Right after step 2 it holds that Γ is a set of k-anti-Horn clauses. This is preserved by step 6, since $\delta \in \Phi_x$ and Φ_x is a k-anti-Horn formula. Also step 9 preserves this property since, by the choice of α and β in step 8, it holds that $\mathrm{RHS}(\alpha) \subseteq \mathrm{RHS}(\beta)$. Moreover, right after step 2 we have $||\Gamma|| = 1$, step 6 increases $||\Gamma||$ at most by 1, and if $||\Gamma|| = p(q(n))^{k+1} > 1$ then step 9 decreases $||\Gamma||$ by 1. This shows the first statement of the claim, and analogously we can show the second one. For the third statement we note that if all clauses of Γ have the left-hand side z, then the choice of δ in step 5 implies that it has also the left-hand side z. Finally, using statement (iii), we can show statement (iv) analogously to the first statement. This proves Claim A1.

Claim A2 *If Γ, right before the execution of step 8, is satisfied by S, then the choice of α, β and γ in step 8 is possible and can be carried out in time polynomial in* $\max\{n, |z|\}$.

So assume that we are right before the execution of step 8, and that Γ is satisfied by S. For $0 \leq i \leq k$ let Γ_i be the set of k-anti-Horn clauses $\gamma \in \Gamma$ such that there appear exactly i words on the right-hand side of γ. From Claim A1(iii) it follows that $||\Gamma_0|| \leq 1$. Since $\sum_{i=0}^{j} h^i < h^{j+1}$ for all $h, j \in \mathbb{N}$ with $h \geq 2$, the condition $||\Gamma|| = p(q(n))^{k+1}$ in step 7 implies that there exists an $m > 0$ such that $||\Gamma_m|| > p(q(n))^m$. We use this fact in the following subprogram that shows a possible implementation of step 8. It assumes a read access to the program variables Γ and z of `LearnSat`, and it returns the required values α, β, γ.

- $\Gamma_i := \{\gamma \in \Gamma \mid ||\mathrm{RHS}(\gamma)|| = i\}$ for $0 \leq i \leq k$
- $j := 0$ and let $\hat{m}$ be the largest $m > 0$ such that $||\Gamma_m|| > p(q(n))^m$
- `repeat`
- `if` $j = 0$ `then` $\alpha_j := (z \rightarrow)$ `else` $\alpha_j := (z \rightarrow (y_1 \vee y_2 \vee \cdots \vee y_j))$ `endif`
- $j := j + 1$ and $\Delta_j := \{\gamma \in \Gamma_{\hat{m}} \mid \alpha_{j-1} \vdash \gamma\}$
- choose a word $y_j \notin \mathrm{RHS}(\alpha_{j-1})$ that appears in a maximum number (note: set n_j to that number) of the right-hand sides of clauses in Δ_j
- `until` $n_j < \frac{||\Delta_j||}{p(q(n))}$
- let $\alpha := \alpha_{j-1}$, choose disjoint k-anti-Horn clauses $\beta, \gamma \in \Delta_j$ and `stop`.

