Reductions to Sets of Low Information Content[*]

V. Arvind[†] *Y. Han*[‡] *L. Hemachandra*[‡]
J. Köbler[§] *A. Lozano*[¶] *M. Mundhenk*[§] *M. Ogiwara*[//]
U. Schöning[§] *R. Silvestri*[**] *T. Thierauf*[††]

April 13, 1992

[*]Some of these results will be presented at the 19th International Colloquium on Automata, Languages, and Programming (Vienna, Austria, July 1992).

[†]Department of Computer Science and Engineering, Indian Institute of Technology–Delhi, New Delhi 110016, India. Work done while visiting Universität Ulm. Supported in part by an Alexander von Humboldt Postdoctoral Research Fellowship.

[‡]Department of Computer Science, University of Rochester, Rochester, NY 14627, USA. Supported in part by the National Science Foundation under research grant CCR-8957604.

[§]Abteilung für Theoretische Informatik, Universität Ulm, Oberer Eselsberg, D-W-7900 Ulm, Germany. Supported in part by the DAAD through Acciones Integradas 1991, 313-AI-e-es/zk.

[¶]Department of Software (L.S.I.), Universitat Politècnica de Catalunya, Pau Gargallo 5, E-08028 Barcelona, Spain. Work supported in part by ESPRIT-II Basic Research Actions Program of the EC under Contract No. 3075 (project ALCOM) and by the DAAD through Acciones Integradas 1991, 313-AI-e-es/zk.

[//]Department of Computer Science and Information Mathematics, University of Electro-Communications, 1-5-1, Chofugaoka, Chofu-shi, Tokyo 182, Japan. Work done in part while visiting SUNY–Buffalo. Supported in part by the National Science Foundation under research grant CCR-9002292.

[**]Dipartimento di Scienze dell'Informazione, Università degli Studi di Roma "La Sapienza," 00198 Rome, Italy. Work done in part while visiting the University of Rochester. Supported in part by Ministero della Pubblica Istruzione through "Progetto 40%: Algoritmi, Modelli di Calcolo e Strutture Informative."

[††]Abteilung für Theoretische Informatik, Universität Ulm, Oberer Eselsberg, D-W-7900 Ulm, Germany. Work done in part while visiting the University of Rochester. Supported in part by a DFG Postdoctoral Stipend and by the DAAD through Acciones Integradas 1991, 313-AI-e-es/zk.

1 Overview

This paper is concerned with three basic questions about sparse sets:

Question 1 With respect to what types of reductions might NP have hard or complete sparse sets?[1]

Question 2 If a set A reduces to a sparse set, does it follow that A is reducible to some sparse set that is "simple" relative to A?

Question 3 With respect to what types of reductions might NP have hard or complete sets of low instance complexity, and, relatedly, what is the structure of the class of sets with low instance complexity?

With respect to the first and third questions, intuitively one would expect that even with respect to flexible reductions NP is unlikely to have complete sets whose information content is low. With respect to the second question, one might intuitively feel that the structure imposed on a set by the fact that it reduces to a sparse set makes it plausible that we can indeed find a simple sparse set that can masquerade as the original sparse set. These two intuitions are in many ways certified by the current literature, and by the results of this paper.

The rest of this section summarizes the results of this paper and compares them with previous work.

With regard to Question 1:

- We show that if any NP-complete set conjunctively reduces to a sparse set,[2] then $\mathrm{P} = \mathrm{NP}$. This result, which has been obtained independently by Ranjan and Rohatgi [RR], extends Mahaney's Theorem [Mah82] and is incomparable with the strongest previously known result, which is due to Ogiwara and Watanabe: if any NP-complete set bounded truth-table reduces to a sparse set, then $\mathrm{P} = \mathrm{NP}$ [OW91].

  In fact, a more general result holds regarding the impossibility (if $\mathrm{P} \neq \mathrm{NP}$) of NP-complete sets reducing to sparse sets: if any NP-complete set bounded truth-table reduces to a set that conjunctively reduces to a sparse set, then $\mathrm{P} = \mathrm{NP}$ [AKM].

[1]For the reductions we will discuss, the question of sparse hard sets is equivalent to asking what type of reductions might reduce many-one complete sets for NP to some sparse set; we will often use this formulation.

[2]A conjunctive reduction from A to B means that there is a Turing machine with oracle B that accepts A, and the Turing machine's acceptance mechanism is that it accepts if and only if every string it queries is a member of B [LLS75].

- One might ask whether in the above-mentioned result of Ogiwara and Watanabe the bounded truth-table case is optimal, or whether it can be extended by making the bound on the number of queries bigger than constant, for example, by making it some function that is $\omega(\log n)$. We show that there are relativized worlds in which the boolean hierarchy does not collapse and yet there are tally NP-complete with respect to such reductions. This provides relativized upper bounds to the work of Ko, Orponen, Schöning, and Watanabe [KOSW86,Orp90,OKSW], of Ogiwara and Watanabe [OW91], and of the current paper. Our result strengthens a result of Homer and Longpré [HL91], who independently of the work of this paper showed that there are relativized worlds in which there are sparse sets that are NP-complete with respect to such bounds and yet (relativized) $\mathrm{P} \neq \mathrm{NP}$.

With regard to Question 2:

- In the context of recent comparisons between equivalence and reducibility to sparse sets [AHOW,GW91], it is interesting to know, for various classes of sets that reduce to sparse sets, the complexity of the easiest sparse sets to which such sets reduce. We show that any set A that disjunctively reduces (respectively, disjunctive bounded truth-table reduces, 2-truth-table reduces) to a sparse set in fact disjunctively reduces (respectively, disjunctive bounded truth-table reduces, 2-truth-table reduces) to a sparse set that is in $\mathrm{P}^{\mathrm{NP}^A}$ (respectively, $\mathrm{P}^{\mathrm{NP}^A[\log]}$, $\mathrm{P}^{\mathrm{NP}^A[\log]}$).[3] Thus, for such sets, reducing to some sparse set implies reducing to some relatively simple sparse set. The nearest previous result is one of Allender, Hemachandra, Ogiwara, and Watanabe [AHOW]: If $\mathrm{P} = \mathrm{NP}$ and set A 2-truth-table reduces to a sparse set, then A truth-table reduces to some sparse set that itself truth-table reduces to A. However, A does not *two*-truth-table reduce to the particular sparse set constructed in [AHOW]. Via census-functions, graph-coloring, and the Erdős-Rado sunflower lemma, our techniques avoid the level of explicit coding (and thus the complexity of reduction) required by previous methods.

With regard to Question 3:

- We completely characterize the sets of low instance complexity (that is, the class IC[log, poly] [KOSW86,Orp90,OKSW]) in terms of reductions to tally sets:

[3]The [log] means that there is an $\mathcal{O}(\log n)$ bound on the number of calls made to the NP^A oracle (see [Wag90]).

IC[log, poly] is exactly the class of sets that both disjunctively and conjunctively reduce to tally sets.

- We show that Orponen's result [Orp90] on 1-truth-table-complete sets for NP generalizes to disjunctive and conjunctive reductions: If $\mathrm{P} \neq \mathrm{NP}$ and A is $\leq_c^p$-hard or $\leq_d^p$-hard for NP, then $A \notin \mathrm{IC}[\log, \mathrm{poly}]$.

Section 3 discusses the results associated with Question 1. Section 4 discusses the results associated with Question 2. Section 5 discusses the results associated with Question 3. The Appendix provides a simple direct proof of one of our corollaries.

2 Notation

A set T is said to be a *tally* set if $T \subseteq 0^*$. $S^{=n}$ will denote the length n strings in S (when S is our alphabet, Σ, we will simply write Σ^n), and $S^{\leq n}$ (respectively, $S^{<n}$) will denote the strings in S of length at most (respectively, strictly less than) n. A set S is said to be *sparse* if it has at most polynomially many elements at each length: S is sparse if and only if for some polynomial p it holds that $(\forall n)[||S^{=n}|| \leq p(n)]$. We use TALLY and SPARSE to represent, respectively, the classes of tally and sparse sets. Tally and sparse sets have come to play a large role in modern complexity theory (see, e.g., the surveys [Mah86, Mah89,HOW]).

The reductions discussed in this paper are polynomial-bounded reductions defined by Ladner, Lynch, and Selman [LLS75]. Table 1 lists the abbreviations we will use for various types of reductions.

We will use the following notation to describe downward closures of classes under various reductions.

Notation 2.1 [BK88,AHOW] For any reducibility $\leq_r^p$ and any class of sets $\mathcal{C}$, let $\mathrm{R}_r^p(\mathcal{C}) = \{A \mid (\exists B \in \mathcal{C})[A \leq_r^p B]\}$.

The interrelations among $\mathrm{R}_r^p(\mathrm{SPARSE})$ classes have been studied by Book and Ko [BK88], Ko [Ko89], Allender, Hemachandra, Ogiwara, and Watanabe [AHOW], and Gavaldà [Gav92]. All the inclusions shown in Figure 1 are proper.[4]

[4] $\mathrm{R}_m^p(\mathrm{SPARSE}) \subsetneq \mathrm{R}_b^p(\mathrm{SPARSE})$ and $\mathrm{R}_m^p(\mathrm{SPARSE}) \subsetneq \mathrm{R}_c^p(\mathrm{SPARSE})$ are respectively from [BK88] and [Ko89]. From the result of [AHOW] that $\mathrm{R}_b^p(\mathrm{SPARSE}) \subseteq \mathrm{R}_d^p(\mathrm{SPARSE})$ and the result of [Gav92] that $\mathrm{R}_c^p(\mathrm{SPARSE}) \not\subseteq \mathrm{R}_d^p(\mathrm{SPARSE})$, it follows that $\mathrm{R}_b^p(\mathrm{SPARSE}) \subsetneq \mathrm{R}_b^p(\mathrm{R}_c^p(\mathrm{SPARSE}))$. From the result of [Ko89] that $\mathrm{R}_b^p(\mathrm{SPARSE}) \not\subseteq \mathrm{R}_c^p(\mathrm{SPARSE})$, it follows that $\mathrm{R}_c^p(\mathrm{SPARSE}) \subsetneq \mathrm{R}_b^p(\mathrm{R}_c^p(\mathrm{SPARSE}))$. It remains only to

Name	Notation
many-one	$\leq_m^p$
one truth-table	$\leq_{1\text{-}tt}^p$
k truth-table	$\leq_{k\text{-}tt}^p$
k disjunctive (truth-table or Turing)	$\leq_{k\text{-}d}^p$
bounded (truth-table or Turing)	$\leq_b^p$
$f(n)$ truth-table	$\leq_{f(n)\text{-}tt}^p$
conjunctive (truth-table or Turing)	$\leq_c^p$
disjunctive (truth-table or Turing)	$\leq_d^p$
bounded conjunctive (truth-table or Turing)	$\leq_{bc}^p$
bounded disjunctive (truth-table or Turing)	$\leq_{bd}^p$
Turing	$\leq_T^p$
nondeterministic conjunctive (truth-table or Turing)	$\leq_c^{NP}$

Table 1: Polynomial-time Reductions

3 Sets Reducing to Sparse Sets

The study of sparse complete sets was sparked by the conjecture of L. Berman and J. Hartmanis [BH77] that there are no sparse NP-complete sets; they were motivated to make this conjecture since if it fails then there are NP-complete sets that are not polynomial-time isomorphic (and at that time they conjectured that all NP-complete sets were polynomial-time isomorphic, though recent work has dimmed hopes on that issue [JY85,KMR89]).

The first result along the lines of their sparseness conjecture was P. Berman's proof that P = NP if some subset of 0^* is NP-complete [Ber78]. This result was quickly followed by Fortune's proof that if there is a sparse coNP-complete set, then P = NP [For79]. Finally, Mahaney obtained the striking result that P = NP if any NP-complete set many-one reduces to a sparse set [Mah82].

Although Mahaney obtained the complete collapse of the polynomial hierarchy in the

show that $\mathrm{R}_b^p(\mathrm{SPARSE}) \subsetneq \mathrm{R}_d^p(\mathrm{SPARSE})$. [AHOW] shows that $\mathrm{R}_b^p(\mathrm{SPARSE}) \subseteq \mathrm{R}_d^p(\mathrm{SPARSE})$, and from this it follows that $\mathrm{coSPARSE} \subseteq \mathrm{R}_d^p(\mathrm{SPARSE})$, which in turn implies $\mathrm{SPARSE} \subseteq \mathrm{R}_c^p(\mathrm{coSPARSE})$, and hence $\mathrm{R}_c^p(\mathrm{SPARSE}) \subseteq \mathrm{R}_c^p(\mathrm{R}_c^p(\mathrm{coSPARSE})) = \mathrm{R}_c^p(\mathrm{coSPARSE})$. If $\mathrm{R}_b^p(\mathrm{SPARSE}) = \mathrm{R}_d^p(\mathrm{SPARSE})$, then, since $\mathrm{R}_b^p(\cdot)$ classes are closed under complement, $\mathrm{R}_b^p(\mathrm{SPARSE}) = \mathrm{R}_c^p(\mathrm{coSPARSE})$. The previous two sentences imply $\mathrm{R}_c^p(\mathrm{SPARSE}) \subseteq \mathrm{R}_b^p(\mathrm{SPARSE})$, and thus $\mathrm{R}_c^p(\mathrm{SPARSE}) \subseteq \mathrm{R}_d^p(\mathrm{SPARSE})$, contradicting the result of [Gav92].

case of many-one reducibility, possible collapses in the case of more flexible reducibilities have remained an active research area. For the case of Turing reductions, it is known that the existence of sparse Turing-complete sets for NP would collapse the polynomial hierarchy to $\mathrm{P}^{\mathrm{NP}[\log]}$ [Kad89], and the existence of sparse Turing-hard sets for NP would collapse the polynomial hierarchy to $\Sigma_2^p \bigcap \Pi_2^p$ [KL80]; both these results are known to be essentially optimal with respect to relativizable proof techniques [Kad89,Hel86].

As just noted, for the cases of many-one and Turing reductions the consequences of sparse NP-complete sets are well-understood. However, with respect to reductions whose strength lies between Turing and many-one reductions, the question of extending Mahaney's many-one result has proved considerably more challenging. For the case of bounded truth-table reductions, Ukkonen [Ukk83] generalized Berman's result [Ber78] by showing that if there is a tally bounded truth-table hard set for NP, then P = NP. Yesha [Yes83] generalized Fortune's result [For79] by showing that if there is a sparse bounded positive truth-table hard set for coNP, then P = NP. Yesha also (partially) generalized Mahaney's Theorem [Mah82] by showing that if there is a sparse bounded positive truth-table complete set for NP, then P = NP. These results regarding bounded truth-table reductions have been recently subsumed by Ogiwara and Watanabe ([OW91], see also [Wat88] and [JY90, footnote 9]), who successfully extended Mahaney's result and showed that if there is a sparse bounded truth-table hard set for NP, then P = NP.

For the case of conjunctive reductions, Ukkonen [Ukk83] and Yap [Yap83] generalized Fortune's result [For79] by showing that if there is a sparse conjunctive hard set for coNP, then P = NP. Yap [Yap83] also (partially) generalized Mahaney's Theorem [Mah82] by showing that if there is a sparse set that is both conjunctive and disjunctive complete for NP, then P = NP. However, in the decade since Mahaney's Theorem, it has remained an open question whether his result can be extended to the case of conjunctive reductions. Section 3.1 resolves this question.

3.1 NP, PP, and $\mathrm{C}_{=}\mathrm{P}$

We show that if there is a sparse set that is conjunctive hard for NP, then P = NP (Corollary 3.4). In fact, in Corollary 3.5 we establish that if $\mathrm{NP} \subseteq \mathrm{R}_b^p(\mathrm{R}_c^p(\mathrm{SPARSE}))$ then P = NP, thus extending the result of Ogiwara and Watanabe [OW91] (see Figure 1). Theorems 3.2, 3.8, and 3.9 originally appeared in [AKM].

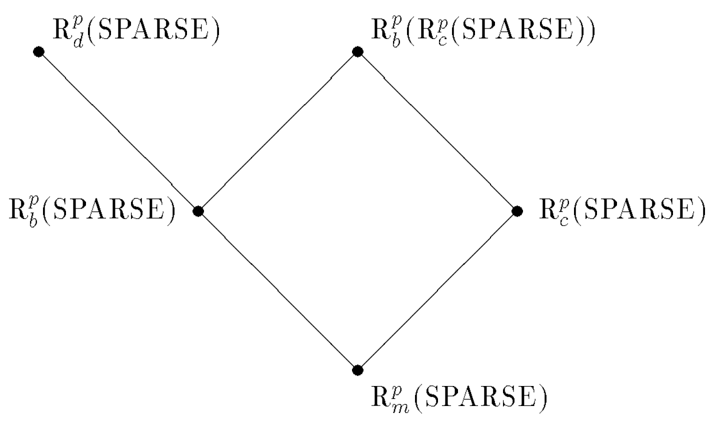

Figure 1: Inclusion structure of some reduction classes to sparse sets; all inclusions indicated are proper (see Footnote 4).

Definition 3.1 [OW91] Let A be in NP, let W be a set in P, and let q be a polynomial such that $A = \{\, x \,|\, (\exists\, w \in \Sigma^{q(|x|)})\,[\langle x, w\rangle \in W]\,\}$. For $x \in A$ let $w_{max}(x) = \max\{\, w \in \Sigma^{q(|x|)} \,|\, \langle x, w\rangle \in W\,\}$. We will say that $Left(A) = \{\,\langle x, w\rangle \,|\, x \in A,\ w \in \Sigma^{q(|x|)}$ and $w \leq w_{max}(x)\}$ is the *left set of A*.[5]

Theorem 3.2 If $A \in$ NP and $Left(A) \in R_b^p(R_c^p(\text{SPARSE}))$, then A is in P.

Although the next result is a direct consequence of the above theorem, a simple direct proof for it is given in Appendix A.

Theorem 3.3 If $A \in$ NP and $Left(A) \in R_c^p(\text{SPARSE})$, then A is in P.

From Theorem 3.3, it immediately follows that Mahaney's Theorem generalizes to the case of conjunctive reductions. Corollary 3.4 was obtained independently by Ranjan and Rohatgi [RR].

Corollary 3.4 If any NP-complete set conjunctively reduces to a sparse set, then P = NP.

Indeed, $Left(A) \leq_m^p A$ for any NP-complete set A, and thus the above theorems apply to the case in which some NP-complete (or NP-hard) set reduces (via the reductions named above) to some sparse set.

[5]The left set tacitly depends on the particular witness relation chosen.

Corollary 3.5 If any NP-complete set is in $\mathrm{R}^p_b(\mathrm{R}^p_c(\mathrm{SPARSE}))$, then P = NP.

The rest of this section is devoted to proving Theorem 3.2, and to giving applications of the obtained results to other complexity classes. The following characterization, due to Hausdorff, of the boolean closure of certain classes of sets plays a central role in our proof of Theorem 3.2.

Theorem 3.6 [Hau14,Wec85] Let K be any class of sets closed under finite unions and intersections that includes $\emptyset$ and Σ^*. Let $BC(K)$ be the closure of K under finite union, finite intersection, and complement. Every $A \in BC(K)$ can be represented as $A = \bigcup_{i=1}^{k}(A_{2i-1} \cap \overline{A_{2i}})$, where $A_j \in K$ (for $1 \leq j \leq 2k$) and $A_1 \supseteq A_2 \supseteq \cdots \supseteq A_{2k}$.

In order to use this characterization for sets in $R^p_b(R^p_c(\mathrm{SPARSE}))$, we need to show the closure of $R^p_c(\mathrm{SPARSE})$ under finite unions and intersections. By the recent result of Buhrman, Longpré, and Spaan [BLS92] showing that SPARSE $\subsetneq R^p_c(\mathrm{TALLY})$, it follows that $R^p_b(R^p_c(\mathrm{SPARSE})) = R^p_b(R^p_c(\mathrm{TALLY}))$. The following lemma is straightforward and is stated without proof.