By Claim A1(iii), it holds that all elements of $\Gamma_{\hat{m}} \subseteq \Gamma$ have the left-hand side z. Thus $\Delta_{j+1} = \{\gamma \in \Gamma_{\hat{m}} \mid y_1, y_2, \ldots, y_j \text{ appear at the right hand side of } \gamma\}$. So, as long as the algorithm does not stop, it holds that $||\Delta_{j+1}||$ is equal to n_j, and $||\Delta_{j+1}|| \geq ||\Delta_j||/p(q(n))$. If we reach the $\hat{m}^{\text{th}}$ pass, we have $\alpha_{\hat{m}-1} = (y_1, y_2, \ldots, y_{\hat{m}-1})$ which in turn implies $n_{\hat{m}} = 1$ (note that the right-hand sides of clauses in $\Gamma_{\hat{m}}$ consist of exactly $\hat{m}$ elements, and we do not have two or more identical clauses since $\Gamma_{\hat{m}}$ is a set). So we obtain $||\Delta_{\hat{m}}||/p(q(n)) \geq ||\Gamma_{\hat{m}}||/p(q(n))^{\hat{m}} > 1 = n_{\hat{m}}$, and it follows that the algorithm leaves the loop at the latest after the $\hat{m}^{\text{th}}$ pass. Assume that the algorithm leaves the loop right after the j'^{th} pass with $1 \leq j' \leq \hat{m}$. Then we have $||\Delta_{j'}|| \geq ||\Gamma_{\hat{m}}||/p(q(n))^{j'-1} \geq ||\Gamma_{\hat{m}}||/p(q(n))^{\hat{m}} > 1$ (note that $\Delta_1 = \Gamma_{\hat{m}}$). Thus, when we have left the loop, it holds that $j = j'$, and there exist two disjoint k-anti-Horn clauses $\beta, \gamma \in \Delta_{j'}$. Since the loop's body is passed through at most $\hat{m} \leq k$ times, and each single step can be carried out in time polynomial in $\max\{n, |z|\}$ (note that by Claim A1 we have $||\Gamma|| \leq p(q(n))^{k+1}$ and all words appearing in elements of Γ are of length at most $\max\{q(n), |z|\}$), it follows that the above subprogram works in time polynomial in $\max\{n, |z|\}$. Moreover, it returns distinct $\beta, \gamma \in \Gamma_{\hat{m}} \subseteq \Gamma$ and a k-anti-Horn clause α such that $\alpha \vdash \beta$, $\alpha \vdash \gamma$, and α, β, γ have the left-hand side z.

So it remains to show that if the algorithm stops after the j'^{th} pass, then $\alpha_{j'-1} = (z \rightarrow (y_1 \vee y_2 \vee \cdots \vee y_{j'-1}))$ is satisfied by S. Suppose that the subprogram stops after the j'^{th} pass, and that $\alpha_{j'-1}$ is *not* satisfied by S, i.e., $z \in S$ and $y_1, \ldots, y_{j'-1} \notin S$. We know that all clauses of Γ have the left-hand side z, and by assumption, Γ is

satisfied by S. It follows that each $\gamma \in \Delta_{j'} \subseteq \Gamma$ contains a word from S on its right-hand side (remember that these words are no longer than $q(n)$). There are at most $p(q(n))$ words in S that are no longer than $q(n)$. By a pigeon-hole argument there exists at least one word $y'_{j'} \in S$ such that $|y'_{j'}| \leq q(n)$ and $y'_{j'}$ appears in the right-hand side of at least $||\Delta_{j'}||/p(q(n))$ elements of $\Delta_{j'}$. From $y_1, \ldots, y_{j'-1} \notin S$ it follows that $y'_{j'} \notin \mathrm{RHS}(\alpha_{j'-1})$ and $n_{j'} \geq ||\Delta_{j'}||/p(q(n))$ which is a contradiction to our assumption that the algorithm stops after the j'^{th} pass. This proves Claim A2.

Using a SAT oracle in combination with binary search, step 4 of `LearnSat` can be carried out in time polynomial in $\max\{n, |z|\}$ (note that the size of Γ is polynomially bounded by Claim A1). By Claim A2, also step 8 can be carried out in time polynomial in $\max\{n, |z|\}$. This shows the first part of Claim A, i.e., that `LearnSat` is a Δ_2^{p} algorithm. The remaining part is shown in the following claim.

Claim A3 `LearnSat`$(0^n, z)$ *returns a k-anti-Horn formula Γ' such that*

(i) each $\gamma \in \Gamma'$ has the left-hand side z,

(ii) Γ' is satisfied by S, and

(iii) $\Gamma' \vdash \Phi_x$ for all $x \in \mathrm{SAT}$ with $|x| \leq n$.

Assume for the moment that `LearnSat`$(0^n, z)$ returns some Γ', i.e., `LearnSat`$(0^n, z)$ stops in step 4. From Claim A1 it follows that statement (i) holds, and that Γ' is a k-anti-Horn formula. Clearly, Γ is satisfied by S after its initialization in step 2, and step 6 preserves this property, since $\delta \in \Phi_{\hat{x}}$ and $\hat{x} \in \mathrm{SAT}$. From the choice of α in step 8, it follows that step 9 also preserves the property that Γ is satisfied by S. This shows (ii). Since the algorithm stops in step 4, we have $\Gamma' \vdash \Phi_x$ for all $x \in \mathrm{SAT}$ with $|x| \leq n$. This shows (iii).

So it remains to show that `LearnSat`$(0^n, z)$ stops in step 4. Let us assign a weight $w(\theta)$ to each k-anti-Horn clause θ, $\theta = (v_0 \rightarrow (v_1 \vee v_2 \vee \cdots \vee v_j))$, such that $w(\theta)$ is greater than the sum of the weights of $p(q(n))^{k+1}$ (i.e., the number bounding $||\Gamma||$ in `LearnSat`$(0^n, z)$) k-anti-Horn clauses that have more than j words on their right-hand side. For a k-anti-Horn clause θ and a k-anti-Horn formula Θ we define $w(\theta) = \big(p(q(n))^{(k+1)} + 1\big)^{(k-||\mathrm{RHS}(\theta)||)}$ and $w(\Theta) = \sum_{\theta \in \Theta} w(\theta)$.

By Claim A1(iii), in each step of the computation `LearnSat`$(0^n, z)$ it holds that all clauses of Γ have the left-hand side z. From the fact that δ (in step 6) and α (in step 9) have the left-hand side z, it follows that $\{\gamma \in \Gamma \,\big|\, \delta \vdash \gamma\} = \{\gamma \in \Gamma \,\big|\, \mathrm{RHS}(\delta) \subseteq \mathrm{RHS}(\gamma)\}$

(in step 6) and $\{\gamma \in \Gamma \mid \alpha \vdash \gamma\} = \{\gamma \in \Gamma \mid \mathrm{RHS}(\alpha) \subseteq \mathrm{RHS}(\gamma)\}$ (in step 9). From the choice of δ (respectively, α) it follows that it was not in Γ, right before step 6 (respectively, step 9). For δ this holds by its definition in step 5, and for α this is due to its definition in step 8 and Claim A1(iv). Thus, in step 6 (respectively, step 9) we add one new clause δ (respectively, α) to Γ, and simultaneously we delete all clauses $\gamma \in \Gamma$ such that $\mathrm{RHS}(\delta) \subsetneq \mathrm{RHS}(\gamma)$ (respectively, $\mathrm{RHS}(\alpha) \subsetneq \mathrm{RHS}(\gamma)$). From Claim A1(i) it follows that both steps increase the value of $w(\Gamma)$. Hence, each pass through the loop of $\texttt{LearnSat}(0^n, z)$ increases $w(\Gamma)$. We reach the highest possible value $w(\Gamma) = \big(p(q(n))^{(k+1)} + 1\big)^k$ when $\Gamma = \{(z \rightarrow)\}$. At least at this point $\texttt{LearnSat}(0^n, z)$ stops in step 4. Since we start with $\Gamma = \{(z \rightarrow z)\}$ and $w(\{(z \rightarrow z)\}) \geq 0$, we actually reach the end of the body of the loop at most $\big(p(q(n))^{(k+1)} + 1\big)^k$ times. Thus $\texttt{LearnSat}(0^n, z)$ stops in step 4. This proves Claim A3.

This shows Claim A and completes the first part of the proof of Theorem 4.2.

PART II:

In this part we prove Claim B, i.e., we construct a Δ_2^{p} algorithm `LearnAll` and show that on input 0^n this algorithm will compute a list $\mathcal{L}_n$ of k-anti-Horn formulas $\Gamma_1, \Gamma_2, \ldots, \Gamma_m$ such that the following holds for all words x of length at most n,

$$x \in \mathrm{SAT} \quad \Longleftrightarrow \quad (\Gamma_i \vdash \Phi_x \text{ for all } 1 \leq i \leq m).$$

We give the listing of the algorithm `LearnAll`, which works on inputs of the form 0^n with $n \in \mathbb{N}$.