Lemma 3.7 $R^p_c(\mathrm{TALLY})$ is closed under finite unions and intersections.

Now we are ready to prove Theorem 3.2.

Proof of Theorem 3.2:
Let q be a polynomial and let P_A be a polynomial-time set such that $A = \{x \mid (\exists w \in \Sigma^{q(|x|)})[\langle x, w\rangle \in P_A]\}$. Recall that $Left(A) = \{\langle x, w\rangle \mid x \in A \ \wedge \ w \in \Sigma^{q(|x|)} \ \wedge \ w \leq w_{max}\}$, where $w_{max} = \max\{w \in \Sigma^{q(|x|)} \mid \langle x, w\rangle \in P_A\}$. In the following we describe an algorithm for testing membership in A, that computes w_{max} (the lexicographically largest witness, if it exists) by a breadth-first search of the tree of prefixes of all potential witnesses. In order to do this we use the set $prefix(Left(A)) = \{\langle x, y\rangle \mid (\exists z)[\langle x, yz\rangle \in Left(A)]\}$. Each prefix y actually represents the interval of all possible extensions of y to length $q(|x|)$. It is not hard to see that $prefix(Left(A))$ is many-one equivalent to $Left(A)$ and thus $prefix(Left(A)) \in R^p_b(R^p_c(\mathrm{SPARSE})) = R^p_b(R^p_c(\mathrm{TALLY}))$.

Using Lemma 3.7 and the representation theorem of Hausdorff stated as Theorem 3.6, it is easy to see that there exists a tally set T and sets $C_i \in R^p_c(T)$, $D_i \in R^p_d(T)$ such that $prefix(Left(A)) = \bigcup_{i=1}^{k}(C_i \cap D_i)$ and $C_1 \supseteq \overline{D_1} \supseteq C_2 \supseteq \overline{D_2} ... \supseteq C_k \supseteq \overline{D_k}$.

Let f_i be the conjunctive reduction that witnesses $C_i \in R^p_c(T)$ and g_i be the disjunctive reduction that witnesses $D_i \in R^p_d(T)$. Without loss of generality, we assume that these reductions are all computable in time bounded by a fixed polynomial p.

We first give an intuitive overview of the polynomial-time[6] algorithm recognizing A. As stated above, this algorithm performs a breadth-first search through the tree of witness prefixes for an input x. Let x be an element of A, and let $N = \{y_1, \ldots, y_t\}$ be a lexicographically ordered set of prefixes (all the same length) that includes the prefix of w_{max} of that length. We exploit some crucial properties of the Hausdorff representation $\bigcup_{i=1}^{k}(C_i \cap D_i)$ of $prefix(Left(A))$ for the design of a procedure pruning N to a polynomially size-bounded set that still includes the prefix of w_{max}.

Let y_m be the prefix of w_{max} in $\{y_1, \ldots, y_t\}$. Then, letting $d = 1$ and $l(0) = 1$, it holds that

$$\{\langle x, y_{l(d-1)}\rangle, \ldots, \langle x, y_m\rangle\} \subseteq C_d$$

Let $r(d)$ be the largest index r such that $\{\langle x, y_{l(d-1)}\rangle, \ldots, \langle x, y_r\rangle\} \subseteq C_d$ and let $l(d)$ be the least index l such that $1 \leq l \leq r(d) + 1$ and $\{\langle x, y_l\rangle, \ldots, \langle x, y_{r(d)}\rangle\} \subseteq \overline{D_d}$. Observe that since $\{\langle x, y_{l(d-1)}\rangle, \ldots, \langle x, y_m\rangle\} \subseteq C_d$ it follows that $r(d) \geq m$. Similarly, since $\{\langle x, y_{m+1}\rangle, \ldots, \langle x, y_{r(d)}\rangle\} \subseteq \overline{D_d}$, it holds that $l(d) \leq m + 1$. We consider the following two cases separately.

1. $\langle x, y_m\rangle \in D_d$.

   Then $l(d) = m + 1$ since $y_m \notin \{y_{l(d)}, \ldots, y_{r(d)}\}$, i.e., $l(d) > m$.

2. $\langle x, y_m\rangle \notin D_d$. (This case is only possible if $d < k$.)

   In this case, $y_m \in \{y_{l(d)}, \ldots, y_{r(d)}\}$. Since $\{\langle x, y_{l(d)}\rangle, \ldots, \langle x, y_m\rangle\} \subseteq prefix(Left(A))$ but $\{\langle x, y_{l(d)}\rangle, \ldots, \langle x, y_m\rangle\} \subseteq \overline{D_d}$, it follows that $\{\langle x, y_{l(d)}\rangle, \ldots, \langle x, y_m\rangle\} \subseteq C_{d+1}$, and the above analysis can be repeated.

If we could compute the prefixes $y_{l(d)}$ and $y_{r(d)}$ defined above in polynomial time, we could use the above properties in order to design a recursive procedure that collects all the prefixes $y_{l(d)-1}$ found in the recursive calls. This procedure would return a small subset of N containing y_m. Starting with $N = \{\epsilon\}$, the overall algorithm can repeatedly use such a pruning step at each level of the tree of possible witness prefixes by first expanding all the prefixes y in N to $y0$ and $y1$ (thus doubling N) and then pruning N back to a small subset. In that way, the algorithm finally computes a small subset of $\Sigma^{q(|x|)}$ that, if $x \in A$, contains w_{max}.

Although we cannot explicitly compute the required prefixes $y_{l(d)}$ and $y_{r(d)}$, instead we can compute, given $y_{l(d-1)}$, in polynomial time (polynomially size-bounded) sets $J_{right}(d)$

[6]It is implicit in this section that polynomial time and polynomial size always mean polynomial in $|x|$.

and $J_{left}(d)$ of prefixes such that $y_{r(d)} \in J_{right}(d)$ and $y_{l(d)} \in J_{left}(d)$. This suffices since for each prefix candidate $y \in J_{left}(d)$, the search for $y_{l(d+1)}$ can be done recursively. Since the depth of the recursion is a constant, namely k, the resulting sets $J_{left}(d)$ of candidates for $y_{l(d)}$ still have polynomially bounded cardinality.

We now describe the algorithm in detail. The algorithm calls a recursive pruning procedure PRUNE, which in turn calls two functions SEARCH-RIGHT and SEARCH-LEFT. SEARCH-RIGHT is used to search for candidates for $y_{r(d)}$ that are to the right of previously found candidates for $y_{l(d-1)}$, resulting in a polynomially size-bounded set $J_{right}(d)$ containing $y_{r(d)}$. SEARCH-LEFT is used to search to the left of the prefixes in $J_{right}(d)$, to form a polynomially size-bounded set $J_{left}(d)$ containing $y_{l(d)}$.

```
SEARCH-RIGHT($d, N, y_l, x$)
(* returns a set $J \subseteq N = \{y_1, \ldots, y_t\}$ that includes the largest prefix
$y_r \in N$ such that $\{\langle x, y_l\rangle, \ldots, \langle x, y_r\rangle\} \subseteq C_d$ *)
begin
  $J := \{y_t\}$
  **for** $j := 0$ **to** $p(|x|)$ **do**
    $J := J \cup \{y_h \mid y_{h+1}$ is the smallest $y$ in $N$ s.t. $y \geq y_l$ and $0^j \in f_d(\langle x, y\rangle)\}$
  **end**
  **return** $J$
end
```

Claim 1 Function SEARCH-RIGHT(d, N, y_l, x), when called with parameter $y_l = y_{l(d-1)}$, returns a set J containing $y_{r(d)}$.

Proof of Claim 1: There are two cases. If $r(d) = t$, then $y_{r(d)}$ is clearly in the returned set J. Otherwise, since $\{\langle x, y_{l(d-1)}\rangle, \ldots, \langle x, y_{r(d)}\rangle\} \subseteq C_d$ and $\langle x, y_{r(d)+1}\rangle \not\in C_d$, all the queries in the sets $f_d(\langle x, y_{l(d-1)}\rangle), \ldots, f_d(\langle x, y_{r(d)}\rangle)$ are in T but at least one query 0^j in $f_d(\langle x, y_{r(d)+1}\rangle)$ is not in T. Thus $y_{r(d)+1}$ is the smallest prefix y in N such that $y \geq y_{l(d-1)}$ and $0^j \in f_d(\langle x, y\rangle)$, i.e., $y_{r(d)}$ is included in J in the j^{th} run of the for-loop. ∎

```
SEARCH-LEFT($d, N, y_r, x$)
(* returns a set $J \subseteq N = \{y_1, \ldots, y_t\}$ that includes the smallest prefix
$y_l \in N$ such that $\{\langle x, y_l\rangle, \ldots, \langle x, y_r\rangle\} \subseteq \overline{D_d}$ *)
begin
```

```
  $J := \{y_1\}$
  for $j := 0$ to $p(|x|)$ do
    $J := J \cup \{y_h \mid y_{h-1}$ is the largest $y$ in $N$ s.t. $y \leq y_r$ and $0^j \in g_d(\langle x, y\rangle)\}$
  end
  return $J$
end
```

Claim 2 Function SEARCH-LEFT(d, N, y_r, x), when called with parameter $y_r = y_{r(d)}$, returns a set J containing $y_{l(d)}$.

Proof of Claim 2: Again, there are two cases. If $l(d) = 1$, then $y_{l(d)}$ is clearly in the returned set J. Otherwise, since $\{\langle x, y_{l(d)}\rangle, \dots, \langle x, y_{r(d)}\rangle\} \subseteq \overline{D_d}$ and $\langle x, y_{l(d)-1}\rangle \in D_d$, all the queries in the sets $g_d(\langle x, y_{l(d)}\rangle), \dots, g_d(\langle x, y_{r(d)}\rangle)$ are outside of T but at least one query 0^j in $g_d(\langle x, y_{l(d)-1}\rangle)$ is in T. Thus $y_{l(d)-1}$ is the largest prefix y in N such that $y \leq y_{r(d)}$ and $0^j \in g_d(\langle x, y\rangle)$, i.e., $y_{l(d)}$ is included in J in the j^{th} run of the for-loop. ∎

```
PRUNE$(N, J'_{left}, d, x)$
(* returns a subset of $N = \{y_1, \dots, y_t\}$ that contains the prefix $y_m$ of $w_{max}$ if
$y_m \in N \cap C_d$ and $\{y_l, \dots, y_m\} \subseteq C_d$ for a $y_l \in J'_{left}$ with $l \leq m$ *)
begin
  if $d = k + 1$ then return $\emptyset$ end
  $J_{right} := \emptyset$
  for each $z \in J'_{left}$ do
    $J_{right} := J_{right} \cup$ SEARCH-RIGHT$(d, N, z, x)$
  end
  $J_{left} := \emptyset$
  for each $z \in J_{right}$ do
    $J_{left} := J_{left} \cup$ SEARCH-LEFT$(d, N, z, x)$
  end
  return $\{y_{l-1} \mid y_l \in J_{left}\} \cup$ PRUNE$(N, J_{left}, d + 1, x)$
end
```

Claim 3 If $y_m \in Nu$, $\langle x, y_m\rangle \in C_d$, and $y_{l(d-1)} \in J'_{left}$ then function PRUNE(N, J'_{left}, d, x) returns a set I containing y_m.

Proof of Claim 3: If $y_m \in N$ and $\langle x, y_m\rangle \in C_d$ then $\langle x, y_m\rangle$ is also in the sets $\overline{D_{d-1}}, \ldots, \overline{D_1}$. By the above analysis (since case 2 always happens up to $d-1$) it follows that $\{\langle x, y_{l(d-1)}\rangle, \ldots, \langle x, y_m\rangle\} \subseteq C_d$. Since $y_{l(d-1)} \in J'_{left}$, using Claim 1, $y_{r(d)}$ is included in J_{right} by the call of SEARCH-RIGHT($d, N, y_{l(d-1)}, x$). Using Claim 2, $y_{l(d)}$ is included in J_{left} by the call of SEARCH-LEFT($d, N, y_{r(d)}, x$). Now we can prove by induction that y_m is included in the set returned by PRUNE. If $\langle x, y_m\rangle \in D_d$ (which must be true in the base case $d = k$), then $y_m = y_{l(d)-1}$ and y_m is included in the set returned by PRUNE. If $\langle x, y_m\rangle \notin D_d$ then $\langle x, y_m\rangle$ is in C_{d+1} and we can use the induction hypothesis. ∎

We complete the algorithm with a description of the main program.

```
input $x$
begin
$N := \{\epsilon\}$
for $i := 1$ to $q(|x|)$ do
   $N := \{y0 \mid y \in N\} \cup \{y1 \mid y \in N\}$  (* expand the prefixes to length $i$ *)
   $N :=$ PRUNE($N, \{y_1\}, 1, x$)
end
(* $N$ now includes $w_{max}$ if $x \in A$ *)
if there is a witness for $x$ in $N$ then accept else reject end
end
```

In order to prove the correctness of the algorithm it suffices to observe that it follows from Claim 3 that the prefix y_m of w_{max} is included in the pruned set returned by PRUNE($N, \{y_1\}, 1, x$), provided that y_m is in N. Also, since the sets returned by SEARCH-RIGHT and SEARCH-LEFT are bounded in size by $p(|x|)+2$, it follows inductively that the set J_{left} computed by PRUNE at level d is bounded in size by $(p(|x|)+2)^{2d}$. Thus, since the depth of recursion of function PRUNE is bounded by a constant, the finally returned set—being the union of all the J_{left}'s—is polynomially size-bounded, and it is easy to see that the algorithm runs in polynomial time. ∎

The Hausdorff characterization of boolean closures of classes of sets has turned out to be also useful in proving related results concerning randomized reductions and nondeterministic reductions to sparse sets (see [AKM]).

We now briefly discuss the application of the above results to the classes UP, PP and $\mathrm{C}_=\mathrm{P}$. Since for every set $A \in$ UP it holds that $Left(A)$ is in UP, it also follows that if UP is contained in $\mathrm{R}^p_b(\mathrm{R}^p_c(\mathrm{SPARSE}))$ then P = UP. This strengthens the results of

Watanabe [Wat91], who showed that if P $\neq$ UP then there exists a set in UP that does not many-one polynomial-time reduce to any sparse set.

Consider the set $\{\langle x, m\rangle \mid$ there are at least m satisfying assignments for $x\}$, which has properties similar to left sets and is complete for PP. Under the assumption that this set is in $\mathrm{R}^p_b(\mathrm{R}^p_c(\mathrm{SPARSE}))$, we can use the algorithm described in the proof of Theorem 3.2 to compute in polynomial time a set of numbers that includes $\#\mathrm{SAT}(x)$, the number of satisfying assignments of formula x. Now we can use the result of Cai and Hemachandra [CH91] and Toda (see [ABG90]) that P = PP if there is an FP function that computes on input x a set of numbers that includes $\#\mathrm{SAT}(x)$. Alternatively, Theorem 3.8 could be proved along the same lines as Theorem 3.9.

Theorem 3.8 If PP is contained in $\mathrm{R}^p_b(\mathrm{R}^p_c(\mathrm{SPARSE}))$, then P = PP.

We can also show that if $\mathrm{C}_=\mathrm{P}$ is contained in $\mathrm{R}^p_b(\mathrm{R}^p_c(\mathrm{SPARSE}))$ then P = $\mathrm{C}_=\mathrm{P}$.

Theorem 3.9 If $\mathrm{C}_=\mathrm{P}$ is contained in $\mathrm{R}^p_b(\mathrm{R}^p_c(\mathrm{SPARSE}))$ then P = $\mathrm{C}_=\mathrm{P}$.

Proof of Theorem 3.9:
There exist complete sets in $\mathrm{C}_=\mathrm{P}$ that are one word-decreasing self-reducible [OL]. Balcazár has shown that every one word-decreasing self-reducible set in $\mathrm{R}^p_T(\mathrm{SPARSE})$ is in Σ^p_2 [Bal90]. So it follows from the assumption of the theorem that $\mathrm{C}_=\mathrm{P} \subseteq \Sigma^p_2$. Furthermore, since coNP $\subseteq \mathrm{C}_=\mathrm{P}$, if $\mathrm{C}_=\mathrm{P} \subseteq \mathrm{R}^p_b(\mathrm{R}^p_c(\mathrm{SPARSE}))$ then also NP $\subseteq \mathrm{R}^p_b(\mathrm{R}^p_c(\mathrm{SPARSE}))$, and it follows from Corollary 3.5 that P = Σ^p_2. ∎

3.2 Mod$_k$P

Mod$_k$P [CH90,BGH90] is the class of sets L for which there is a nondeterministic polynomial-time Turing machine M such that for every x, $x \in L$ if and only if the number of accepting computation paths of M on x is not a multiple of k.

By applying a different proof technique we obtain results similar to Corollary 3.4 for the classes Mod$_k$P, $k \geq 2$.

Definition 3.10 A set L is *rotatively one word-decreasing self-reducible* if there exist a deterministic polynomial-time Turing transducer M and a polynomial p satisfying the following conditions:

1. L is a set of strings of the form $\langle x, y, i\rangle$ with $i < p(|x|)$,

2. for every x and for every y, z, there is some $d < p(|x|)$ such that for every $i < p(|x|)$, $\chi_L(\langle x, y, i\rangle) = \chi_L(\langle x, z, i \circ d\rangle)$, where $i \circ d = (i + d) \bmod p(|x|)$, and

3. for every x and y, either

   (a) $M(x, y)$ outputs $\chi_L(\langle x, y, 0\rangle) \cdots \chi_L(\langle x, y, p(|x|) - 1\rangle) \in \Sigma^{p(|x|)}$, or

   (b) $M(x, y)$ outputs $d < p(|x|)$ such that for every $i < p(|x|)$, $\chi_L(\langle x, y, i\rangle) = \chi_L(\langle x, \mathrm{pred}(y), i \circ d\rangle)$.

Theorem 3.11 Any rotatively one word-decreasing self-reducible set that conjunctively reduces to a sparse set is in P.

Since each set in Mod_kP, $k \geq 2$, is many-one reducible to a rotatively one word-decreasing self-reducible set in Mod_kP (in fact, these are essentially the strictly one word-decreasing self-reducible sets of [OL] that are complete for Mod_kP) we have the following corollary.

Corollary 3.12 For each $k \geq 2$: if Mod_kP has a sparse conjunctively-hard set then P $=$ Mod_kP.

Proof of Corollary 3.12
Let L be an arbitrary set in Mod_kP. Let W be in P and let p be a polynomial such that for all x

$$x \in L \iff ||\{\, y \in \Sigma^{p(|x|)} \mid \langle x, y\rangle \in W \,\}|| \not\equiv 0 (\mathrm{mod} k).$$

Define A to be the set of strings of the form $\langle x, y, i\rangle$ such that z is not equivalent to i modulo k, where z is the number $y' \in \Sigma^{p(|x|)}$ such that $y' \leq y$ and $\langle x, y'\rangle \in W$. Note that

- A is rotatively one word-decreasing self-reducible and
- for every x, $x \in L$ iff $\langle x, 1^{m(|x|)}, 0\rangle \in A$, and thus, L is many-one reducible to A.

So, if Mod_kP has a sparse conjunctively-hard set S, then $A \leq_c^p S$, and so $A \in$ P. Thus $L \in$ P. ∎

Proof of Theorem 3.11:
Let L be a rotatively one word-decreasing self-reducible set, as certified by machine M and polynomial p as in Definition 3.10. Suppose that L is R_c-reducible to a sparse set S via a function f. We will give a polynomial-time algorithm for L. Without loss of generality, we may assume that there exist polynomials q and r such that for every w, $f(w)$ is an encoding of a set in $\Sigma^{\leq q(|w|)}$ and for every n, $||S^{\leq q(n)}|| \leq r(n)$. Let $w_0 = \langle x_0, y_0, i_0\rangle$ be a

fixed input whose membership in L we are testing. As we have fixed the input, let p, q, and r denote $p(|x_0|)$, $q(|w_0|)$, and $r(|w_0|)$, respectively. Let $I = \{0, \cdots, q-1\}$. For $a, b \in I$, let $a \circ b = (a+b) \bmod q$.