1. Algorithm: $\texttt{LearnAll}(0^n)$
2. `for` $i = 1$ `to` n
3. $\quad \mathcal{L} := \emptyset$
4. $\quad$ `while` there exists an $x \notin \mathrm{SAT}$ with $|x| \leq i$ such that $\Gamma \vdash \Phi_x$ for all $\Gamma \in \mathcal{L}$
5. $\quad\quad$ let $\hat{x}$ be the smallest x that satisfies this condition

6. for each word v that appears on the left-hand side of some $\gamma \in \Phi_{\hat{x}}$, add $\Theta := \texttt{LearnSat}(0^i, v)$ to the list $\mathcal{L}$

7. `endwhile`

8. $\mathcal{L}_i := \mathcal{L}$

9. `next` i

10. `return` $\mathcal{L}_n$

We want to show that `LearnAll` can be carried out in polynomial time if we are allowed to ask queries to a SAT oracle. To do so, we first show that the number of passes through the while loop is bounded. Then we look into each single step of `LearnAll` and show that it can be carried out in polynomial time (with SAT as an oracle).

Claim B1 *For any fixed i (of step 2), the body of the while loop (steps 4–7) is passed through at most $p(q(i))$ times.*

If the condition in step 4 is satisfied, then, right before step 6, we have $\hat{x} \notin \text{SAT}$, $|\hat{x}| \leq i$, and $\Gamma \vdash \Phi_{\hat{x}}$ for all $\Gamma \in \mathcal{L}$. Since $\hat{x} \notin \text{SAT}$ there exists some $\beta = (v_0 \rightarrow (v_1 \vee v_2 \vee \cdots \vee v_l)) \in \Phi_{\hat{x}}$ that is not satisfied by S, i.e., $v_0 \in S$ and $v_1, v_2, \ldots, v_l \notin S$.

From Claim A it follows that (right before step 6) for each $\Gamma \in \mathcal{L}$ it holds that all clauses of Γ have the same left-hand side and Γ is satisfied by S. Choose an arbitrary $\Gamma \in \mathcal{L}$, and let z be the left-hand side of the elements of Γ. We want to show that $z \neq v_0$. From $\Gamma \vdash \Phi_{\hat{x}}$ it follows that there exists some $\gamma = (z \rightarrow (w_1 \vee w_2 \vee \cdots \vee w_m)) \in \Gamma$ such that $\gamma \vdash \beta$. If $z = v_0$ then we have $\text{RHS}(\gamma) \subseteq \text{RHS}(\beta)$. Thus $w_1, w_2, \ldots, w_m \notin S$ and $z = v_0 \in S$. This contradicts the fact that γ is satisfied by S. So $z \neq v_0$, and it follows that, in each execution of step 6, we add to the list $\mathcal{L}$ at least one Θ whose elements have the left-hand side v_0 such that (i) v_0 does not appear as a left-hand side in some Γ that was on the list $\mathcal{L}$ before, (ii) $|v_0| \leq q(i)$, and (iii) $v_0 \in S$. This proves Claim B1 since the number of words in S of length at most $q(i)$ is bounded by $p(q(i))$.

Claim B2 *Consider the computation* $\texttt{LearnAll}(0^n)$ *for* $n \geq 1$*, and let* $1 \leq j \leq n$*. Then* $\mathcal{L}_j$ *(after its definition in step 8) is a list of* k*-anti-Horn formulas such that for all words* x *of length* $\leq j$ *it holds that* $x \in \mathrm{SAT} \iff (\forall \Gamma \in \mathcal{L}_j)[\Gamma \vdash \Phi_x]$*.*

If step 8 is executed for $i = j$, then the condition in step 4 is false. Hence, for all x of length at most j it holds that $(\forall \Gamma \in \mathcal{L}_j)[\Gamma \vdash \Phi_x] \implies x \in \mathrm{SAT}$. On the other hand, from Claim A it follows that for all x of length at most j we have $x \in \mathrm{SAT} \implies (\forall \Gamma \in \mathcal{L}_j)[\Gamma \vdash \Phi_x]$. This proves Claim B2.

Claim B3 *Using* SAT *as an oracle,* $\texttt{LearnAll}(0^n)$ *can be carried out in time polynomial in* n*.*

By Claim B1, it suffices to show that each single step of $\texttt{LearnAll}(0^n)$ can be carried out in time polynomial in n. First of all we have a look at step 6. Since $|v| \leq q(|\hat{x}|) \leq q(i)$ we obtain from Claim A that $\texttt{LearnSat}(0^i, v)$ can be carried out in time polynomial in i (with SAT as an oracle). It follows that the whole step 6 can be carried out in time polynomial in $i \leq n$. (Note: we are at times being a bit informal regarding the uniformity that holds regarding our "[is]... polynomial in" claims, but this is a common informality and our meaning should be clear.)

Now let us see that we can test the condition in step 4 with one query to SAT. For $i = 1$ this is trivial (without asking any question). If $i > 1$ then we have already computed the list $\mathcal{L}_{i-1}$. By Claim B2, this list allows us to decide $x \in \mathrm{SAT}$ in polynomial time for words x of length at most $i - 1$ (note that by Claim A and Claim B1, the size of $\mathcal{L}_{i-1}$ is polynomial in i). So using the fact that SAT is self-reducible, $\mathcal{L}_{i-1}$ allows us to decide $x \in \mathrm{SAT}$ in time polynomial in i for words x of length at most i. Let N be the polynomial-time machine that achieves this, i.e., on input $(0^i, x, \mathcal{L}_{i-1})$ with $|x| \leq i$ it decides $x \in \mathrm{SAT}$. So the condition in step 4 is equivalent to the following one:

$$(\exists x \in \Sigma^* : |x| \leq i)[(0^i, x, \mathcal{L}_{i-1}) \notin L(N) \wedge (\forall \Gamma \in \mathcal{L})[\Gamma \vdash \Phi_x]].$$

Since this is an NP condition, it can be verified with one query to SAT. This shows that we can test the condition in step 4 with one query to SAT. Analogously one shows that step 5 can be carried out in time polynomial in i by asking queries to SAT (we perform a binary search here). This proves Claim B3.

This completes the second part of the proof of Theorem 4.2, since Claim B follows from the Claims B2 and B3.

PART III:

So far we have shown that if SAT reduces via a $\leq^{\mathrm{p}}_{\text{k–ah}}$ reduction to a sparse set S, then there exists a Δ^{p}_2 algorithm `LearnAll` such that, for all n, `LearnAll`(0^n) returns a list of k-anti-Horn formulas and this list has the nice property that with its help we can answer queries to SAT of length at most n in polynomial time. In the third part of this proof we exploit this property to show that each language from $\mathrm{NP}^{\mathrm{NP}}$ can be accepted by a Δ^{p}_2 algorithm. As is standard, this implies a collapse of the polynomial hierarchy to P^{NP}.

Suppose we are given an arbitrary nondeterministic oracle Turing machine $M^{(\cdot)}$ and a polynomial r bounding its computation time. We define a new machine M' working on inputs of the form $(x, \mathcal{L})$ where x is a word and $\mathcal{L}$ is a list of k-anti-Horn formulas (computed by `LearnAll`). On input $(x, \mathcal{L})$ the machine M' simulates the computation $M^{(\cdot)}(x)$ with the modification that queries q are replaced by tests $(\forall \Gamma \in \mathcal{L})[\Gamma \vdash \Phi_q]$. It is easy to see that $L(M') \in \mathrm{NP}$, since the mentioned test can be carried out in polynomial time.

Now we can describe $N^{(\mathrm{SAT})}$, a *deterministic* polynomial-time Turing machine with SAT oracle, which is such that $N^{(\mathrm{SAT})}$ accepts the same language as does $M^{(\mathrm{SAT})}$. On input x, the machine works as follows:

(i) Determine $\mathcal{L} = \texttt{LearnAll}(0^{r(|x|)})$.

(ii) Accept if and only if $(x, \mathcal{L}) \in L(M')$.