For a string y, let $\alpha(y)$ denote $\chi_L(\langle x, y, 0\rangle) \cdots \chi_L(\langle x, y, p-1\rangle)$. For a string $u \in \Sigma^p$ and $d \in I$, let $\rho(u, d)$ denote $u_{d+1} \cdots u_p u_1 \cdots u_d$. Note that, by definition, for every y, it holds that

$$(*)\ M(x_0, y) \text{ is either } \alpha(y) \text{ or } d \in I \text{ such that } \alpha(y) = \rho(\alpha(\mathrm{pred}(y)), d).$$

For a string y and $i \in I$, let $w(y, i)$ denote $\langle x_0, y, i\rangle$. An *argument sequence* is a sequence $(A_0, \cdots, A_{p-1})$ with each $A_i \subseteq \Sigma^{\leq q}$. An argument sequence $\mathcal{A} = (A_0, \cdots, A_{p-1})$ is said to be *correct* for a string y if for every $i \in I$, $w(y, i) \in L$ iff $A_i \subseteq S$. Note that if $\mathcal{A} = (A_0, \cdots, A_{p-1})$ is correct for y, then for every $i \in I$ with $||A_i|| > r$, $w(y, i) \notin L$ because $||S^{\leq q}|| \leq r$. For an argument sequence $\mathcal{A} = (A_0, \cdots, A_{p-1})$ and $d \in I$, $\mathcal{A}(d)$ denotes an argument sequence $(A_d, \cdots, A_{p-1}, A_0, \cdots, A_{d-1})$. Let $\mathcal{A} = (A_0, \cdots, A_{p-1})$ and $\mathcal{B} = (B_0, \cdots, B_{p-1})$ be given two argument sequences. We write $\mathcal{B} \subseteq_L \mathcal{A}$ to denote that for every $i \in I$ with $||A_i|| \leq r$, $B_i \subseteq A_i$. Also, $\mathcal{A} \bigcup \mathcal{B}$ denotes $(A_0 \bigcup B_0, \cdots, A_{p-1} \bigcup B_{p-1})$. For y, Φ_y denotes the argument sequence defined by $(f(w(y, 0)), \cdots, f(w(y, p-1)))$. Note that Φ_y is correct for y for every y.

Claim 4 Let $\mathcal{A} = (A_0, \cdots, A_{p-1})$ and $\mathcal{B} = (B_0, \cdots, B_{p-1})$ be argument sequences that are correct for y and z, respectively. Let $d \in I$ be such that $\mathcal{B}(d) \subseteq_L \mathcal{A}$. Then, $\rho(\alpha(z), d) = \alpha(y)$.

Proof of Claim 4 Let $\mathcal{A}$, $\mathcal{B}$, y, z, and d be as in the hypothesis. Since $\mathcal{A}$ is correct for y, for every $i \in I$, $w(y, i) \in L$ iff $(||A_i|| \leq r$ and $A_i \subseteq S)$. Since $\mathcal{B}(d) \subseteq_L \mathcal{A}$, for every $i \in I$ with $||A_i||$, $B_{i \circ d} \subseteq A_i$, and thus, for every $i \in I$ with $w(y, i) \in L$, $w(z, i \circ d) \in L$. Since $\mathcal{B}$ is correct for z, this implies that for every $i \in I$ with $w(y, i) \in L$, $w(z, i \circ d) \in L$. Since the number of i with $w(y, i) \in L$ is equal to the number of i with $w(z, i) \in L$, we have that for every $i \in I$ with $w(y, i) \notin L$, $w(z, i \circ d) \notin L$. Thus, $\rho(\alpha(z), d) = \alpha(y)$ ∎

Claim 5 Let $\mathcal{A} = (A_0, \cdots, A_{p-1})$ and $\mathcal{B} = (B_0, \cdots, B_{p-1})$ be argument sequences that are correct for y and z, respectively. Let $d \in I$ be such that $\mathcal{B}(d) \subseteq_L \mathcal{A}$. Then

1. if $M(x_0, z) = u$ for some $u \in \Sigma^p$, then $\rho(u, d) = \alpha(y)$, and
2. if $M(x_0, z) = e$ for some $e \in I$, then $\rho(\alpha(\mathrm{pred}(z)), e \circ d) = \alpha(y)$, and $\mathcal{A} \bigcup \Phi_y(e \circ d) = (A_0 \bigcup f(w(\mathrm{pred}(z), e \circ d)), \cdots, A_{p-1} \bigcup f(w(\mathrm{pred}(z), (p-1) \circ (e \circ d))))$ is correct for y.

Proof of Claim 5 Let $\mathcal{A}$, $\mathcal{B}$, y, z, and d be as in the hypothesis. From Claim 4, we have $\alpha(y) = \rho(\alpha(z), d)$. Suppose that $M(x_0, z) = u$ for some $u \in \Sigma^p$. By definition, $u = \alpha(z)$, and thus $\alpha(y) = \rho(u, d)$.

On the other hand, suppose that $M(x_0, z) = e$ for some $e \in I$. As discussed previously, it holds that $\alpha(z) = \rho(\alpha(\text{pred}(z)), e)$. By taking $\rho(\cdot, d)$ of both sides, we have $\rho(\alpha(z), d) = \rho(\rho(\alpha(\text{pred}(z)), e), d)$, and thus, $\alpha(y) = \rho(\alpha(\text{pred}(z)), e \circ d)$.

By definition, $(f(w(\text{pred}(z), 0)), \cdots, f(w(\text{pred}(z), p-1)))$ is correct for $\text{pred}(z)$. Thus, for every $i \in I$, $w(y, i) \in L$ iff $A_i \subseteq S$ iff $w(y, i \circ d \circ e) \in L$ iff $f(w(y, i \circ d \circ e)) \subseteq S$. So, $w(y, i) \in L$ iff $A_i \bigcup f(w(y, i \circ d \circ e)) \subseteq S$. This proves the claim. ▮

Now we define the algorithm. We operate on d, a string y, and an argument sequence $\mathcal{A} = (A_0, \cdots, A_{p-1})$. Initially, we set y to y_0, d to 0, and A_i to $f(w(y_0, i))$ for each $i \in I$, so that the following two conditions are satisfied:

(c1) $\mathcal{A}$ is correct for y_0, and

(c2) $\rho(\alpha(y), d) = \alpha(y_0)$.

The main part of the algorithm is the repetition of two steps defined below. We require that at the beginning of the first step both (c1) and (c2) hold.

First, we find $z \leq y$ such that

(d1) for some $c \in I$, $\Phi_z(c) \subseteq_L \mathcal{A}$, and

(d2) either $M(x_0, z) \in \Sigma^p$ or there is no e such that $\Phi_{\text{pred}(z)}(e) \subseteq_L \mathcal{A}$.

Note that, under the assumption that (c1) and (c2) hold at the beginning of this step, we have (i) every $z \leq y$ satisfies at least one of these conditions, (ii) $z = y$ satisfies (d1), and (iii) $z = \epsilon$ satisfies (d2). So, by executing a simple divide-and-conquer algorithm over $[\epsilon, y]$, we can easily find z for which (d1) and (d2) are satisfied. We set c to one of the values establishing (c1).

Next, we compute $M(x_0, z)$. If this is in Σ^p, from Claim 5, $\alpha(y_0) = \rho(\alpha(z), c) = \rho(M(x_0, z), c)$. So, $w_0 = w(y_0, i_0)$ is in L iff $w(z, i_0 \circ c) \in L$, and this is easily computed from the output of M. Hence, if this is the case, we obtain $\chi_L(w_0)$ and we accept w_0 iff it is 1. If $M(x_0, z)$ is not in Σ^p, i.e., it is in I, let e be the value and we set A_i to $A_i \bigcup f(w(z, i \circ c \circ e))$ for every i, and set $\mathcal{A}$ to the resulting sequence. We claim that $\mathcal{A}$ is correct for y_0. This is seen as follows. Clearly, $\Phi_{\text{pred}(z)}$ is correct for $\text{pred}(z)$. Since $M(x_0, z) = e$, $\Phi_{\text{pred}(z)}(e)$ is correct for z. Since $\Phi_z(c) \subseteq_L \mathcal{A}$, $\Phi_{\text{pred}(z)}(c \circ e)$ is correct for y_0. So $\Phi_{\text{pred}(z)}(c \circ e) \bigcup \mathcal{A}$ is correct for y_0.

After executing the above two steps, we set y to pred(z) and d to $c \circ e$. At this point, if A_{i_0} has more than r elements, then since $A_{i_0} \subseteq S$ iff $w_0 \in L$ and it is impossible that $A_{i_0} \subseteq S$, we reject w_0 and terminate the algorithm. If A_{i_0} has at most r elements, we go back to the start of the first step and repeat these two steps.

Note that each time $\mathcal{A}$ is updated there is some $i \in I$ such that A_i gets at least one new element. So the loop is executed at most $\mathcal{O}(pr)$ times, and this is bounded by some polynomial in $|w_0|$. It is not hard to see that all the other operations can be done in time polynomial in $|w_0|$, and the algorithm correctly decides whether $w_0 \in L$. So $L \in$ P. ∎

3.3 Nearly Near-Testable Sets

For nearly near-testable sets [HH91], the class of sets that have "implicit" polynomial-time membership tests, a similar result holds.

Definition 3.13 [HH91] A set A is *nearly near-testable* if there exists a polynomial-time function $N\colon \Sigma^* \to \{\mathit{true}, \mathit{false}, \leftrightarrow, \not\leftrightarrow\}$ such that for every x one of the following holds ($x - 1$ denotes the string lexicographically preceding x):

- $N(x) = \mathit{true}$ and $x \in A$
- $N(x) = \mathit{false}$ and $x \notin A$
- $x \neq \epsilon$ and $N(x) = \leftrightarrow$ and $(x \in A \iff x - 1 \in A)$
- $x \neq \epsilon$ and $N(x) = \not\leftrightarrow$ and $(x \in A \iff x - 1 \notin A)$

Theorem 3.14 Any nearly near-testable set that conjunctively reduces to a sparse set is in P.

Proof of Theorem 3.14:
Let A be a nearly near-testable set such that $A \leq_c^p S$, where S is a sparse set. Let f be the function that witnesses the reduction. Let P be the function associated with the nearly near-testability of A, as mentioned above. In this proof we consider the conjunctive reduction function f on input x as generating a set $f(x)$ of strings. Thus, by definition, for any string x it holds that $x \in A$ iff $f(x) \subseteq S$.

Since the function f is computable in polynomial time, and since there is a polynomial that bounds the number of strings of each length in S, there exists some polynomial $p(n)$ that bounds the number of different strings belonging to S that can appear in $f(x)$ on inputs x up to length n, i.e., for all n,

$$||\{z \in S \,\big|\, (\exists x : |x| \leq n)[z \in f(x)]\}|| \ \leq\ p(n)$$

Now we define a function SEARCH(z_1,z_2,$R(z)$), which is used in the polynomial-time algorithm that decides A. Given a polynomial-time computable predicate R on Σ^*, and two strings z_1 and z_2 such that $z_1 > z_2$ and $R(z_1)$ and $\neg R(z_2)$ hold, the function outputs a string y, with $z_1 \leq y \leq z_2$, such that $R(y)$ and $\neg R(y-1)$ hold.

This is done by a binary search. For a string x let ord(x) be the natural number whose binary representation is $1x$.

```
SEARCH($z_1$,$z_2$,$R(z)$)
begin
  $a := z_1$
  $b := z_2$
  while $a \neq b + 1$ do
    set $c$ to the string such that $\mathrm{ord}(c) = \lfloor(\mathrm{ord}(a) + \mathrm{ord}(b))/2\rfloor$
    if $R(c)$ then $a := c$ else $b := c$ end
  end
  return $a$
end
```

Since at each iteration of the loop SEARCH halves the range of the search, the initial (possibly exponential) range is reduced to one in a polynomial number of iterations. It can be seen that $R(a)$ and $\neg R(b)$ are loop invariants. So when SEARCH exits the loop, the string found satisfies the desired conditions.

With the help of SEARCH we will develop a polynomial-time algorithm that decides A. The algorithm uses two sets C_1 and C_2 with the initial values $C_1 = f(x)$ and $C_2 = \emptyset$. The algorithm has the following three invariants.

$$x \in A \Longleftrightarrow C_1 \subseteq S \tag{1}$$

$$\text{if } C_2 \neq \emptyset : x \notin A \Longleftrightarrow C_2 \subseteq S \tag{2}$$

$$\text{if } C_2 \neq \emptyset : C_1 \subseteq S \Longleftrightarrow C_2 \not\subseteq S \tag{3}$$

Consider the following polynomial-time predicate Q.

$$Q(x) = [f(x) \subseteq C_1 \vee f(x) \subseteq C_2]$$

For any string w such that $Q(w)$ is true, we define $s(w)$ as the smaller integer $i \in \{1,2\}$ such that $f(w) \subseteq C_i$. We say that a string w is *connected* if $Q(w)$ is true and $f(w) \subseteq S \Longrightarrow C_{s(w)} \subseteq S$.

The algorithm uses a function CONNECT repeatedly in a loop in order to either increase the size of one of the sets C_1 and C_2 or to find a connected string whose membership in A is already determined. In the case that the cardinality of one of the sets C_1 or C_2 exceeds $p(|x|)$, the algorithm can correctly decide the membership of x in A by the above invariants.

The function CONNECT has two inputs u, v, and assumes that u is connected and $u > v$. CONNECT can decide membership of x (considered as a global variable) in A or, otherwise, its output is a tuple (b_1, b_2, b_3, w) that satisfies one of the following conditions (b_1, b_2 and b_3 are boolean variables and w is a string, $v \leq w \leq u$):

(c1) $b_1 = \mathit{true}$ and v is connected

(c2) $b_1 = \mathit{false}$, $b_2 = \mathit{true}$ and $\neg Q(w)$

(c3) $b_1, b_2 = \mathit{false}$ and $(w \in A \Longleftrightarrow b_3 = \mathit{true})$.

We now define the function CONNECT. Note that when $b_1 = \mathit{true}$ it is not necessary to define the rest of the tuple, and when $b_1 = \mathit{false}$ and $b_2 = \mathit{true}$, the value of b_3 is not used; the value of the irrelevant components of the tuple will be denoted with —.

```
CONNECT($u$,$v$)
begin
(1) if not $Q(v)$ then return ($\mathit{false}$,$\mathit{true}$,—,$v$) end
    $K := f(v)$
    $w := v$
    repeat
          if $f(u) \subseteq K$ then $w := u$
          else
                $w :=$ SEARCH($u$,$w$,[$f(z) \not\subseteq K$])
(2)             if $N(w) \in \{\mathit{true}, \mathit{false}\}$ then return ($\mathit{false}$,$\mathit{false}$,$N(w)$,$w$) end
                if $f(w) \subseteq C_{s(v)}$ then
(3)                   if $N(w) = \not\leftrightarrow$ then accept iff $s(v) = 2$
```

```
(4)               else $K := K \cup f(w)$
                  end
              else
                  if $f(w) \subseteq C_{3-s(v)}$ then
(5)                   if $N(w) = \leftrightarrow$ then return (*false*,*false*,*true*,$w$)
(6)                   else return (*true*,—,—,—)
                      end
(7)               else return (*false*,*true*,—,$w$)
                  end
              end(*if*)
          end (*if $f(w) \subseteq K$ *)
      until $w = u$
(8)   if $s(u) = s(v)$ then return (*true*,—,—,—)
(9)   else if $s(u) = 1$ then *accept* else *reject* end
      end
end
```

Note that in lines (3) and (9) CONNECT can actually accept or reject the input x. The following claim proves the correctness of CONNECT.

Claim The function CONNECT either decides x correctly or it returns a tuple fulfilling condition (c1), (c2), or (c3).

Proof of Claim: In order to prove the claim we analyze certain parts of function CONNECT. The statements of the algorithm are referred to by their line numbers.

Line (4)

We show that this line ensures the following is an invariant for the repeat loop.

$$f(v) \subseteq S \Longrightarrow K \subseteq S \tag{4}$$

The invariant is clearly true when the loop is first entered. In order to prove that it holds at the beginning of each iteration we need only check that it is preserved after executing line (4), since at any other numbered line in the loop, the loop is exited. At line (4) we can observe (by the nested if statements) that $w \in A \Longleftrightarrow w-1 \in A$ and that $f(w-1) \subseteq K$.

Suppose that $f(v) \subseteq S$. We assume that, in this case, $K \subseteq S$. By the preceding observations, $f(w-1) \subseteq S$ and thus $f(w) \subseteq S$. So we have that the new K, which is the union of the old one and $f(w)$, must be included in S. This proves that (4) is an invariant.

Lines (1), (2) and (7)

In these cases, it is easy to see that the output satisfies the conditions.

Line (2)

In line (2), membership of w in A is given by $N(w)$. This corresponds to condition (c3).

Line (3)

Since $N(w) = \not\to$, either $f(w) \not\subseteq S$ or $f(w-1) \not\subseteq S$. In either case we get the same conclusion.

- Suppose that $f(w) \not\subseteq S$. Since $f(w) \subseteq C_{s(v)}$, it follows that $C_{s(v)} \not\subseteq S$.
- Suppose that $f(w-1) \not\subseteq S$. The string w has been computed by SEARCH, which assured us that $f(w-1) \not\subseteq K$. We have that $K \not\subseteq S$. By invariant (4), it follows that $f(v) \not\subseteq S$, and hence $C_{s(v)} \not\subseteq S$.

In both cases $C_{s(v)} \not\subseteq S$ (and thus $C_{3-s(v)} \subseteq S$). Hence, x is in A iff $s(v) = 2$.

Line (5)

We show that w is in A, i.e., the output fulfills condition (c3). The call to SEARCH ensures that $f(w-1) \subseteq K$. The conditions in the two preceding if statements ensure that $f(w) \subseteq C_{3-s(v)}$ and $w \in A \iff w-1 \in A$. Suppose that $w \notin A$. Then $w-1 \notin A$ and consequently $K \not\subseteq S$ (using the above observations). Invariant (4) implies that v is not in A, hence $C_{s(v)} \not\subseteq S$ and $C_{3-s(v)} \subseteq S$. Thus $w \in A$, which is a contradiction.

Line (6)

Observe that $f(w-1) \subseteq K$ and that the two if statements preceding line (5) assure that $f(w) \subseteq C_{3-s(v)}$ and that $w \in A \iff w-1 \notin A$. This last fact can also be written as $f(w) \subseteq S \iff f(w-1) \not\subseteq S$.

We prove the implication $f(v) \subseteq S \implies C_{s(v)} \subseteq S$, which shows that v is connected. Suppose that $f(v) \subseteq S$. By invariant (4) we have that $K \subseteq S$, hence $f(w-1) \subseteq S$. This implies that $f(w) \not\subseteq S$ and $C_{3-s(v)} \not\subseteq S$. Thus, by invariant (3), $C_{s(v)} \subseteq S$.

Lines (8) and (9)

Note that at the end of each iteration of the loop it is true that $f(w) \subseteq K$, and at the end of the loop $w = u$. Thus when the repeat loop is exited we have $f(u) \subseteq K$. Since u is connected it follows from invariant (4) that $f(v) \subseteq S$ implies $f(u) \subseteq S$, which in turn implies $C_{s(u)} \subseteq S$.

Thus if $s(u) = s(v)$ it follows that v is connected and the output in line (8) is correct. On the other hand, if $s(u) \neq s(v)$, consider the following two cases.