By Claim B, (i) can be done in polynomial time with queries to SAT. Since $L(M') \in \mathrm{NP}$, (ii) can be verified in polynomial time with one query to SAT. This shows that $N^{(\mathrm{SAT})}$ works in deterministic polynomial time. Furthermore, by Claim B we have $q \in \mathrm{SAT} \iff (\forall \Gamma \in \mathcal{L})[\Gamma \vdash \Phi_q]$ for all words q of length at most $r(|x|)$. Since $M^{(\mathrm{SAT})}(x)$ can only asks queries of length at most $r(|x|)$, the computation $M^{(\mathrm{SAT})}(x)$ is equivalent in outcome to that of $M'(x, \mathcal{L})$. It follows that $L(N^{(\mathrm{SAT})}) = L(M^{(\mathrm{SAT})})$. This shows $\mathrm{P}^{\mathrm{NP}} = \mathrm{NP}^{\mathrm{NP}}$, and it follows that $\mathrm{P}^{\mathrm{NP}} = \mathrm{PH}$. ∎

The following theorem strengthens Theorem 4.2, i.e., it holds that $\mathrm{P}^{\mathrm{NP}} = \mathrm{PH}$ even if SAT reduces via a $\leq^{\mathrm{p}}_{\mathrm{c(btt)}}$ reduction to a sparse set. Here the reduction formulas are (unbounded) conjunctions of formulas of bounded length.

Theorem 4.3 *If* SAT *reduces via a* $\leq^{\mathrm{p}}_{\mathrm{c(btt)}}$ *reduction to a sparse set, then* $\mathrm{P}^{\mathrm{NP}} = \mathrm{PH}$.

Proof Suppose SAT reduces via a $\leq^{\mathrm{p}}_{\mathrm{c(btt)}}$ reduction to a sparse set S. From Proposition 3.3 it follows that there exist a constant $k \geq 1$ and a polynomial-time machine that, given an arbitrary word x, computes a boolean formula of words, Φ_x, in conjunctive normal form such that each conjunct contains at most k words, and $x \in \mathrm{SAT}$ if and only if Φ_x is satisfied by S. Observe that there is a bijection $f : (\Sigma^*)^0 \cup (\Sigma^*)^1 \cup \cdots \cup (\Sigma^*)^k \longrightarrow \Sigma^*$ that is polynomial-time computable and polynomial-time invertible. For $0 \leq i \leq k$, let $S_i = f(S^i) = \{f(w_1, w_2, \ldots, w_i) \mid w_1, w_2, \ldots, w_i \in S\}$, and note that, for each $0 \leq i \leq k$, S^i is sparse since S is sparse. Since f is polynomial-time invertible, one can also show that S_i is sparse. Thus $S' = S_0 \cup S_1 \cup \cdots \cup S_k$ is sparse.

We want to show that $\mathrm{SAT} \leq^{\mathrm{p}}_{\mathrm{k\text{–}ah}} S'$. To do so, we consider the conjuncts of Φ_x for some word x. For each conjunct we perform the following transformation:

$$(\overline{v_1} \vee \overline{v_2} \vee \cdots \vee \overline{v_i} \vee w_1 \vee w_2 \vee \cdots \vee w_j) \;\; \mapsto \;\; (\overline{f(v_1, v_2, \ldots, v_i)} \vee f(w_1) \vee f(w_2) \vee \cdots \vee f(w_j)).$$

It is easy to see that this transformation can be carried out in polynomial time. Moreover, it holds that a conjunct is satisfied by S if and only if the transformed conjunct is satisfied by S' (note that $v_1 \notin S \vee \cdots \vee v_i \notin S \iff f(v_1, \ldots, v_i) \notin S'$, and $w \in S \iff f(w) \in S'$ for all words $w, v_1, \ldots, v_i$). This shows that SAT reduces via a $\leq^{\mathrm{p}}_{\mathrm{k\text{–}ah}}$ reduction to a sparse set S'. From Theorem 4.2 it follows that $\mathrm{P}^{\mathrm{NP}} = \mathrm{PH}$. ■

The proofs in this section show even more: It turns out that we can replace SAT by any polynomial-time length-decreasing self-reducible set. (A language T is called polynomial-time length-decreasing self-reducible if and only if there exists a polynomial-time oracle machine $M^{(\cdot)}$ such that the language accepted by $M^{(T)}$ is equal to T, and on input x this machine queries the oracle only about words w with $|w| < |x|$).