- If $v \in A$ then $f(v) \subseteq S$. This in turn implies that $f(u) \subseteq S$. Since u is connected, it follows that $C_{s(u)} \subseteq S$.
- If $v \notin A$ then $C_{s(v)} \not\subseteq S$. By invariant (3) and since $s(u) \neq s(v)$ it follows that $C_{s(u)} \subseteq S$.

In either case we have $C_{s(u)} \subseteq S$. Hence CONNECT correctly decides the input x.

End of Proof of Claim

Now we are ready to give the algorithm for A.

```
input $x$
begin
   $C_1 := f(x)$
   $C_2 := \emptyset$
   $b'_2 := \mathit{false}$
   $b'_3 := \chi_A(\epsilon)$
   while $(||C_1|| \leq p(|x|))$ and $(||C_2|| \leq p(|x|))$ do
      $a := x$
      $b := \epsilon$
      while $b + 1 < a$ do
         set $c$ to the string such that $\mathrm{ord}(c) = \lfloor(\mathrm{ord}(a) + \mathrm{ord}(b))/2\rfloor$
         $(b_1, b_2, b_3, w) :=$ CONNECT$(a, c)$
         if $b_1$ then $a := c$
         else
            $b := w$
            $b'_2 := b_2$
            $b'_3 := b_3$
         end
      end (*while*)
      if $N(a) \in \{\mathit{true}, \mathit{false}\}$ then
         if $(N(a) = \mathit{true})$ xor $(s(a) = 2)$ then accept else reject end
      else
         if $b'_2$ then ADD$(b)$
         else if $(s(a) = 1)$ xor $b'_3$ xor $(N(a) = \leftrightarrow)$ then accept else reject end
```

```
            end
          end(*if*)
        end (*while*)
        if $||C_1|| \leq p(|x|)$ then accept else reject end
      end
```

It is easy to see that at the end of the inner loop, a is connected and $a = b + 1$. Furthermore, either $\neg Q(b)$ holds (in the case $b'_2 = \mathit{true}$) or (in the case $b'_2 = \mathit{false}$) membership of b in A is determined by b'_3.

If $N(a) \in \{\mathit{true}, \mathit{false}\}$ then it is easy to see that membership of x in A is correctly decided.

If $b'_2 = \mathit{false}$, then the membership of a in A is given by b'_3 and function P, and hence, since a is connected, membership of x is also determined.

Finally, if b'_2 is true, at least one new element can be added to C_1 or C_2 while preserving the invariants. The method for adding new elements is the following:

```
ADD(y)
begin
  if $(N(y+1) = \leftrightarrow)$ then $C_{s(y+1)} := C_{s(y+1)} \cup f(y)$
  else $C_{3-s(y+1)} := C_{3-s(y+1)} \cup f(y)$
  end
end
```

It is easy to see that the running time of the algorithm is polynomially bounded. What remains is to show that invariant (1) and invariant (2) are preserved by ADD (invariant (3) is a consequence of these two invariants).

C_1 is increased only if either (a) $N(a) = \leftrightarrow$ and $s(a) = 1$, or (b) $N(a) = \not\leftrightarrow$ and $s(a) = 2$. In either case it is easy to see that $C_1 \subseteq S$ if and only if $f(b) \subseteq S$.

Next we consider the case when C_2 is increased. ADD increases C_2 if either (a) $N(a) = \not\leftrightarrow$ and $s(a) = 1$, or (b) $N(a) = \leftrightarrow$ and $s(a) = 2$. In order to prove invariant (2), we consider two subcases: When ADD includes elements in C_2 for the first time, since a is connected, it holds that $s(a) = 1$. Hence $N(a) = \not\leftrightarrow$ and it follows that $x \notin A$ if and only if $f(b) \subseteq S$. In subsequent increases of C_2 it is not hard to see that $C_2 \subseteq S$ if and only if $f(b) \subseteq S$.

This shows that invariant (1), invariant (2), and invariant (3) are preserved by ADD. ∎

3.4 An Upper Bound

As discussed previously, Ogiwara and Watanabe [OW91] showed that if for some k NP has a k-truth-table hard sparse set then P = NP. It is natural to ask whether the result of Ogiwara and Watanabe can be extended to truth-tables that have non-constant bounds on their number of queries. This section shows that, even with respect to the weakened conclusion that the boolean hierarchy [CGH+88,CGH+89] collapses, the Ogiwara-Watanabe result cannot be improved to $\omega(\log n)$-bounded truth-table reductions by any relativizable proof technique. Our result strengthens the work of Homer and Longpré [HL91], who independently of the work of this paper showed that there are relativized worlds in which there are sparse sets that are NP-complete with respect to such bounds and yet P $\neq$ NP. The strongest earlier result was the result of Kadin ([Kad89], see also [IM89,CGH+89]) that for every nice function $f(n) = o(\log n)$ there are relativized worlds in which the polynomial hierarchy does not collapse to $\mathrm{P}^{\mathrm{NP}[f(n)]}$ yet NP has sparse Turing complete sets. Since NP has sparse Turing complete sets if and only if NP has sparse truth-table complete sets—this can be seen either directly, via the parallel census technique discussed later in this subsection, or as a consequence of Hartmanis's sparse set that is truth-table complete for the sparse sets in NP ([Har83], only Turing completeness is stated, but Hartmanis's set is clearly truth-table complete)—Kadin's result applies equally well to the (seemingly stronger) truth-table case.

The boolean hierarchy [CGH+88,CGH+89] is the closure of NP under boolean operations; equivalently, it is the class of sets that can be accepted by finite amounts of hardware applied to NP predicates. For the purposes of this paper, we will define the hierarchy via one if its normal forms—namely, as the union of differences of NP sets.

Definition 3.15 ([CGH+88])

1. For $k \geq 1$, the *k-th* level of the boolean hierarchy is defined by: $L \in \mathrm{NP}(k)$ if and only if there exist $L_1, \ldots, L_k \in \mathrm{NP}$ such that

$$L = \begin{cases} (L_1 - L_2) \cup \ldots \cup (L_{k-2} - L_{k-1}) \cup L_k & \text{if } k \text{ is odd} \\ (L_1 - L_2) \cup \ldots \cup (L_{k-1} - L_k) & \text{if } k \text{ is even.} \end{cases}$$

2. $\mathrm{coNP}(k) = \{L \mid \overline{L} \in \mathrm{NP}(k)\}$.

3. $\mathrm{BH} = \bigcup_{k \geq 1} \mathrm{NP}(k)$, defines the *boolean hierarchy*.

Like the polynomial hierarchy, the boolean hierarchy does have downward separation; if for some k_0 it holds that $\mathrm{NP}(k_0) = \mathrm{coNP}(k_0)$, then $\mathrm{NP}(k_0) = \mathrm{BH}$ [CGH+88]. In such a case, we say that the boolean hierarchy *collapses* (to level k_0).

Theorem 3.16 If f is a polynomial-time computable, nondecreasing function such that $f(n) = \omega(\log n)$, then there exist a set A and a tally set T such that BH^A does not collapse and T is $\leq^{p,A}_{f(n)\text{-}tt}$-complete for NP^A.

One part of the proof of Theorem 3.16 is a diagonalization ensuring that the boolean hierarchy does not collapse. Such a diagonalization was first accomplished by Cai et al. [CGH+88] via "forcing" machines to accept. Unfortunately, each "forced" acceptance potentially adds a polynomial number of strings to the oracle. As will become clear in our proof of Theorem 3.16, adding so many strings would taint our requirement that the relativized world have complete sets with respect to parallel reductions making far fewer than a polynomial number of queries. A different separation technique is needed. In the proof below, we separate the boolean hierarchy by exploiting its logical structure; this allows us, via adding at most k strings to the oracle, to keep any particular $\mathrm{NP}^A(k)$ machine from accepting a certain $\mathrm{coNP}^A(k)$ language.

Proof of Theorem 3.16

A will be of the form $QBF \oplus X$; X will be constructed below. The tally set T will be an encoding of X.

We will construct a particular X relative to which the boolean hierarchy does not collapse. X will potentially have strings only at certain lengths, and will contain at most one word of each length; this will help us create the tally set T.

For any set X and for any n and i, we define the predicate $P^n_i(X) \equiv (X^{=n+i} \neq \emptyset)$. We use the following test languages for even k

$$T_k(X) = \{\, 1^n \;\Big|\; \bigwedge_{i=1}^{k/2} (P^n_{2i-1}(X) \vee \neg P^n_{2i}(X)) \,\}.$$

Clearly $T_k(X) \in \mathrm{coNP}^X(k)$. We construct the set X in stages, such that $T_k(X) \notin \mathrm{NP}^X(k)$ for all even k, thereby separating the entire boolean hierarchy.

In fact, in order to (eventually) keep small the number of truth-table queries needed to make T complete, we will use only a small segment of words of each length, and thus the strings we use will have relatively low "information content." For $m \leq n$ let

$$S^{n,m}_i = \{\, w1^{n+i-|w|} \;\Big|\; w \in \Sigma^m \,\}.$$

Note that $S_i^{n,m} \subseteq \Sigma^{n+i}$ for all $m \leq n$ and $i \geq 0$.

Let $N_1, N_2, \ldots$ be an enumeration of NP machines in which each machine is enumerated infinitely often. For each $i \geq 1$, let $p_i(n) = n^i + i$. Let $power(0) = 1$ and $power(t) = 2^{power(t-1)}$ for $t \geq 1$. Let $\langle \cdots \rangle$ be a standard multi-arity pairing function, e.g., that of [OH91] (see Section 4 for more details).

Initially X is empty and $n_0 = 0$.

At stage $s = \langle k, i \rangle$ for even k and $i = \langle i_1, \ldots, i_k \rangle$, we diagonalize against $N_{i_1}, \ldots, N_{i_k}$ (and at other stages we do nothing). We use p_s as the polynomial-time clock of these machines.

For n_s and $l(n_s)$ appropriately chosen, as will soon be described, we will put at most one word of each of the k segments $S_1^{n_s,l(n_s)}, \ldots, S_k^{n_s,l(n_s)}$ into X (and will put no other words in X), in the process freezing the queries of no more than k computation paths of the NP machines. To guarantee that there are unfrozen words in each segment, we choose n_s to be the smallest number such that

1. $n_s = power(t)$ for some t,
2. $n_s > n_{s-1}$, and
3. $2^{l(n_s)} > k\, p_s(n_s)$, where $l(n_s) = \lfloor f(p_s^{-1}(n_s))/k - 1 \rfloor$.

The choice of $l(n_s)$ will become clear below. Note that since $f(n) = \omega(\log n)$, such an n_s always exists.

Define the $\mathrm{NP}^X(k)$ set

$$L_k(X) = \bigcup_{j=1}^{k/2} \left(L(N^X_{i_{2j-1}}) - L(N^X_{i_{2j}}) \right).$$

Let $J \subseteq \{1, 3, \ldots, k-1\}$. Define $L_k(J; X)$ to be the subset of $L_k(X)$ that unions together only those pairs flagged by J.

$$L_k(J; X) = \bigcup_{j \in J} \left(L(N^X_{i_j}) - L(N^X_{i_{j+1}}) \right)$$

We show how to extend X so that $L_k(X) \neq T_k(X)$.

```
Extension of $X$.
$J := \{1, 3, \ldots, k-1\}$    (* $L_k(J;X) = L_k(X)$ *)
**repeat**    (* $1^{n_s} \in T_k(X)$ *)
```

```
    if $1^{n_s} \notin L_k(J;X)$ then exit
    else
        let $j$ be some element of $J$ satisfying $1^{n_s} \in L(N_{i_j}^X) - L(N_{i_{j+1}}^X)$.
        Freeze the queries of an accepting path of $N_{i_j}^X(1^{n_s})$ and
        put an unfrozen word from $S_{j+1}^{n_s,l(n_s)}$ into $X$.
        (* Now $P_{j+1}^{n_s}(X)$ is true, and so $1^{n_s} \notin T_k(X)$.*)
        if $1^{n_s} \notin L(N_{i_{j+1}}^X)$ then exit   (* $1^{n_s} \in L_k(X)$ *)
        else
            freeze the queries of an accepting path of $N_{i_{j+1}}^X(1^{n_s})$ and
            put an unfrozen word from $S_j^{n_s,l(n_s)}$ into $X$.
            (* Now $P_j^{n_s}(X)$ is true, and so $1^{n_s} \in T_k(X)$ yet $1^{n_s} \notin L(N_{i_j}^X) - L(N_{i_{j+1}}^X)$.
            Note that the fact that $1^{n_s} \notin L(N_{i_j}^X) - L(N_{i_{j+1}}^X)$ will be preserved
            throughout the rest of the construction. *)
        end
    end
    $J := J - \{j\}$
until $J = \emptyset$
```

Each iteration of the repeat loop either terminates (by coming to an *exit*, in which case we clearly are done) or cancels an index j from J, in which case we have have diagonalized against $L(N_{i_j}^X) - L(N_{i_{j+1}}^X)$ (while adding at most two more strings to the oracle). This can continue until $J = \emptyset$. However, if we reach the case $J = \emptyset$, then $1^{n_s} \notin L(N_{i_j}^X) - L(N_{i_{j+1}}^X)$ for all odd $j \leq k$ (and so $1^{n_s} \notin L_k(X)$), yet, by our construction, 1^{n_s} will belong to $T_k(X)$, thus successfully diagonalizing.

The above process will ensure that $\mathrm{NP}^X(k) \neq \mathrm{coNP}^X(k)$ for all $k \geq 1$. It is not hard to see that all the machines above having X as their oracle could instead have been given $QBF \oplus X$, yielding a separation relative to $A = QBF \oplus X$, rather than relative to X. Henceforth, we assume that this was done.

Define the tally set T, which we will see to be an encoding of sorts of X, by

$$T = \{\, 1^{\langle n,j,b\rangle} \,\big|\, (\exists\, w \in X^{=n})[\text{the } j\text{'th bit of } w \text{ is } b]\}.$$

We show that T is $\leq_{f(n)\text{-}tt}^{p,A}$-complete for NP^A, thus completing the proof.

Clearly $T \in \mathrm{NP}^X \subseteq \mathrm{NP}^A$.

Let $L = L(M^A) \in \mathrm{NP}^A$ and let polynomial p_t be the time bound of M. Note that on input $|x|$, the longest strings in X that might be touched during the run of machine M^A

on input x are of length at most $n_s + k$, where $n_s = n_s(x,t) = \max\{\, n_j \mid n_j \leq p_t(|x|) \,\}$. If $|x| < n_t$, we use table lookup to determine whether x belongs to L; henceforth, suppose $|x| \geq n_t$ and thus $s \geq t$.

We first show that there is a P^T machine that, on input x, finds exactly the set of strings of lengths $n_s+1, \ldots, n_s+k$ in X and uses at most $f(|x|)$ queries in parallel to T. Note that $s = \langle k, i\rangle$, n_s, and $l(n_s)$ are computable in polynomial time in $|x|$. By the construction of X, for $1 \leq i \leq k$, there is at most one word of length $n_s + i$ in X, which (if it exists) will have the form $w1^{n_s+i-|w|}$, where $|w| = l(n_s)$. These words can be constructed by asking in parallel the following set $Q = \bigcup_{i=1}^{k} Q_i$ of queries

$$Q_i = \{\, 1^{\langle n_s,1,0\rangle} \,\} \cup \{\, 1^{\langle n_s,i,1\rangle} \mid 1 \leq i \leq l(n_s) \,\}.$$

It is clear that, for all $1 \leq i \leq k$, the length $n_s + i$ strings of X can be reconstructed from the set Q_i of queries (see Theorem 3.17 for a more general reconstruction).

There are at most $f(|x|)$ words in Q:

$$\begin{aligned}
||Q|| &= (1 + l(n_s))k \\
&\leq f(p_s^{-1}(n_s)), \quad \text{since } l(n_s) = \lfloor f(p_s^{-1}(n_s))/k - 1 \rfloor \\
&\leq f(p_t^{-1}(n_s)), \quad \text{since } t \leq s, \text{ and thus } p_t \leq p_s \\
&\leq f(|x|), \quad \text{since } f \text{ is nondecreasing.}
\end{aligned}$$

So a P^A machine—that is additionally allowed $f(|x|)$ truth-table accesses to T and that uses those accesses to obtain the length n_s strings of X—can decide whether $x \in L$ as follows: Since $X^{<n_s} = X^{\leq \log n_s}$, all words in $X^{<n_s}$ can be computed by simply enumerating $\Sigma^{\leq \log n_s}$ and asking X directly (recall our machine has access to A), for each word, whether that word is in X. Now all the relevant words of X are known, and the remaining computation of $M^{QBF \oplus X}(x)$ can be solved by one query to QBF. ∎

The coding mentioned at the end of the proof is itself of independent interest. In the case of preceding proof, we ensured via our construction that X had only one word of each length, and this made it easy to recover X via few parallel accesses to an NP set. However, not all interesting sets have at most one string per length. Thus, it is natural to ask, given a set S of low density (for example, a sparse set), whether we can find its elements via *parallel* access to some NP^S set (rather than using the obvious sequential prefix-searching algorithm). The answer, perhaps surprisingly, is that parallel access suffices. Though the tools to note this have been implicit since the important sparse set research of Hartmanis, Immerman, and Sewelson [HIS85], it is noted most clearly—in slightly different form—in a recent paper of Selman ([Sel90], see that paper for a fuller discussion of the history of this notion).

The following result states that one can tighten the bound on the number of queries needed slightly beyond that found in [Sel90], and can extend the range of applicability of the technique beyond the sets in NP.

Theorem 3.17 Let S be a sparse set and let d be a polynomial-time computable function such that $d(n) = n^{\mathcal{O}(1)}$ and $(\forall n)[||S^{=n}|| \leq d(n)]$. There is an $\mathrm{FP}^{\mathrm{NP}_c^S \cap \mathrm{TALLY}}_{(1+n\binom{d(n)+1}{2})\text{-}tt}$ algorithm[7] that, on input 1^n, outputs all length n strings belonging to S.[8]

Proof of Theorem 3.17 (Following [Sel90]):
We code S as a tally set T using the encoding scheme developed by Hartmanis, Immerman, and Sewelson [HIS85]:

$$\begin{aligned} 1^{\langle n,m,i,j,b\rangle} \in T \iff & (\exists\, w_1 <_{lex} \cdots <_{lex} w_m)(\forall\, \ell : 1 \leq \ell \leq m) \\ & [w_\ell \in S^{=n} \text{ and the } j\text{'th bit of } w_i \text{ is } b]. \end{aligned}$$

We claim that, for any given n, the length n strings of S can be computed by asking the following set of queries to T.

$$Q = \{\, 1^{\langle n,1,1,1,0\rangle} \} \cup \{\, 1^{\langle n,m,i,j,1\rangle} \,\big|\, 1 \leq m \leq d(n),\ 1 \leq j \leq m,\ 1 \leq i \leq n \,\}.$$

First we look for the largest value m such that there exist i and j such that a word $1^{\langle n,m,i,j,1\rangle}$ is in T; call that value m_0. If there is such a word in T, then m_0 is the number of length n elements in S. If not, then either $1^{\langle n,1,1,1,0\rangle}$ is in T, in which case there is exactly one length n string in S, namely 0^n, or $1^{\langle n,1,1,1,0\rangle}$ is not T, in which case S contains no length n strings. Knowing the correct census, c, of $S^{=n}$, we can construct all the words of length n in S by looking at the queries in Q having $m = c$. ∎

Since a conjunctive reduction is a positive reduction, if S is in NP, then $\mathrm{NP}_c^S = \mathrm{NP}$. In this case, the above result states and generalizes Selman's result that all sparse NP sets

[7]That is, a polynomial-time machine given $1+n\binom{d(n)+1}{2}$ parallel queries to a set in NP_c^S, the class of sets that nondeterministically conjunctively reduce [LLS75] to S.

[8]Clearly, the algorithm requires no queries for the case $d(n) = 0$, and it can be seen that there are relativized worlds in which for some set S having at most one string of each length it holds that $1+n\binom{1+1}{2} = n+1$ queries are actually required. We commend to the reader the open question of whether the $1+n\binom{d(n)+1}{2}$ bound can be replaced in general by some tighter bound; we conjecture that it cannot. At issue here is the rather interesting question of the exact amount of parallel access needed to recover information about sparse sets.

are printable via parallel access to NP. Recall that a set L is P-printable [HY84] if there is a polynomial-time computable function f such that, for every n, on input 1^n the function f outputs a list of all strings in L of length at most n; relativized P-printability is defined analogously.

Corollary 3.18 If S is a sparse set, then S is $\mathrm{P}_{\mathrm{tt}}^{\mathrm{NP}_c^S \cap \mathrm{TALLY}}$-printable. In particular, all sparse sets are $\mathrm{P}_{\mathrm{tt}}^{\mathrm{TALLY}}$-printable [Rub90], and all sparse NP sets are $\mathrm{P}_{\mathrm{tt}}^{\mathrm{NP} \cap \mathrm{TALLY}}$-printable [Sel90].

4 Reductions to Simple Sparse Sets

In this section, we explore the second question mentioned in the introduction:

> If a set A reduces to a sparse set, does it follow that A is reducible to some sparse set that is "simple" relative to A?

Earlier work along these lines has been done both for the case (which is also the case of this paper) of reductions less flexible than Turing reductions [AHOW] and for the case of Turing reductions [GW91]. However, both these papers are concerned with "equivalence," and this saddles the results with weaknesses that are best illustrated by an example.

Theorem 4.1 ([AHOW]) If P = NP and set A 2-truth-table reduces to a sparse set S, then there is another sparse set $\widehat{S}$ to which A is truth-table equivalent.

The key point to notice here is that no claim is being made that A 2-truth-table reduces to $\widehat{S}$; the "equivalence" claim hides a slippage (from 2-truth-table potentially to truth-table, though in fact [AHOW] holds the slippage to 5-truth-table) of the complexity of the reduction from A. This is not a trivial point; the slippage is crucial to the structure of earlier proofs. When reducing a set A to a sparse set via a certain fixed reducing function (we speak now not of the reduction type, such as 2-truth-table, but of the actual function that generates the queries), there will in general be many possible sparse sets to which A reduces; however, for the sparse set to be consistently defined by a reduction back to A, exactly one such set must be selected. The slippage in earlier results occurs exactly because of the cost of the disambiguating down to a single sparse set.

We now show that one can obtain results that contain no slippage at all. Very informally, to do this we avoid coding too much of the disambiguating information into the sparse set. Using this type of approach, we obtain the following results about 2-truth-table and bounded disjunctive reductions. (Note that Theorems 4.3 and 4.4 are incomparable.)

Theorem 4.2 If $A \leq_{bd}^{p} S$ for some sparse set S, then there is a sparse set $\widehat{S}$ such that $A \leq_{bd}^{p} \widehat{S}$ and $\widehat{S} \in \mathrm{P}^{\mathrm{NP}^{A}[\log]}$.

Theorem 4.3 If $A \leq_{2\text{-}tt}^{p} S$ for some sparse set S, then there is a sparse set $\widehat{S}$ such that $A \leq_{2\text{-}tt}^{p} \widehat{S}$ and $\widehat{S} \in \mathrm{P}^{\mathrm{NP}^{A}[\log]}$.

Theorem 4.4 For $k = 2$ and $k = 3$: If $A \leq_{k\text{-}d}^{p} S$ for some sparse set S, then there is a sparse set $\widehat{S}$ such that $A \leq_{k\text{-}d} \widehat{S}$ and $\widehat{S} \in \mathrm{P}^{\mathrm{NP}\oplus A}$.

We also have the following results for unbounded disjunctive and unbounded conjunctive reductions.

Theorem 4.5 If $A \leq_{d}^{p} S$ for some sparse set S, then there is a sparse set $\widehat{S}$ such that $A \leq_{d}^{p} \widehat{S}$ and $\widehat{S} \in \mathrm{P}^{\mathrm{NP}^{A}}$.

Theorem 4.6 If $A \leq_{c}^{p} S$ for some sparse set S, then there is a sparse set $\widehat{S}$ such that $A \leq_{c}^{p} \widehat{S}$ and $\widehat{S} \in \mathrm{NP}^{A}$.

The rest of this section is devoted to proving the above results.

We introduce some notations and lemmas that will be helpful in proving Theorem 4.2. A collection of distinct sets $a_1, \ldots, a_h$ is called an h-sunflower if the intersection $a_i \cap a_j$ is the same for every pair of distinct indices; the common part $a_i \cap a_j$ is called the center of the sunflower. A collection W of sets is called h-compact if there are no subcollections of W that are $(h+1)$-sunflowers. The following combinatorial lemma about sunflowers, due to Erdős and Rado [ER60] (see also [BS90]), will be used extensively.

Lemma 4.7 ([ER60]) If W is an h-compact collection of sets, each of cardinality at most k, then there are at most $h^k k!$ sets in W.

We say that $A \leq_{k\text{-}d}^{p} B$ via σ if σ is a polynomial-time function such that, for all x, $||\sigma(x)|| \leq k$ and $x \in A \iff \sigma(x) \cap B \neq \emptyset$.

Let $\langle \cdot, \cdot \rangle_2$ denote a (non-onto) pairing function over finite strings with the standard nice computability, and invertibility properties, and such that $(\forall x, y)[|\langle x, y\rangle_2| = 2|x| + |y|]$. For every $k \geq 2$, let $\langle y_1, y_2, \ldots, y_k \rangle$ denote $\langle k, \langle y_1, \langle y_2, \langle \ldots, \langle y_{k-1}, y_k \rangle_2 \cdots \rangle_2 \rangle_2 \rangle_2 \rangle_2$.

The proof of Theorem 4.2 is obtained from the following technical lemma.

Lemma 4.8 Let $k \geq 1$. Suppose that $A \leq_{k\text{-}d}^{p} S$ via σ and B, D are two sets such that:

1. S is a sparse set,

2. $D \in \mathrm{NP}^{A \oplus B}$, $S \cap D = \emptyset$, and

3. there exists a polynomial-time function r such that, for all z and y, $z \in \sigma(y) \Rightarrow |y| = r(0^{|z|})$, and for some polynomial p, $(\forall n)\,[r(0^n) \leq p(n)]$.

Then there is a sparse set $\widehat{S}$ such that $A \leq^p_{k\text{-}d} \widehat{S}$ via σ, $\widehat{S} \in \mathrm{P}^{\mathrm{NP}^{A\oplus B}[\log]}$, and $\widehat{S} \cap D = \emptyset$.

Proof of Lemma 4.8:
The proof is by induction on k. If $k = 1$, define $\widehat{S} = \{z \mid (\exists y)[|y| = r(0^{|z|})$ and $y \in A$ and $z \in \sigma(y)]\}$. It is easy to verify that $\widehat{S}$ satisfies the thesis. Indeed, in this case it even holds that $\widehat{S} \in \mathrm{NP}^A$.

Let $k > 1$, and suppose that $A \leq^p_{k\text{-}d} S$ via σ and B, D are two sets that satisfy conditions (1)-(3). Since σ is computable in polynomial time and S is a sparse set, there exists a polynomial p such that, for all y and z, $z \in \sigma(y) \Rightarrow |z| \leq p(|y|)$, and, for all n, $||S^{\leq n}|| \leq p(n)$. Let q be a polynomial such that $q(n) > p(p(r(0^n)))$. The crucial fact that allows us to apply the inductive hypothesis is the following claim, which follows from the sparseness bound and is stated without proof.

Claim 1 Let n be an integer and $h = q(n)$. If $y_1, \ldots, y_h \in A$ are such that $|y_1| = \cdots = |y_h| = r(0^n)$, and the collection of sets $\sigma(y_1), \ldots, \sigma(y_h)$ is an h-sunflower whose center is c, then $c \cap S \neq \emptyset$.

Observe that, in the case described in Claim 1, the cardinality of the center c certainly is strictly less than k. For each $m = 1, \ldots, k-1$, define A_m and σ_m as follows: $A_m = \{\langle n, y_1, \ldots, y_h\rangle \mid h = q(n)$ and $y_1, \ldots, y_h \in A$ and $|y_1| = \cdots = |y_h| = r(0^n)$ and $\{\sigma(y_1), \ldots, \sigma(y_h)\}$ is an h-sunflower whose center has cardinality m $\}$, and

$$\sigma_m(y) = \begin{cases} c & \text{if } y = \langle n, y_1, \ldots, y_h\rangle \text{ and } h = q(n) \text{ and} \\ & |y_1| = \cdots = |y_h| = r(0^n) \text{ and } \{\sigma(y_1), \ldots, \sigma(y_h)\} \text{ is} \\ & \text{an } h\text{-sunflower whose center } c \text{ has cardinality } m \\ \emptyset & \text{otherwise.} \end{cases}$$

Now, we prove that for all m, $1 \leq m \leq k-1$, it holds that $A_m \leq^p_{m\text{-}d} S$ via σ_m. Take m such that $1 \leq m \leq k-1$. Suppose that $y \in A_m$. Then there are $n, y_1, \ldots, y_h$ such that $y = \langle n, y_1, \ldots, y_h\rangle$, $h = q(n)$, $y_1, \ldots, y_h \in A$, $|y_1| = \cdots = |y_h| = r(0^n)$, and $\{\sigma(y_1), \ldots, \sigma(y_h)\}$ is an h-sunflower whose center c has cardinality m; thus $\sigma_m(y) = c$, and from Claim 1 it holds that $c \cap S \neq \emptyset$, and so $\sigma_m(y) \cap S \neq \emptyset$. If $y = \langle n, y_1, \ldots, y_h\rangle \notin A_m$, then we have two cases: if there is a string y_i that does not belong to A, then $\sigma(y_i) \cap S = \emptyset$ and thus $\sigma_m(y) \cap S = \emptyset$ (since $\sigma_m(y) \subseteq \sigma(y_i)$); on the other hand, if every y_i belongs to A, then, since $y \notin A_m$, it must be the case that $\sigma_m(y) = \emptyset$.

Let $OUT = \{z \,\big|\, (\exists y)[|y| = r(0^{|z|})$ and $y \notin A$ and $z \in \sigma(y)]\}$, i.e., the set of strings that are "provably" out of S. Let $D' = D \cup OUT$. It is easy to see that, for all m, $1 \leq m \leq k-1$, it holds that A_m, S, σ_m, $A \oplus B$, and D' satisfy conditions (1)-(3) (with $A_m \rightarrow A$, $S \rightarrow S$, $\sigma_m \rightarrow \sigma$, $A \oplus B \rightarrow B$, $D' \rightarrow D$), so we can apply the inductive hypothesis, which ensures, for every $1 \leq m \leq k-1$, the existence of a sparse set S_m such that $A_m \leq^p_{m\text{-}d} S_m$ via σ_m, $S_m \in \mathrm{P}^{\mathrm{NP}^{A_m \oplus (A \oplus B)}[\log]}$, and $S_m \cap D' = \emptyset$. Define $S' = S_1 \cup \cdots \cup S_{k-1}$; since $A_m \in \mathrm{P}^A$, it holds that $S' \in \mathrm{P}^{\mathrm{NP}^{A \oplus B}[\log]}$. Furthermore, if $y \notin A$ then $\sigma(y) \cap S' = \emptyset$ (since $S' \cap D' = \emptyset$). Unfortunately, if $y \in A$ we cannot prove that $\sigma(y) \cap S' \neq \emptyset$, thus we have to add other elements to S'. Consider the collection of sets that are not "covered" by S' : $V = \{a \,\big|\, (\exists y)[y \in A$ and $\sigma(y) = a]$ and $a \cap S' = \emptyset\}$. This collection of sets can be subdivided into subcollections: $V_n = \{a \,\big|\, a \in H_n$ and $a \cap S' = \emptyset\}$, where $H_n = \{a \,\big|\, (\exists y)[y \in A$ and $|y| = r(0^n)$ and $\sigma(y) = a]\}$. Clearly $V = \bigcup_n V_n$. The collection V_n has the following important property.

Claim 2 If $a \in V_n$ then there is no collection E such that $E \subseteq H_n$, $a \in E$, and E is a $q(n)$-sunflower.

Proof of Claim 2: Let $a \in V_n$. Suppose that there exists a collection $E \subseteq H_n$ with $a \in E$, and E is a $q(n)$-sunflower. Then there exist $y_1, \ldots, y_h$ with $h = q(n)$, such that $y_1, \ldots y_h \in A$, $\sigma(y_1) = a$, and $\{\sigma(y_1), \ldots, \sigma(y_h)\} = E$. Let c be the center of E, and $m = ||c|| < k$. Thus it holds that $\langle n, y_1, \ldots, y_h\rangle \in A_m$ and $\sigma_m(\langle n, y_1, \ldots, y_h\rangle) = c$, and, since $A_m \leq^p_{m\text{-}d} S_m$, it follows that $c \cap S_m \neq \emptyset$. Furthermore, $c \subseteq \sigma(y_1) = a$, and thus $a \cap S_m \neq \emptyset$, which contradicts the assumption that $a \in V_n$.
End of Proof of Claim 2

Claim 2 suggests consideration of the following collection: $W_n = \{a \,\big|\, a \in H_n$ and there is no collection E such that $E \subseteq H_n$, $a \in E$, and E is a $q(n)$-sunflower $\}$. Define $S'' = \{z \,\big|\, (\exists a)[a \in W_{|z|}$ and $z \in a]\} - D'$, and $\widehat{S} = S' \cup S''$. From the definition of W_n it is clear that W_n is $(q(n)-1)$-compact, thus by Lemma 4.7 S'' is a sparse set. Furthermore, it is not hard to verify that $A \leq^p_{k\text{-}d} \widehat{S}$ via σ. It remains to show that $S'' \in \mathrm{P}^{\mathrm{NP}^{A \oplus B}[\log]}$. A naïve algorithm, based directly on the definitions of S'' and W_n, yields only $S'' \in \Sigma_2^{p, A \oplus B}$. In order to accomplish our goal we need the following algorithm, which, given a collection of sets T and an integer h, "approximately" checks whether or not T is h-compact. We assume a total ordering of all finite sets of strings.

```
APPROX($T$,$h$)
begin
  for each subset $c$ of some set $a \in T$ do
    $I := \emptyset$
    $m := 0$
    while $m = 0$ do
      find (if such exists) the minimum (with respect to
      the total ordering of finite sets) set $a \in T$
      such that: (1) $a \notin I$, (2) $c \subseteq a$, and (3) $(\forall b \in I)[b \cap a = c]$
      if such $a$ exists then $I := I \cup \{a\}$
      else $m := ||I||$
      end
    end (*while*)
    if $m > h$ then reject end
  end (*for*)
accept
end
```

It is not hard to see that the above algorithm has the following properties.

1. If the cardinalities of the sets of the input collection T are bounded by a constant then APPROX runs in polynomial time.
2. If $(\forall a \in T)[||a|| \leq k]$ and APPROX(T,h) accepts then T is kh-compact.
3. If APPROX(T,h) rejects then T is not h-compact.
4. If APPROX(T,h) accepts and APPROX($T \cup \{a\}$,h) rejects then there is a collection $E \subseteq T \cup \{a\}$ such that $a \in E$ and E is a $(h+1)$-sunflower.

Claim 3 If $T \subseteq H_n$ and APPROX(T,$q(n)$) accepts then APPROX($T \cup W_n$,$q(n)$) accepts.
Proof of Claim 3: By induction on the cardinality of W_n, using property (4) of APPROX.
End of proof of Claim 3

Let $m_n = \max\{||T|| \,\big|\, T \subseteq H_n$ and APPROX(T,$q(n)$) accepts$\}$. From Claim 3 we have that if $T \subseteq H_n$, APPROX(T,$q(n)$) accepts and $||T|| = m_n$ (T is maximal), then $W_n \subseteq T$. Thus, if we knew m_n and $||W_n||$, then given any string z with $|z| = n$ we could check via one query to a suitable NP^A oracle whether or not there is a set $a \in W_n$ such that $z \in a$.

In fact, an NP^A machine can guess a collection $T \subseteq H_n$ and verify that $||T|| = m_n$ and that APPROX(T,$q(n)$) accepts (observe that from property (2) of APPROX and Lemma 4.7 there is a polynomial that bounds m_n), subsequently (let $h = m_n - ||W_n||$) the NP^A machine guesses sets $a_1, \ldots, a_h$ and collections $E_1, \ldots, E_h$ such that $E_i \subseteq H_n$, $a_i \in E_i \cap T$, and $a_i \neq a_j$, and verifies that for every i it holds that E_i is a $q(n)$-sunflower; at this point, the computation accepts if and only if there is a set $a \in T - \{a_1, \ldots, a_h\}$ such that $z \in a$.

It is not hard to see that m_n and subsequently $||W_n||$ can be computed in polynomial time via $\mathcal{O}(\log n)$ queries to a suitable NP^A oracle. ▮

Proof of Theorem 4.2:
Let $A \leq^p_{k\text{-}d} S$ via σ, and let S be a sparse set. Define $S' = \{\langle 0^l, x\rangle_2 \,\big|\, x \in S \text{ and } l \geq 0\}$, and, for each x, $\sigma'(x) = \{\langle 0^{|x|p(|x|)}, y\rangle_2 \,\big|\, y \in \sigma(x)\}$, where p is a polynomial such that, for all y and z, $z \in \sigma(y) \Rightarrow |z| \leq p(|y|)$. It is easy to verify that $A \leq^p_{k\text{-}d} S'$ via σ', and that A, S', and σ' satisfy conditions (1)-(3) of Lemma 4.8 (with $B = D = \emptyset$). Thus, applying Lemma 4.8, we obtain a sparse set $\widehat{S}$ such that $A \leq^p_{k\text{-}d} \widehat{S}$ and $\widehat{S} \in \mathrm{P}^{\mathrm{NP}^A[\log]}$. ▮

We now turn to the proof of Theorem 4.3. Recall, as we discussed earlier, that when reducing a set A to a sparse set via a certain fixed reducing function, there will usually be many possible sparse sets to which A reduces. For the sparse set to be consistently defined by a reduction back to A, exactly one such set must be selected. The following proof tries to eliminate this ambiguity as follows. Via a series of binary searches for census information, the machine to accept a sparse set (via access to the original set) obtains information about it. The searches are deeply "promise"-like (see, e.g., [Sel88]) in that each round is passed a census that is too complex for it to certify correct. Finally, the machine for the sparse set has obtained census information about certain subsets of the strings in and out of itself; using this, it reduces (taking the case of Proposition 4.18 as an example) all remaining ambiguity to a feasible coloring problem. Crucially, all inputs (of the same size) will obtain the same coloring problem, and this will cause the sparse set to be consistently and correctly defined.

Proof of Theorem 4.3:
Without loss of generality, it is assumed that exactly two queries are asked in the 2-truth-table reductions. Given an input y, a 2-truth-table reduction generates a truth-table as well as a pair of queried strings; the acceptance of the input is decided by looking up the truth-table using the result of queries as an index. There are sixteen different truth-tables that can be generated; they are shown in Figures 2–7. (In the tables, entry 1 means acceptance,

and 0 means rejection.) Let $\tau(y)$ denote the truth-table generated on the input y; and, let $\mathcal{T}$ be the set of sixteen truth-tables of arity two. It is easily seen that the input string domain is completely covered by disjoint sets each of which is associated with the corresponding truth-table. For each $t \in \mathcal{T}$, let $C_t = \{y \in \Sigma^* \mid \tau(y) = t\}$. Since τ is a single-valued total function, the following proposition is clear.