Theorem 4.4 *If a polynomial-time length-decreasing self-reducible set T reduces via a $\leq^{\mathrm{p}}_{\mathrm{c(btt)}}$ reduction to a sparse set, then $\mathrm{NP}^T \subseteq \mathrm{P}^{\mathrm{NP}}$.*

Note that if T is a polynomial-time length-decreasing self-reducible set, then also the complement $\overline{T}$ is polynomial-time length-decreasing self-reducible. Moreover, $T \leq^{\mathrm{p}}_{\mathrm{d(btt)}} S$ implies $\overline{T} \leq^{\mathrm{p}}_{\mathrm{c(btt)}} S$ for any set S (just by negation of the reduction formulas). Thus for $\leq^{\mathrm{p}}_{\mathrm{d(btt)}}$ reductions a result analogous to Theorem 4.4 holds.

5 On Theorem 1.1

As noted earlier, Theorem 1.1 follows from the techniques of [CNS95, Theorem 10], as has been mentioned by [vM97,BFT97,Siv00]. For completeness, we note that one can see this, for example, as follows. If SAT disjunctively reduces to some sparse set S, then also the following set $L \in \mathrm{NP}$ disjunctively reduces to S say via some function $f \in \mathrm{FP}$ (cf. [CNS95, Theorem 10]). Let

$$L = \{\langle \psi, 1^m, u, v\rangle \mid m = 2\cdot 3^l,\ u, v \in GF(2^m),\ (\exists \vec{a} = (a_0, \ldots, a_{n-1}))[\psi(\vec{a}) \wedge \sum_{i=0}^{n-1} a_i u^i = v]\},$$

where ψ is an n-ary boolean formula, $\vec{a}$ an assignment for ψ, and u and v are elements of the finite field that has 2^m elements (m is of the form $2\cdot 3^l$ for some $l \geq 0$ to guarantee that this field exists). We assume that for some given word x, $f(x)$ is a set of words (that is interpreted as a disjunction of words).

Now we follow the proof of Theorem 9 which can be found in Appendix B of [CNS95]. Let q be a polynomial such that for all $n', m \geq 0$ and all boolean formulas ψ of size n' it holds that the words in $f(\langle \psi, 1^m, u, v\rangle)$ are of length at most $q(n', m)$. Moreover, let p be a polynomial such that the number of strings in S of length at most $q(n', m)$ is bounded by $p(n', m)$ for all $n', m \geq 0$.

Let ϕ be an n-ary boolean formula of size $n' \geq n$ that has exactly one satisfying assignment; we will determine this assignment. Choose the smallest suitable m (i.e., $m = 2 \cdot 3^l$ for some $l \geq 0$) such that $2^m/p(n', m) \geq n$, call it $\hat{m}$, and let $F = GF(2^{\hat{m}})$. Note that $\hat{m} = O(\log n')$.

Instead of estimating probabilities as it is done in the original proof of Theorem 9 [CNS95] let us proceed as follows. For all $u, v \in F$ we compute $f(\langle \phi, 1^{\hat{m}}, u, v\rangle)$ in polynomial time. Since ϕ has exactly one satisfying assignment, for each $u \in F$ there is a unique $v_u \in F$ such that $\langle \phi, 1^{\hat{m}}, u, v_u\rangle \in L$. For each u, let $S_u = \bigcup_{v \in F} f(\langle \phi, 1^{\hat{m}}, u, v\rangle)$. Now observe the following facts.

1. For each $u \in F$ there exists an $s \in S \cap S_u$ with $|s| \leq q(n', \hat{m})$.

2. The number of elements in S that are of length at most $q(n', \hat{m})$ is bounded by $p(n', \hat{m})$.

It follows that there is some $w \in S$ that appears in at least $2^{\hat{m}}/p(n', \hat{m}) \geq n$ sets S_u.