Proposition 4.9 $(\forall t, t' \in \mathcal{T})[(t \neq t' \Longrightarrow C_t \cap C_{t'} = \emptyset)]$ and $\bigcup_{t \in \mathcal{T}} C_t = \Sigma^*$.

The proof of Theorem 4.3 hinges on the observation that the query string domain as well as the input string domain can be grouped into disjoint sets each of which is associated with the corresponding truth-table. Because of this, each truth-table can be handled independently in its own domain, and the final results can be combined together without causing any adverse side effects. In order to be able to discuss separately each of the truth-tables that may be used in a 2-truth-table reduction, we introduce several definitions.

Definition 4.10 Fixed truth-table reductions [Wec85] A truth-table reduction is called a fixed truth-table reduction if it uses the same truth-table for all its inputs. We denote such a reduction with $\leq_t^p$, where t denotes a truth-table: $A \leq_t^p B$ means $A \leq_{tt}^p B$ with the fixed truth-table t. We also denote such a reduction with $\leq_{ftt}^p$: i.e., $A \leq_{ftt}^p B$ *with* t means $A \leq_{tt}^p B$ with the fixed truth-table t. Similarly, $A \leq_{k\text{-}ftt}^p B$ *with* t means $A \leq_{k\text{-}tt}^p B$ with the fixed truth-table t.

Recall that the definitions of polynomial-time reductions in [LLS75] are of the form:

> $L_1 \leq_r^p L_2$ if there exists a polynomial-time computable relation R of type r such that $x \in L_1 \Longleftrightarrow R^{L_2}(x)$.

Here, we define a generalization of polynomial-time reductions.

Definition 4.11 Base-limited reductions $A : B \leq_r^p S : Q$ if there exists a polynomial-time computable relation R of type r such that
(i) if $x \in B$, then $x \in A \Longleftrightarrow R^S(x)$, and
(ii) if $x \in B$, then all strings queried in $R^S(x)$ are members of Q.
We call B and Q as the *input base* and the *query base*, respectively.

If the input base is Σ^*, the base-limited reductions are equivalent to traditional reductions except that the set of queried strings is localized within the query base.

The next proposition establishes that if the query bases are pairwise disjoint, we can find a sparse set independently in each query base of the corresponding fixed truth-table reduction, and combine the results together to form a full sparse set.

Proposition 4.12 Let $\mathcal{Q} = \{Q_t \in \Sigma^* \,|\, t \in \mathcal{T}\}$ be such that $(\forall t, t' \in \mathcal{T})[t \neq t' \implies Q_t \cap Q_{t'} = \emptyset]$. Assume that, for all $t \in \mathcal{T}$, $A : C_t \leq_t^p S_t : Q_t$ and $S_t \subseteq Q_t$. Let $\widehat{S} = \bigcup_{t \in \mathcal{T}} S_t$. Then
(i) $A \leq_{2\text{-}tt}^p \widehat{S}$, and
(ii) if S_t is sparse for all $t \in \mathcal{T}$, then $\widehat{S}$ is sparse.

Proof of Proposition 4.12: (i): On input y, the 2-tt reduction simulates the reduction of $A : C_{\tau(y)} \leq_{\tau(y)}^p S_{\tau(y)} : Q_{\tau(y)}$, but uses the oracle $\widehat{S}$ instead of $S_{\tau(y)}$. Since the queries are in $Q_{\tau(y)}$, and $Q_{\tau(y)} \cap S_{\tau(y)} = Q_{\tau(y)} \cap \widehat{S}$, $\widehat{S}$ answers each query exactly as $S_{\tau(y)}$ would. Thus, it is clear that this 2-tt reduction accepts y if and only if the corresponding base-limited reduction whose input base contains y accepts.
(ii): Clear since the union of a finite number of sparse sets is itself sparse. ∎

If S_0 is sparse and $A \leq_{2\text{-}tt}^p S_0$, there exist a sparse set S and a 2-*tt* reduction g such that $A \leq_{2\text{-}tt}^p S$ via g and the query bases are pairwise disjoint as required in Proposition 4.12. To see this, consider an input y to the original 2-*tt* reduction, and let $q_1(y)$ and $q_2(y)$ be the first queried string and the second queried string, respectively. It is easy to see that $g(y) = \{\langle 0^{|y|}, \tau(y), 0, q_1(y)\rangle, \langle 0^{|y|}, \tau(y), 1, q_2(y)\rangle\}$ is the desired reduction function with oracle $S = \{\langle 0^n, t, i, x\rangle \,|\, n \text{ is integer} \wedge t \in \mathcal{T} \wedge i \in \{0,1\} \wedge x \in S_0\}$. Clearly, g is honest and S is sparse. In the remaining part of the proof of Theorem 4.3, we assume that this reduction g is used all the time.

Note that the set of input bases and the set of query bases remain fixed by our choice of g. Armed with the "partitioning" ability of g and the "recombining" power of Proposition 4.12, we are now ready to consider individual base-limited *ftt* reductions. In each individual *ftt* case, $S_t \subseteq Q_t$ can be maintained by choosing members of S_t only from actually queried strings. In the proofs of Propositions 4.13–4.18, we choose sparse S_t while preserving the corresponding *ftt* reduction and keeping the complexity of S_t within $\mathrm{P}^{\mathrm{NP}^A[\log]}$. As a result, Proposition 4.12 will automatically establish Theorem 4.3.

Let's define some notations before diving into individual fixed truth-table reductions. Given g, it is clear that there exist polynomial-time functions σ_1, σ_2, and σ_3 that compute the first three components of a queried string x, respectively. σ_3 is used only in the proof of Proposition 4.15. Recall that C_t is the equivalence class whose elements, when reduced, generate the truth-table t. Note that membership in C_t can be checked deterministically;

this is very handy since, by using C_t as a deterministic filter, we can treat a base-limited reduction as if it were an ordinary reduction. Thus, in the following propositions, we do not use base-limited reduction notation; it will be understood that appropriate safeguards are employed when using C_t.

Proofs of the following two propositions are immediate, and thus are omitted.

Proposition 4.13 If $A \leq^p_{2\text{-}ftt} S$ via g with any of the truth-tables of Figure 2, and S is sparse, then there exists a sparse set S' in P^A such that $A \leq^p_{2\text{-}ftt} S'$ via g with the same truth-table.

Table	First Query Answered "yes"		First Query Answered "no"	
Number	2nd Ans. "yes"	2nd Ans. "no"	2nd Ans. "yes"	2nd Ans. "no"
1	0	0	0	0
2	1	1	1	1

Figure 2: Trivial truth-tables of arity two.

Proposition 4.14 If $A \leq^p_{2\text{-}ftt} S$ via g with any of the truth-tables of Figure 3, and S is sparse, then there exists a sparse set S' in NP^A such that $A \leq^p_{2\text{-}ftt} S'$ via g with the same truth-table.

Table	First Query Answered "yes"		First Query Answered "no"	
Number	2nd Ans. "yes"	2nd Ans. "no"	2nd Ans. "yes"	2nd Ans. "no"
3	0	0	1	1
4	1	1	0	0
5	0	1	0	1
6	1	0	1	0

Figure 3: 1-tt-related truth-tables of arity two.

In the proof of the next proposition, the decision to split first queries from second queries is amply repaid.

Proposition 4.15 If $A \leq^p_{2\text{-}ftt} S$ via g with any of the truth-tables of Figure 4, and S is sparse, then there exists a sparse set S' in NP^A such that $A \leq^p_{2\text{-}ftt} S'$ via g with the same truth-table.

Table	First Query Answered "yes"		First Query Answered "no"	
Number	2nd Ans. "yes"	2nd Ans. "no"	2nd Ans. "yes"	2nd Ans. "no"
7	0	0	1	0
8	1	1	0	1
9	0	1	0	0
10	1	0	1	1

Figure 4: Implication-related truth-tables of arity two.

Proof of Proposition 4.15: These tables are variations of the truth-table for implication. We prove for the case of t = *truth-table* 10. The other cases can be proved similarly.

Let $\mathit{FirstQuery}(x)$ be true if and only if $\sigma_3(x) = 0$. Let $\mathit{SecondQuery}(x)$ be true if and only if $\sigma_3(x) = 1$. Let $S_1 = \{x \mid \mathit{FirstQuery}(x) \wedge (\exists y)[y \in C_t \cap \overline{A} \wedge x \in g(y)]\}$. S_1 has the strings in S that guarantee all the strings in $\overline{A}$ to be rejected. However, some of the strings in A may also get rejected if S_1 is used instead of S in the reduction. In order to remedy this problem, we add S_2 to S', where S_2 is given by $S_2 = \{x \mid \mathit{SecondQuery}(x) \wedge (\exists x', y_1, y_2)[y_1 \in C_t \cap \overline{A} \wedge y_2 \in C_t \cap A \wedge \mathit{FirstQuery}(x') \wedge x' \in g(y_1) \wedge g(y_2) = \{x', x\}]\}$. Since S_2 contains only second queries, there is no more chaining of side effects. Note that, if the first and second queries were not separated, some strings in S_2 might be used as first queries, potentially leading to a chain of side effects. It is easy to verify that $S' = S_1 \cup S_2$ satisfies the condition. ∎

The following proposition is essentially a special case of Theorem 4.6. We omit its proof.

Proposition 4.16 If $A \leq^p_{2\text{-}ftt} S$ via g with any of the truth-tables of Figure 5, and S is sparse, then there exists a sparse set S' in NP^A such that $A \leq^p_{2\text{-}ftt} S'$ via g with the same truth-table.

Table	First Query Answered "yes"		First Query Answered "no"	
Number	2nd Ans. "yes"	2nd Ans. "no"	2nd Ans. "yes"	2nd Ans. "no"
11	1	0	0	0
12	0	1	1	1

Figure 5: Conjunctive-related truth-tables of arity two.

The following proposition is essentially a special case of Theorem 4.2. We omit its proof.

Proposition 4.17 If $A \leq^p_{2\text{-}ftt} S$ via g with any of the truth-tables of Figure 6, and S is sparse, then there exists a sparse set S' in $\mathrm{P}^{\mathrm{NP}^A[\log]}$ such that $A \leq^p_{2\text{-}ftt} S'$ via g with the same truth-table.

Table Number	First Query Answered "yes"		First Query Answered "no"	
	2nd Ans. "yes"	2nd Ans. "no"	2nd Ans. "yes"	2nd Ans. "no"
13	1	1	1	0
14	0	0	0	1

Figure 6: Disjunctive-related truth-tables of arity two.

Proposition 4.18 If $A \leq^p_{2\text{-}ftt} S$ via g with any of the truth-tables of Figure 7, and S is sparse, then there exists a sparse set S' in $\mathrm{P}^{\mathrm{NP}^A[\log]}$ such that $A \leq^p_{2\text{-}ftt} S'$ via g with the same truth-table.

Table Number	First Query Answered "yes"		First Query Answered "no"	
	2nd Ans. "yes"	2nd Ans. "no"	2nd Ans. "yes"	2nd Ans. "no"
15	0	1	1	0
16	1	0	0	1

Figure 7: Exclusive-or-related truth-tables of arity two.

Proof of Proposition 4.18: These tables are variations of the truth-table for exclusive-or. We prove for the case of $t =$ *truth-table* 15. The other case can be proved similarly.

We will show that, given a string x that satisfies $\sigma_2(x) = t$, we can determine whether $x \in S'$ for some unambiguously determined S', that is a well-behaved approximation to S. Although we cannot guarantee $S' \subseteq S$, we can make sure that S' is sparse and satisfies the requirements of the proposition.

Let $B_x = \{y \mid \sigma_1(x) = 0^{|y|} \wedge \sigma_2(x) = \tau(y)\}$, i.e., the equivalence class whose elements, when reduced, might query x. Let $g(A) = \bigcup_{y \in A} g(y)$. Since g is honest, it follows that $||g(B_x) \cap S||$ is bounded by a polynomial in $|x|$; let $s(|x|)$ denote this polynomial bound.

Let $G_1 = \{g(y) \mid y \in B_x \cap A\}$, $G_2 = \{g(y) \mid y \in B_x \cap \overline{A}\}$, and $G = G_1 \cup G_2$. The problem can be easily visualized using the graph defined by the set of edges G. G consists of two different types of edges: exclusive-or type and coexistence type. The edges in G_1 are of exclusive-or type: of the two nodes of each edge in G_1, one is in S while the other is in $\overline{S}$. The edges in G_2 are of coexistence type: the two nodes of each edge in G_2 either both are in S or both are in $\overline{S}$.

Suppose that a sparse set S' is chosen, and that all the elements in S' are given the same color while those in $\overline{S'}$ are given another color. It is easy to see that, in order to satisfy the requirement of the above proposition, it suffices to show that the two endpoints of each edge in G_1 are colored differently while those of each edge in G_2 have the same color. Thus, the problem of choosing S' can be considered as a two-coloring problem in which the sets G_1 and G_2 has to be preserved. We now describe a $\mathrm{P}^{\mathrm{NP}^A[\log]}$ algorithm to choose such as S'.

(1) Find H, the set of heavily linked elements in S each of which is characterized by too many incident edges in the graph G_1. This can be found along some computation path of an NP machine, once $||H||$ is known; note that $||H||$ can be obtained by a binary search using the NP^A oracle defined by the following machine description.

```
Machine $M_H^A$ on input $\langle 0^m, x\rangle$
    Guess $m$ distinct strings, $x_1, x_2, ..., x_m$.
    For each $x_i (i = 1, ..., m)$ guess $s(|x|) + 1$ distinct edges.
    If all the guessed edges of all the nodes $x_1, ..., x_m$ belong to the graph $G_1$ (that
    is, $x_1, x_2, ..., x_m \in H$), then accept.
```

(Similar approaches can be used in the following steps to find the various sets involved. In the remainder of this proof, we omit the details of finding such sets.)
Find $H' = H \cup \{v \mid (\exists w)[w \in H \wedge v$ and w are in the same connected component of $G_2]\}$.

(2) Find $G_L = \{e \mid e \in G_1 \wedge e \cap H = \emptyset\}$, i.e., the graph obtained from G_1 by removing all the edges incident on the nodes in H. Clearly, $||G_L|| \leq s^2(|x|)$.

(3) Find $D = \{v \mid v \in \bigcup_{e \in G_L} e \wedge$ v is in a connected component of G in which the number of the elements in $\overline{S}$ exceeds $s(|x|)\}$. (This can be done by two-coloring the connected component of G to which v belongs: as soon as the size of one color group is found to exceed $s(|x|)$, we know that the group is a subset of $\overline{S}$. While finding D, for each $v \in D$, it can be easily checked whether $v \in S$.)
Find also $D_S = D \cap S$.

Find $D'_S = D_S \cup \{v \,\big|\, (\exists w)[w \in D_S \wedge v$ and w are in the same connected component of $G_2]\}$.

(4) Find $G_s = \{e \,\big|\, (\exists e')[e' \in G_L \wedge e' \cap D = \emptyset \wedge e$ and e' are in the same connected component of $G]\}$. Clearly, $||G_s|| \leq s^2(|x|)$.

(5) Two-color G_s using a fixed polynomial-time two-coloring algorithm. Let V_s be the set of elements in one fixed color group. Put the elements of $H' \cup D'_S \cup V_s$ in S'. It is easy to see that $||H' \cup D'_S \cup V_s||$ is bounded by some polynomial.

(6) If $x \in H' \cup D'_S \cup V_s$, then accept, else reject.

It is easy to see that S', via the reduction g, preserves G_1 and G_2; this proves the correctness of the algorithm. It is not hard to fill in the details of the algorithm in a way that guarantees that $S' \in \mathrm{P}^{\mathrm{NP}^A[\log]}$. ▮

The above propositions, via Proposition 4.12, prove Theorem 4.3. ▮

Proof of Theorem 4.4:
We prove Theorem 4.4 for the case $k = 3$. Let $A \leq^p_{3\text{-}d} S$ via σ, where S is a sparse set. Without loss of generality, we assume that $(\forall x)[||\sigma(x)|| = 3]$. We need that the reduction be honest so we define $S' = \{\langle 0^l, z\rangle \,\big|\, z \in S$ and $l \geq 0\}$ and, for each x, $\sigma'(x) = \{\langle 0^{|x|}, z\rangle \,\big|\, z \in \sigma(x)\}$. It is easy to verify that S' is a sparse set and that $A \leq^p_{3\text{-}d} S'$ via σ'. Let p be a polynomial such that $(\forall x)(\forall y \in \sigma'(x))[|y| \leq p(|x|)]$ and $(\forall n)[||S'^{\leq n}|| \leq p(n)]$.

We need some notations. Define $\mathcal{R} = \{t \subseteq \Sigma^* \,\big|\, (\exists x)[\sigma'(x) = t]\}$ and $\mathcal{R}_A = \{t \subseteq \Sigma^* \,\big|\, (\exists x)[\sigma'(x) = t$ and $x \in A]\}$. Clearly, $\mathcal{R}_A \subseteq \mathcal{R}$. For all $l \geq 0$ and $z \in \Sigma^*$ define $\eta(\langle 0^l, z\rangle) = l$. It holds that $(\forall t \in \mathcal{R})(\forall y \in t)[\eta(y) \leq |y| \leq p(\eta(y))]$ and, for every x, if $\sigma'(x) = \{y_1, y_2, y_3\}$ then $\eta(y_1) = \eta(y_2) = \eta(y_3) = |x|$.

Now we introduce some concepts that are crucial for the rest of the proof. We say that a collection $\mathcal{T} \subseteq \mathcal{R}$ is a *two-flower* (*one-flower*) if there exists a set $c \subseteq \Sigma^*$ such that $||c|| = 2$ ($||c|| = 1$), for every $t, r \in \mathcal{T}$ with $r \neq t$, $t \cap r = c$, and $||\mathcal{T}|| = p(p(\eta(u))) + 1$ for $u \in c$. The set c is called the *center* of $\mathcal{T}$. A collection $\mathcal{T} \subseteq \mathcal{R}$ is said a *super-flower* if there exist $\mathcal{T}_1, \ldots, \mathcal{T}_h$ and $y \in \Sigma^*$ such that $h = p(p(\eta(y))) + 1$, $\mathcal{T} = \mathcal{T}_1 \cup \cdots \cup \mathcal{T}_h$, and $\mathcal{T}_1, \ldots, \mathcal{T}_h$ are two-flowers whose centers $c_1, \ldots, c_h$ are such that, for all distinct i, j, $c_i \cap c_j = \{y\}$. The set $\{y\}$ is called the *center* of the super-flower $\mathcal{T}$.

The basic fact about "flowers" is the following Claim, which follows from the sparseness bound and is stated without proof.

Claim 1 If $\mathcal{T}$ is either a one-flower, a two-flower, or a super-flower whose center is c, then

$$\mathcal{T} \subseteq \mathcal{R}_A \Longleftrightarrow c \cap S' \neq \emptyset.$$

The set $\widehat{S}$ will be defined as the union of three sets: $\widehat{S}_1$, $\widehat{S}_2$, and $\widehat{S}_3$. In order to define $\widehat{S}_1$ consider the following sets:

$$\begin{aligned} SF &= \{y \mid \{y\} \text{ is the center of a super-flower}\} \\ OF &= \{y \mid \{y\} \text{ is the center of a one-flower}\}. \end{aligned}$$

It is easy to see that $SF, OF \in \mathrm{NP}$. Furthermore, using prefix search and by Claim 1, it is not hard to show that $(SF \cup OF) \cap S'$ belongs to $\mathrm{P}^{\mathrm{NP} \oplus A}$. We define $\widehat{S}_1 = (SF \cup OF) \cap S'$. Thus, $\widehat{S}_1$ allows us to decide whether a string $y \in SF \cup OF$ belongs to S' or not. Now we consider the case in which $y \notin SF \cup OF$. First of all, we show that $\overline{SF \cup OF} - OUT \in \mathrm{P}^{\mathrm{NP} \oplus A}$, where $OUT = \{y \mid (\exists x)[y \in \sigma'(x) \text{ and } x \notin A]\}$ i.e., the set of strings that are "provably" out of S'. In order to do that we need the following simple fact about graph, which is stated without proof.

Claim 2 Suppose that k is any positive integer and G is a graph such that each vertex has degree at most k and G does not contain $k+1$ mutually disjoint edges. Then the number of edges of G is at most $k(2k-1)$.

Claim 3 $\overline{SF \cup OF} - OUT \in \mathrm{P}^{\mathrm{NP} \oplus A}$

Proof of Claim 3: For the sake of brevity, let $X = \overline{SF \cup OF} - OUT$. Given a string y, first we check whether $y \in \widetilde{\mathcal{R}}$, where $\widetilde{\mathcal{R}} = \{u \mid \exists t \in \mathcal{R} : u \in t\}$ (if $y \notin \widetilde{\mathcal{R}}$ then $y \in X$). Successively, we check whether $y \in \overline{SF \cup OF}$, which can be done in P^{NP}. It remains to decide whether or not $y \in OUT$, when $y \in \overline{SF \cup OF} \cap \widetilde{\mathcal{R}}$. Assume that $y \in \overline{SF \cup OF} \cap \widetilde{\mathcal{R}}$ and consider the following sets:

$$\begin{aligned} TF_y &= \{u \mid \{y, u\} \text{ is the center of a two-flower }\} \\ \mathcal{T}_y &= \{\{y, v, w\} \subseteq \Sigma^* \mid (\exists x)[\sigma'(x) = \{y, v, w\} \text{ and } v, w \notin TF_y]\} \end{aligned}$$

It is not hard to see that: $y \notin OUT$ if and only if $(\forall u \in TF_y)[\{y, u\} \cap S' \neq \emptyset]$ and $\mathcal{T}_y \subseteq \mathcal{R}_A$. Thus, it is quite easy to see that, if $||TF_y||$ and $||\mathcal{T}_y||$ were bounded by a polynomial in the length of y then we could check, by a $\mathrm{P}^{\mathrm{NP} \oplus A}$ algorithm, whether or not $y \in OUT$. Since $y \notin SF$, it holds that $||TF_y|| \leq p(p(\eta(y)))$. With regard to $\mathcal{T}_y$ consider the graph $G = (V, E)$ defined by $E = \{\{v, w\} \mid \{y, v, w\} \in \mathcal{T}_y\}$ and $V = \{u \mid (\exists a \in E)[u \in a]\}$. Clearly $||E|| = ||\mathcal{T}_y||$. Since $(\forall u \in V)[u \notin TF_y]$, each vertex of G has degree $\leq p(p(\eta(y)))$.

Furthermore, since $y \notin OF$, G does not contain $p(p(\eta(y))) + 1$ mutually disjoint edges. Thus, graph G satisfies the hypotheses of Claim 2 with $k = p(p(\eta(y)))$. It follows that $||\mathcal{T}_y|| = ||E|| \leq p(p(\eta(y)))(2p(p(\eta(y))) - 1)$.

End of proof of Claim 3

Let TFS be the set defined by

$$TFS = \{y \,\big|\, (\exists u)[\{y, u\} \text{ is the center of a two-flower } \mathcal{T} \text{ with } \mathcal{T} \subseteq \mathcal{R}_A \text{ and } u \notin \widehat{S}_1]\}.$$

We define $\widehat{S}_2 = (\overline{SF \cup OF} \cap TFS) - OUT$. According to Claim 3, for proving that $\widehat{S}_2 \in \mathrm{P}^{\mathrm{NP}\oplus A}$ it is enough to show that $\overline{SF \cup OF} \cap TFS \in \mathrm{P}^{\mathrm{NP}\oplus A}$, and this can be done in a way similar to that in the proof of Claim 3.

It remains to define $\widehat{S}_3$. Let NTF be the set defined by

$$NTF = \{y \,\big|\, (\exists x, u, v)[\sigma'(x) = \{y, u, v\}, \text{ and } u, v \notin \widehat{S}_1 \text{ and } \{u, v\} \text{ is not the center of a two-flower } \mathcal{T} \text{ with } \mathcal{T} \subseteq \mathcal{R}_A]\}.$$

We define $\widehat{S}_3 = (\overline{SF \cup OF} \cap \overline{TFS} \cap NTF) - OUT$. Using Claim 3 it is not hard to show that $\widehat{S}_3 \in \mathrm{P}^{\mathrm{NP}\oplus A}$.

At this point we can define $\widehat{S} = \widehat{S}_1 \cup \widehat{S}_2 \cup \widehat{S}_3$. We have proved that $\widehat{S} \in \mathrm{P}^{\mathrm{NP}\oplus A}$. Now we show that $A \leq^p_{3\text{-}d} \widehat{S}$ via σ'. Let x_0 be any string in A. Then $\sigma'(x_0) \cap S' \neq \emptyset$. Two cases are possible.

$\sigma'(x_0) \cap \widehat{S}_1 \neq \emptyset$: In this case $\sigma'(x_0) \cap \widehat{S} \neq \emptyset$.

$\sigma'(x_0) \cap \widehat{S}_1 = \emptyset$: Let $\sigma'(x_0) = \{y_1, y_2, y_3\}$. Without loss of generality, suppose that $y_1 \in S'$. Since $\sigma'(x_0) \cap \widehat{S}_1 = \emptyset$, $y_1 \in \overline{SF \cup OF}$. Distinguish two cases.

$y_1 \in TFS$: In this case, since $y_1 \notin OUT$, $y_1 \in \widehat{S}_2$.

$y_1 \notin TFS$: We have to distinguish other two cases.

$y_1 \in NTF$: In this case $y_1 \in \widehat{S}_3$.

$y_1 \notin NTF$: Then, since $y_2, y_3 \notin \widehat{S}_1$, it must be the case that $\{y_2, y_3\}$ is the center of a two-flower $\mathcal{T}$ with $\mathcal{T} \subseteq \mathcal{R}_A$. Thus by Claim 1, $\{y_2, y_3\} \cap S' \neq \emptyset$. Without loss of generality, suppose that $y_2 \in S'$. It holds that $y_2 \in S'$, $y_2 \in \overline{SF \cup OF}$, $y_2 \notin OUT$, and $y_2 \in TFS$. Hence $y_2 \in \widehat{S}_2$.

In any case it holds that $\sigma'(x_0) \cap \widehat{S} \neq \emptyset$.

Conversely, let x_0 be any string not in A. Then $\sigma'(x_0) \cap S' = \emptyset$, that is, $\sigma'(x_0) \subseteq OUT$. Furthermore, it is clear that $\widehat{S} \cap OUT = \emptyset$. Thus $\sigma'(x_0) \cap \widehat{S} = \emptyset$.

It remains to show that $\widehat{S}$ is a sparse set. In order to do that we prove that $\widehat{S}_1$, $\widehat{S}_2$, and $\widehat{S}_3$ are sparse sets. Clearly, $\widehat{S}_1$ is sparse since $\widehat{S}_1 \subseteq S'$. To prove that $\widehat{S}_2$ is sparse it is enough to show that TFS is sparse. For any u, let $I_u = \{y \,\big|\, \{u,y\}$ is the center of a two-flower $\mathcal{T}$ with $\mathcal{T} \subseteq \mathcal{R}_A\}$. Note that, if $|u| > p(\eta(u))$ then $I_u = \emptyset$, if $y \in I_u$ then $|y| \geq \eta(u)$, and if $u \notin SF$ then $||I_u|| \leq p(p(\eta(u)))$. Furthermore, if $y \in TFS - S'$ then $(\exists u \in S')[u \in \overline{SF \cup OF}$ and $y \in I_u]$. Thus, for any n, $TFS^{\leq n} \subseteq S'^{\leq n} \cup \bigcup_{u \in S' \cup \overline{SF \cup OF} \wedge \eta(u) \leq n} I_u$. Thus,

$$\begin{aligned} ||S'^{\leq n} \cup \bigcup_{u \in S' \cup \overline{SF \cup OF} \wedge \eta(u) \leq n} I_u|| &\leq p(n) + \sum_{u \in S' \cup \overline{SF \cup OF} \wedge \eta(u) \leq n} ||I_u|| \\ &\leq p(n) + \sum_{u \in S' \cup \overline{SF \cup OF} \wedge |u| \leq p(n)} p(p(n)) \\ &\leq p(n) + p(p(n))p(p(n)). \end{aligned}$$

Thus $||TFS^{\leq n}|| \leq p(n) + p(p(n))p(p(n))$. Hence TFS is a sparse set.

To prove that $\widehat{S}_3$ is a sparse set it is enough to show that $(\overline{TFS} \cap NTF) - OUT$ is sparse. For the sake of brevity, let $X = (\overline{TFS} \cap NTF) - OUT$. For any u, let $I_u = \{y \,\big|\, (\exists x, v)[\sigma'(x) = \{y, u, v\}$ and $\{u, v\}$ is not the center of a two-flower $]\}$. It is easy to see that if $y \in X - S'$ then $(\exists u \in S')[u \in \overline{SF \cup OF}$ and $y \in I_u]$. Thus, we have that $X \subseteq S' \cup \bigcup_{u \in S' \cap \overline{SF \cup OF}} (I_u \cap \overline{TFS})$ and so, for any n, $X^{\leq n} \subseteq S'^{\leq n} \cup \bigcup_{u \in S' \cap \overline{SF \cup OF} \wedge \eta(u) \leq n} (I_u \cap \overline{TFS})$. It follows that

$$||X^{\leq n}|| \leq p(n) + \sum_{u \in S' \cap \overline{SF \cup OF} \wedge \eta(u) \leq n} ||I_u \cap \overline{TFS}||.$$

We have to find a bound for $||I_u \cap \overline{TFS}||$ when $u \in S' \cap \overline{SF \cup OF}$. Let $I_u \cap \overline{TFS} = \{y_1, \ldots, y_h\}$ and, for all $i = 1, \ldots, h$, let v_i, x_i be such that $\sigma'(x_i) = \{y_i, u, v_i\}$ and $\{u, v_i\}$ is not the center of a two-flower. Observe that, since $y_i \in \overline{TFS}$, $u \in S'$, and $u \notin S' \cap \overline{SF \cup OF}$, it holds that $\{u, y_i\}$ is not the center of a two-flower. Consider the graph $G = (V, E)$ defined by $E = \{\{v_i, y_i\} \,\big|\, i = 1, \ldots, h\}$ and $V = \{v_1, \ldots, v_h, y_1, \ldots, y_h\}$. Clearly, $||V|| \geq h$ and $||E|| \geq h/2$. Since for all $i \in \{1, \ldots, h\}$ it holds that $\{u, y_i\}$ and $\{u, v_i\}$ are not centers of two-flowers, any vertex of G has degree at most $p(p(\eta(u)))$. Furthermore, since $u \notin OF$, G does not contain $p(p(\eta(u))) + 1$ mutually disjoint edges. Thus, G satisfies the hypotheses of Claim 2 with $k = p(p(\eta(u)))$. It follows that $||E|| \leq p(p(\eta(u)))(2p(p(\eta(u))) - 1)$. Thus, $||I_u \cap \overline{TFS}|| \leq 2||E|| \leq 2p(p(\eta(u)))(2p(p(\eta(u))) - 1)$. We conclude that, for any n, $||X^{\leq n}|| \leq p(n) + 2p(n)p(p(n))(2p(p(n)) - 1)$.

∎

We now turn to the proof of Theorem 4.5.

Proof of Theorem 4.5:
Let $A \leq_d^p S$ via σ, where S is a sparse set. We need that the reduction be honest so we define $S' = \{\langle 0^l, x\rangle \mid x \in S \text{ and } l \geq 0\}$, and, for each x, $\sigma'(x) = \{\langle 0^{|x|}, z\rangle \mid z \in \sigma(x)\}$. It is easy to verify that S' is a sparse set and that $A \leq_d^p S'$ via σ'.

We need some notations. Let $\mathcal{E}$ be a collection of sets. We denote the union of the sets of $\mathcal{E}$ by $\widetilde{\mathcal{E}}$, that is $\widetilde{\mathcal{E}} = \bigcup_{Y \in \mathcal{E}} Y$. A set H is called a *hitting set* for $\mathcal{E}$ if $H \subseteq \widetilde{\mathcal{E}}$ and, for all $Y \in \mathcal{E}$, $Y \cap H \neq \emptyset$. Our problem is to find a "simple" sparse hitting set $\widehat{S}$ for the collection $\mathcal{G} = \{Z \mid (\exists y)[y \in A \text{ and } \sigma'(y) = Z]\}$ with $\widehat{S} \cap OUT = \emptyset$, where $OUT = \{z \mid (\exists y)[y \notin A \text{ and } z \in \sigma'(y)]\}$. The collection $\mathcal{G}$ can be subdivided into finite subcollections: $\mathcal{G}_n = \{Z \mid (\exists y)[y \in A \text{ and } |y| = n \text{ and } \sigma'(y) = Z]\}$. Thus, it is sufficient to find, for each n, a hitting set H_n for $\mathcal{G}_n$ with, at most, a polynomial number of elements, and such that $H_n \cap OUT = \emptyset$. In fact the following holds.

Claim 1 Suppose that:

1. for all n, H_n is a hitting set for $\mathcal{G}_n$ with $H_n \cap OUT = \emptyset$, and
2. there exists a polynomial q such that, for all n, $||H_n|| \leq q(n)$.

Then the set $\widehat{S} = \bigcup_n H_n$ is such that $A \leq_d^p \widehat{S}$ via σ' and $\widehat{S}$ is sparse.

Proof of Claim 1: Firstly, we prove that $A \leq_d^p \widehat{S}$ via σ'. If $y \in A$ then $\sigma'(y) \in \mathcal{G}_{|y|}$ and, since $H_{|y|}$ is a hitting set for $\mathcal{G}_{|y|}$, it holds that $\sigma'(y) \cap H_{|y|} \neq \emptyset$, and thus $\sigma'(y) \cap \widehat{S} \neq \emptyset$. If $y \notin A$ then $\sigma'(y) \subseteq OUT$ and so $\sigma'(y) \cap \widehat{S} = \emptyset$. Now, we prove that $\widehat{S}$ is sparse. Let r be a polynomial such that, for all y and z, $z \in \sigma'(y) \Longrightarrow |y| \leq r(|z|)$. If $z \in \widehat{S}$ then there is a k such that $z \in H_k \subseteq \mathcal{G}_k$, for which there is a string y such that $z \in \sigma'(y)$ and $|y| = k$, and thus $k \leq r(|z|)$. This proves that $\widehat{S}^{\leq n} \subseteq \bigcup_{1 \leq k \leq r(n)} H_k$ and, since $||H_k|| \leq q(k)$, it holds that $||\widehat{S}^{\leq n}|| \leq r(n)q(n)$.
End of Proof of Claim 1

Observe that, for all z, $z \in \widehat{S} \Longleftrightarrow z \in \bigcup_{1 \leq k \leq r(|z|)} H_k$. From this observation and Claim 1 it follows that if there exists a $\mathrm{P}^{\mathrm{NP}^A}$ algorithm that, given n, produces a set H_n that satisfies conditions (1)-(2) of Claim 1, then the set $\widehat{S} = \bigcup_n H_n$ satisfies the thesis.

Consider the following algorithm.

```
HITTING($n$)
**begin**
  $H_n := \emptyset$
  **while** there is an $x \in A^{=n}$ such that $\sigma'(x) \cap H_n = \emptyset$ **do**
```

```
        by prefix search construct such an $x$
        $H_n := H_n \cup (\sigma'(x) - OUT)$
      end
    end
```

Clearly after the above algorithm terminates, H_n is a hitting set for $\mathcal{G}_n$ with $H_n \cap OUT = \emptyset$. It is also clear that at each iteration at least a new string in $S' \cap \widetilde{\mathcal{G}_n}$ is added to H_n. This and the fact that S' is a sparse set imply that the number of iterations of the above algorithm is bounded by a polynomial. Thus, the cardinality of H_n is also bounded by a polynomial. Furthermore, the test in the while loop can be done in NP^A and the prefix search can be done in $\mathrm{P}^{\mathrm{NP}^A}$. Thus, HITTING is a $\mathrm{P}^{\mathrm{NP}^A}$ algorithm that, given n, produces a set H_n that satisfies conditions (1)-(2) of Claim 1. ▮

Finally, we turn to the proof of Theorem 4.6.

Proof of Theorem 4.6:
Assume $A \leq_c^p S$ via g. Without loss of generality, assume that g is honest. Let $\widehat{S} = \{x \mid (\exists y)[y \in A \wedge x \in g(y)]\}$. Clearly, $\widehat{S} \in \mathrm{NP}^A$, and $A \leq_c^p \widehat{S}$ via g. ▮

5 Low Instance Complexity and Polynomial-Time Reductions

Instance complexity, as defined by Ko, Orponen, Schöning, and Watanabe [KOSW86, Orp90,OKSW], is a notion of the complexity of specific instances of a problem—a topic that standard complexity theory is ill-suited to study (as any single instance is trivial). In this paper, we are primarily concerned with sets of "low" instance complexity: IC[log,poly] (introduced in [KOSW86]).

Definition 5.1 We say that a set A is in IC[log,poly] if there exist a constant $c > 0$, a polynomial t and a set $\Pi \subseteq \Sigma^*$ of programs[9] such that for every $x \in \Sigma^*$

1. there exists a $p \in \Pi^{=c\log(|x|)}$ such that p decides x in time $t(|x|)$ according to A, and
2. for every $p \in \Pi$ it holds that if p decides x in time $t(|x|)$ then p decides x according to A.

[9]For a fixed efficient universal machine. In fact, "p decides x in time $\cdots$" in this definition refers to the run of the fixed universal machine on $\langle p, x\rangle$. We refer the reader to [Orp90, p. 21] for details of the universal machine scheme.

We show that the sets of low instance complexity are intimately related to the study of reductions to sets of low information content. In particular, a set A is of low instance complexity if and only if it both conjunctively and disjunctively reduces to tally sets (say T_0 and T_1, respectively). Note that this implies that A reduces disjunctively and conjunctively to the single tally set: $\{0^{2i} \mid 0^i \in T_0\} \cup \{0^{2i+1} \mid 0^i \in T_1\}$.

Theorem 5.2 A is in IC[log,poly] if and only if there exist tally sets T_0 and T_1 such that $A \leq_c^p T_0$ and $A \leq_d^p T_1$.

Proof of Theorem 5.2:
For $A \in$ IC[log,poly], let c be a constant, t be a polynomial and Π be a set of programs as in Definition 5.1. Recall that we denote by $ord(p)$ the position of the string p in the lexicographical enumeration of Σ^*. We can encode Π into the tally set $T = \{0^{ord(p)} \mid p \in \Pi\}$. Let g be a truth-table condition generator that on input x computes the disjunction of the encodings $0^{ord(p)}$ for all programs p of length $c\log(|x|)$ that accept x in time $t(|x|)$. Then $A \leq_d^p T$ via g since x is in A if and only if there exists a program in $\Pi^{=c\log(|x|)}$ that accepts x in time $t(|x|)$.

Similarly, consider the truth-table generator h that on input x computes a conjunction of the encodings $0^{ord(p)}$ for all programs p of length $c\log(|x|)$ that reject x in time $t(|x|)$. Then $A \leq_c^p (0^* - T)$ via h since x is in A if and only if no program rejecting x in time $t(|x|)$ is in $\Pi^{=c\log(|x|)}$.