For each word w that appears in at least n sets $S_{u_1}, \ldots, S_{u_n}$, we do the following: We determine corresponding words $v_1, \ldots, v_n$, such that $w \in f(\langle \phi, 1^{\hat{m}}, u_i, v_i \rangle)$. Then we solve the following equation for $\vec{a} = (a_0, a_1, \ldots, a_{n-1})$ (this is possible since we have a Vandermonde matrix).

$$(a_0, a_1, \ldots, a_{n-1}) \cdot \begin{pmatrix} (u_1)^0 & (u_2)^0 & \cdots & (u_n)^0 \\ (u_1)^1 & (u_2)^1 & \cdots & (u_n)^1 \\ \vdots & \vdots & & \vdots \\ (u_1)^{n-1} & (u_2)^{n-1} & \cdots & (u_n)^{n-1} \end{pmatrix} = (v_1, v_2, \ldots, v_n) \qquad (1)$$

Finally we check whether $\vec{a}$ is a satisfying assignment for ϕ and output $\vec{a}$ in this case.

Note that if we reach some $w \in S$, then all corresponding $\langle \phi, 1^{\hat{m}}, u_i, v_i \rangle$ are elements of L. By the definition of L and the fact that ϕ has exactly one satisfying assignment $(a_0, a_1, \ldots, a_{n-1})$, we have $\sum_{j=0}^{n-1} a_j u_i^j = v_i$ for all i. So if $w \in S$, then (1) is a valid equation. Thus, we really do find the satisfying assignment of ϕ. This shows that $(\exists Q)[\mathrm{USAT}_Q \in \mathrm{P}]$.

Acknowledgments The authors thank Matthias Galota, Edith Hemaspaandra, Harald Hempel, Heinz Schmitz, Klaus W. Wagner, and Gerd Wechsung for helpful discussions and comments.

References

[AHH+93] 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*, pages 1–45. Cambridge University Press, 1993.

[AKM96] V. Arvind, J. Köbler, and M. Mundhenk. Upper bounds for the complexity of sparse and tally descriptions. *Mathematical Systems Theory*, 29:63–94, 1996.

[BFT97] H. Buhrman, L. Fortnow, and L. Torenvliet. Six hypotheses in search of a theorem. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, pages 2–12. IEEE Comptuer Society Press, 1997.

[CHW99] J. Cai, L. Hemaspaandra, and G. Wechsung. Robust reductions. *Theory of Computing Systems*, 32(6):625–647, 1999.

[CNS95] J. Cai, A. Naik, and D. Sivakumar. On the existence of hard sparse sets under weak reductions. Technical Report 95-31, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, July 1995.

[CO97] J. Cai and M. Ogihara. Sparse sets versus complexity classes. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 53–80. Springer-Verlag, 1997.

[GH00] C. Glaßer and L. Hemaspaandra. A moment of perfect clarity I: The parallel census technique. *SIGACT News*, 31(3):37–42, 2000.

[Gla00] C. Glaßer. Consequences of the existence of sparse sets hard for NP under a subclass of truth-table reductions. Technical Report TR 245, Institut für Informatik, Universität Würzburg, Würzburg, Germany, January 2000.

[HOW92] L. Hemachandra, M. Ogiwara, and O. Watanabe. How hard are sparse sets? In *Proceedings of the 7th Structure in Complexity Theory Conference*, pages 222–238. IEEE Computer Society Press, June 1992.

[Kad89] J. Kadin. $P^{NP[\log n]}$ and sparse Turing-complete sets for NP. *Journal of Computer and System Sciences*, 39(3):282–298, 1989.

[KW99] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. *SIAM Journal on Computing*, 28(1):311–324, 1999.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.

[Mah82] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982.

[Mah86] S. Mahaney. Sparse sets and reducibilities. In R. Book, editor, *Studies in Complexity Theory*, pages 63–118. John Wiley and Sons, 1986.

[OW91] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal on Computing*, 20(3):471–483, June 1991.

[Siv00] D. Sivakumar, May 6, 2000. Personal Communication.

[vM97] D. van Melkebeek, 1997. Personal Communication.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[You92] P. Young. How reductions to sparse sets collapse the polynomial-time hierarchy: A primer. *SIGACT News*, 1992. Part I (#3, pages 107–117), Part II (#4, pages 83–94), and Corrigendum to Part I (#4, page 94).