For the reverse inclusion, let $A \leq_c^p C$ and $A \leq_d^p D$ where C and D are tally sets, and g_c and g_d are the respective polynomial-time truth-table condition generators. We assume that all generated queries are in 0^*. For every $0^i \notin O^*$ it is easy to construct a program p_i^c that on input x computes $g_c(x) = y_1 \wedge \ldots \wedge y_m$ and rejects if $0^i \in \{y_j \mid 1 \leq j \leq m\}$. Otherwise p_i^c goes into an infinite loop. It is clear that the running time of p_i^c is polynomially bounded on all inputs that it rejects, and that the size of p_i^c is $\mathcal{O}(\log(i))$. For every $x \notin A$ the conjunction $g_c(x)$ contains a query $0^j \notin C$. Thus for every $x \notin A$ there exists an index j, polynomially bounded in $|x|$, such that $0^j \notin C$ and p_j^c on input x rejects.

Similarly, for every $0^i \in 0^*$ there is a program p_i^d that on input x computes $g_d(x) = z_1 \vee \ldots \vee z_m$ and accepts if $0^i \in \{z_j \mid 1 \leq j \leq m\}$. Otherwise it goes into an infinite loop. The programs p_i^d are also $\mathcal{O}(\log(i))$ in size and have polynomial running time on all inputs that are accepted. For every $x \in A$ the disjunction $g_d(x)$ contains a query $0^j \in D$. Thus for every $x \in A$ there exists an index j, polynomially bounded in $|x|$, such that $0^j \in D$ and p_j^d on input x accepts.

Hence, taking $\Pi = \{p^d_j \mid 0^j \in D\} \cup \{p^c_j \mid 0^j \notin C\}$ as the set of programs, it follows that $A \in$ IC[log,poly]. ∎

Ko [Ko89] showed that SPARSE $\not\subseteq$ R^p_d(TALLY). From this result and Theorem 5.2, it follows that there exist sparse sets that don't have low instance complexity, a result that is a corollary to the proof of the result of [OKSW] that P/lin $\not\subseteq$ IC[log,poly].

Corollary 5.3 SPARSE $\not\subseteq$ IC[log,poly].

Using the above characterization of IC[log,poly] it is easy to see that IC[log,poly] is closed under bounded truth-table reductions (a different proof that explicitly constructs programs appears in [OKSW]):

Theorem 5.4 [OKSW] IC[log,poly] is closed downward under $\leq^p_b$-reductions.

Proof of Theorem 5.4:
It is easy to see that coR^p_d(TALLY) = R^p_c(TALLY), which immediately implies the closure of R^p_d(TALLY)$\cap$$\mathrm{R}^p_c$(TALLY) under complementation. In fact, for every set A in R^p_d(TALLY)$\cap$ R^p_c(TALLY) there exists a single tally set T such that A and $\overline{A}$ are in $\mathrm{R}^p_d(T) \cap \mathrm{R}^p_c(T)$ and thus R^p_d(TALLY) $\cap$ R^p_c(TALLY) is also closed under one truth-table reductions.

The second observation is that $\mathrm{R}^p_{bc}(\mathrm{R}^p_d(\mathrm{TALLY})) \subseteq \mathrm{R}^p_d(\mathrm{R}^p_{bc}(\mathrm{TALLY})) \subseteq \mathrm{R}^p_d(\mathrm{R}^p_m(\mathrm{TALLY})) \subseteq \mathrm{R}^p_d(\mathrm{TALLY})$. Here we use that $\mathrm{R}^p_{bc}(\mathrm{TALLY}) = \mathrm{R}^p_m(\mathrm{TALLY})$ (see [Ko89]).

Similarly, $\mathrm{R}^p_{bd}(\mathrm{R}^p_c(\mathrm{TALLY})) \subseteq \mathrm{R}^p_c(\mathrm{TALLY})$, and thus it follows that R^p_d(TALLY) $\cap$ R^p_c(TALLY) is closed under bounded conjunctive and bounded disjunctive reducibilities. Combining this, we get

$$\begin{aligned}
\mathrm{R}^p_b(\mathrm{IC[log,poly]}) &= \mathrm{R}^p_b(\mathrm{R}^p_d(\mathrm{TALLY}) \cap \mathrm{R}^p_c(\mathrm{TALLY})) \\
&\subseteq \mathrm{R}^p_{bc}(\mathrm{R}^p_{bd}(\mathrm{R}^p_{1-tt}(\mathrm{R}^p_d(\mathrm{TALLY}) \cap \mathrm{R}^p_c(\mathrm{TALLY})))) \\
&\subseteq \mathrm{R}^p_d(\mathrm{TALLY}) \cap \mathrm{R}^p_c(\mathrm{TALLY}) = \mathrm{IC[log,poly]}.
\end{aligned}$$

∎

Corollary 5.5 [OKSW] If P $\neq$ NP and A is $\leq^p_b$-hard for NP, then $A \notin$ IC[log, poly].

Proof of Corollary 5.5:
Suppose that a bounded truth-table hard set for NP is in IC[log,poly]. It follows by Theorem 5.4 that NP $\subseteq$ IC[log,poly]. From the result of [KOSW86] that if a set of low instance complexity is many-one hard for NP then P = NP, it follows that P = NP. ∎

Using Theorem 5.2 and the fact that NP has neither $\leq_c^p$-hard tally sets nor $\leq_d^p$-hard tally sets unless P = NP, it follows that NP has neither $\leq_c^p$-hard sets nor $\leq_d^p$-hard sets in IC[log,poly] unless P = NP.

Corollary 5.6 If P $\neq$ NP and A is $\leq_d^p$-hard for NP, then $A \notin \mathrm{IC}[\log, \mathrm{poly}]$.

Proof of Corollary 5.6:
Suppose that a disjunctive truth-table hard set for NP is in IC[log,poly]. Since $\mathrm{R}_d^p(\mathrm{IC[log,poly]}) \subseteq \mathrm{R}_d^p(\mathrm{TALLY}) \subseteq \mathrm{R}_d^p(\mathrm{coSPARSE})$, it follows from Ukkonen's result [Ukk83] that P = NP. ∎

Corollary 5.7 If P $\neq$ NP and A is $\leq_c^p$-hard for NP, then $A \notin \mathrm{IC}[\log, \mathrm{poly}]$.

Proof of Corollary 5.7:
Suppose that a conjunctive truth-table hard set for NP is in IC[log,poly]. Since $\mathrm{R}_c^p(\mathrm{IC[log,poly]}) \subseteq \mathrm{R}_c^p(\mathrm{SPARSE})$, it follows from Corollary 3.4 that P = NP. ∎

Finally, using Theorem 3.16 and the fact that every tally set is in IC[log, poly], we can conclude that for truth-tables of size $\omega(\log n)$, no analog of Corollary 5.5 can be proven by any relativizable proof technique.

Acknowledgments

For helpful conversations, comments, suggestions, and literature pointers, we are grateful to E. Allender, L. Fortnow, W. Gasarch, A. Hoene, R. Rubinstein, R. Schuler, A. Selman, E. Spaan, J. Torán, S. Wahl, and O. Watanabe. We thank J. Balcázar, M. Hermo, and R. Schuler for discussions from which Theorem 5.2 resulted.

References

[ABG90] A. Amir, R. Beigel, and W. Gasarch. Some connections between bounded query classes and non-uniform complexity. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 232–243. IEEE Computer Society Press, July 1990.

[AHH$^+$] 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 *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming.* Springer-Verlag *Lecture Notes in Computer Science.* To appear.

[AHH$^+$91] 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. Ulmer Informatik-Bericht 91-08, Fakultät für Informatik, Universität Ulm, Ulm, Germany, 1991.

[AHOW] E. Allender, L. Hemachandra, M. Ogiwara, and O. Watanabe. Relating equivalence and reducibility to sparse sets. *SIAM Journal on Computing.* To appear. Preliminary version appears as [AHOW91].

[AHOW91] E. Allender, L. Hemachandra, M. Ogiwara, and O. Watanabe. Relating equivalence and reducibility to sparse sets. In *Proceedings of the 6th Structure in Complexity Theory Conference*, pages 220–229. IEEE Computer Society Press, June/July 1991.

[AKM] V. Arvind, J. Köbler, and M. Mundhenk. Bounded truth-table and conjunctive reductions to sparse and tally sets. Ulmer Informatik-Bericht 92-01, Fakultät für Informatik, Universität Ulm, Ulm, Germany, 1992.

[Bal90] J. Balcázar. Self-reducibility. *Journal of Computer and System Sciences*, 41(3):367–388, 1990.

[Ber78] P. Berman. Relationship between density and deterministic complexity of NP-complete languages. In *Proceedings of the 5th International Colloquium on Automata, Languages, and Programming*, pages 63–71. Springer-Verlag *Lecture Notes in Computer Science #62*, 1978.

[BGH90] R. Beigel, J. Gill, and U. Hertrampf. Counting classes: Thresholds, parity, mods, and fewness. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science*, pages 49–57. Springer-Verlag *Lecture Notes in Computer Science #415*, February 1990.

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[BK88] R. Book and K. Ko. On sets truth-table reducible to sparse sets. *SIAM Journal on Computing*, 17(5):903–919, 1988.

[BLS92] H. Buhrman, L. Longpré, and E. Spaan, 1992. Personal Communication.

[BS90] R. Boppana and M. Sipser. The complexity of finite functions. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 14, pages 757–804. MIT Press/Elsevier, 1990.

[CGH$^+$88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, 1988.

[CGH$^+$89] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy II: Applications. *SIAM Journal on Computing*, 18(1):95–111, 1989.

[CH90] J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, 23(2):95–106, 1990.

[CH91] J. Cai and L. Hemachandra. A note on enumerative counting. *Information Processing Letters*, 38(4):215–219, 1991.

[ER60] P. Erdős and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 35:85–90, 1960.

[For79] S. Fortune. A note on sparse complete sets. *SIAM Journal on Computing*, 8(3):431–433, 1979.

[Gav92] R. Gavaldà. On conjunctive and disjunctive reductions to sparse sets. Manuscript, January 1992.

[GW91] R. Gavaldà and O. Watanabe. On the computational complexity of small descriptions. In *Proceedings of the 6th Structure in Complexity Theory Conference*, pages 89–101. IEEE Computer Society Press, June/July 1991.

[Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 439–445. IEEE Computer Society Press, 1983.

[Hau14] F. Hausdorff. *Grundzüge der Mengenlehre.* Leipzig, 1914.

[Hel86] H. Heller. On relativized exponential and probabilistic complexity classes. *Information and Control*, 71:231–243, 1986.

[HH91] L. Hemachandra and A. Hoene. On sets with efficient implicit membership tests. *SIAM Journal on Computing*, 20(6):1148–1156, 1991.

[HIS85] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP-P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):159–181, 1985.

[HL91] S. Homer and L. Longpré. On reductions of NP sets to sparse sets. In *Proceedings of the 6th Structure in Complexity Theory Conference*, pages 79–88. IEEE Computer Society Press, June/July 1991.

[HOW] L. Hemachandra, M. Ogiwara, and O. Watanabe. How hard are sparse sets? In preparation (will appear in the Proceedings of the 7th Structure in Complexity Theory Conference).

[HY84] J. Hartmanis and Y. Yesha. Computation times of NP sets of different densities. *Theoretical Computer Science*, 34:17–32, 1984.

[IM89] N. Immerman and S. Mahaney. Relativizing relativized computations. *Theoretical Computer Science*, 68:267–276, 1989.

[JY85] D. Joseph and P. Young. Some remarks on witness functions for non-polynomial and non-complete sets in NP. *Theoretical Computer Science*, 39:225–237, 1985.

[JY90] D. Joseph and P. Young. Self-reducibility: Effects of internal structure on computational complexity. In A. Selman, editor, *Complexity Theory Retrospective*, pages 82–107. Springer-Verlag, 1990.

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

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th ACM Symposium on Theory of Computing*, pages 302–309, April 1980.

[KMR89] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails relative to a random oracle. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 157–166. ACM Press, May 1989.

[Ko89] K. Ko. Distinguishing conjunctive and disjunctive reducibilities by sparse sets. *Information and Computation*, 81(1):62–87, 1989.

[KOSW86] K. Ko, P. Orponen, U. Schöning, and O. Watanabe. What is a hard instance of a computational problem? In *Proceedings of the 1st Structure in Complexity Theory Conference*, pages 197–217. Springer-Verlag *Lecture Notes in Computer Science #223*, June 1986.

[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.

[Mah89] S. Mahaney. The isomorphism conjecture and sparse sets. In J. Hartmanis, editor, *Computational Complexity Theory*, pages 18–46. American Mathematical Society, 1989. Proceedings of Symposia in Applied Mathematics #38.

[OH91] M. Ogiwara and L. Hemachandra. A complexity theory for feasible closure properties. In *Proceedings of the 6th Structure in Complexity Theory Conference*, pages 16–29. IEEE Computer Society Press, June/July 1991. To appear in *Journal of Computer and System Sciences*.

[OKSW] P. Orponen, K. Ko, U. Schöning, and O. Watanabe. Instance complexity. *Journal of the ACM*. To appear.

[OL] M. Ogiwara and A. Lozano. On one-query self-reducible sets. *Theoretical Computer Science*. To appear. Preliminary version appears as [OL91].

[OL91] M. Ogiwara and A. Lozano. On one-query self-reducible sets. In *Proceedings of the 6th Structure in Complexity Theory Conference*, pages 139–151. IEEE Computer Society Press, June/July 1991.

[Orp90] P. Orponen. On the instance complexity of NP-hard problems. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 20–27. IEEE Computer Society Press, July 1990.

[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.

[RR] D. Ranjan and P. Rohatgi. On randomized reductions to sparse sets. In *Proceedings of the 7th Structure in Complexity Theory Conference*. IEEE Computer Society Press. To appear.

[Rub90] R. Rubinstein. Relativizations of the P-printable sets and the sets with small generalized Kolmogorov complexity. Technical Report WPI-CS-TR-90-3, Worcester Polytechnic Institute, Worcester, MA, March 1990.

[Sel88] A. Selman. Promise problems complete for complexity classes. *Information and Computation*, 78:87–98, 1988.

[Sel90] A. Selman. A note on adaptive vs. nonadaptive reductions to NP. Technical Report 90-20, State University of New York at Buffalo Department of Computer Science, Buffalo, NY, September 1990.

[Ukk83] E. Ukkonen. Two results on polynomial time truth-table reductions to sparse sets. *SIAM Journal on Computing*, 12(3):580–587, 1983.

[Wag90] K. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

[Wat88] O. Watanabe. On $\leq^p_{1-tt}$ sparseness and nondeterministic complexity classes. In *Proceedings of the 15th International Colloquium on Automata, Languages, and Programming*, pages 697–709. Springer-Verlag *Lecture Notes in Computer Science #317*, July 1988.

[Wat91] O. Watanabe. On intractability of the class UP. *Mathematical Systems Theory*, 24:1–10, 1991.

[Wec85] G. Wechsung. On the boolean closure of NP. In *Proceedings of the 5th Conference on Fundamentals of Computation Theory*, pages 485–493. Springer-Verlag *Lecture Notes in Computer Science #199* , 1985. (An unpublished precursor of this paper was coauthored by K. Wagner).

[Yap83] C. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983.

[Yes83] Y. Yesha. On certain polynomial-time truth-table reducibilities of complete sets to sparse sets. *SIAM Journal on Computing*, 12(3):411–425, 1983.

A A Direct Proof for the Conjunctive Cases

In this section, we give a simple direct proof of Theorem 3.3.

Theorem 3.3 If $A \in \mathrm{NP}$ and $\mathit{Left}(A) \in \mathrm{R}^p_c(\mathrm{SPARSE})$, then A is in P.

Proof of Theorem 3.3:

Let $q(\cdot)$ be a polynomial and let P_A be a polynomial-time set such that $A = \{x \mid (\exists w \in \Sigma^{q(|x|)})[\langle x, w\rangle \in P_A]\}$ and $\mathit{Left}(A) = \{\langle x, w\rangle \mid x \in A \;\wedge\; w \in \Sigma^{q(|x|)} \;\wedge\; w \leq w_{max}\}$ where $w_{max} = \max\{w \in \Sigma^{q(|x|)} \mid \langle x, w\rangle \in P_A\}$. Let S be the sparse set such that $\mathit{Left}(A) \in \mathrm{R}^p_c(S)$ and let s be a polynomial such that $||S^{\leq n}|| \leq s(n)$ for every n. In order to do a breadth-first search on the tree of possible witness prefixes we use the set $\mathit{prefix}(\mathit{Left}(A)) = \{\langle x, y\rangle \mid (\exists z)[\langle x, yz\rangle \in \mathit{Left}(A)]\}$. It is easy to see that $\mathit{prefix}(\mathit{Left}(A))$ is many-one equivalent to $\mathit{Left}(A)$, and so it follows that $\mathit{prefix}(\mathit{Left}(A)) \in \mathrm{R}^p_c(S)$. The corresponding truth-table condition generator can be expressed as

$$g(\langle x, y\rangle) = z_1 \wedge \ldots \wedge z_m,$$

where $z_1, \ldots, z_m$ are conjunctive queries to S and $m \leq r(|x|)$ for some polynomial r. Let $Q(y)$ denote the set $\{z_1, \ldots, z_m\}$ and, for a set M of prefixes, let $Q(M)$ denote $\bigcup_{y \in M} Q(y)$. Note that $\langle x, y\rangle \in \mathit{prefix}(\mathit{Left}(A))$ if and only if $Q(y) \subseteq S$. Let $\tilde{m}$ be the length of the longest string in any of the $Q(y)$, $0 \leq |y| \leq q(|x|)$. Clearly $\tilde{m}$ is bounded by a polynomial in $|x|$. What follows is the polynomial-time algorithm for A.

```
input $x$
begin
$N := \{\epsilon\}$
for $l := 1$ to $q(|x|)$ do
  $N := \{y0 \mid y \in N\} \cup \{y1 \mid y \in N\}$  (* expand prefixes to length $l$ *)
  (* let $y_1, \ldots, y_t$ be the prefixes in $N$ in lexicographical order *)
  $M := \{y_1\}$
  $i := 1$
  repeat
    $i := i + 1$
    if $Q(y_i) \not\subseteq Q(M)$ then $M := M \cup \{y_i\}$ end  (* $Q(M) = Q(\{y_1, \ldots, y_i\})$ *)
  until $||Q(M)|| > s(\tilde{m})$ or $i = t$ (* $||M|| \leq s(\tilde{m}) + 1$ *)
  $N := \{y_{i-1} \mid y_i \in M\} \cup \{y_t\}$
```

```
    **if** there is a witness in $N$ **then** *accept* **else** *reject* **end**
    **end**
```

It is immediate that the above algorithm runs in polynomial time since N never has more than $2(s(\tilde{m})+1)$ many elements.

To prove the correctness of the algorithm, assume that x is in A (if $x \notin A$ there is no witness and x is certainly rejected). We will show inductively that N always contains a prefix of w_{max}. When the for loop is entered, $N = \{\epsilon\}$ and thus certainly contains a prefix of w_{max}. Also, if there is a prefix of w_{max} in N before the expansion of the prefixes then there is one afterwards. Thus let y_h be the prefix of w_{max} in $N = \{y_1, \ldots, y_t\}$ after the expansion of its members. We will show that y_h is included in N after the repeat loop. Since y_t is always included in N we can assume that $h < t$. Then $\langle x, y_{h+1}\rangle \notin prefix(Left(A))$ and for all $j \leq h$, $\langle x, y_j\rangle \in prefix(Left(A))$. Thus $Q(y_{h+1}) \not\subseteq S$ and for all $j \leq h$, $Q(y_j) \subseteq S$, i.e., $Q(y_{h+1}) \not\subseteq Q(\{y_1, \ldots, y_h\})$ and $||Q(\{y_1, \ldots, y_h\})|| \leq s(\tilde{m})$. This shows that y_{h+1} is included in M and thus y_h remains in N after the repeat loop. ∎