

## PROPERTIES OF NP-COMPLETE SETS

CHRISTIAN GLABER<sup>†</sup>, A. PAVAN<sup>‡</sup>, ALAN L. SELMAN<sup>§</sup>, AND SAMIK SENGUPTA<sup>¶</sup>

**Abstract.** We study several properties of sets that are complete for NP. We prove that if  $L$  is an NP-complete set and  $S \not\geq L$  is a p-selective sparse set, then  $L - S$  is  $\leq_m^p$ -hard for NP. We demonstrate existence of a sparse set  $S \in \text{DTIME}(2^{2^n})$  such that for every  $L \in \text{NP} - \text{P}$ ,  $L - S$  is not  $\leq_m^p$ -hard for NP. Moreover, we prove for every  $L \in \text{NP} - \text{P}$ , that there exists a sparse  $S \in \text{EXP}$  such that  $L - S$  is not  $\leq_m^p$ -hard for NP. Hence, removing sparse information in P from a complete set leaves the set complete, while removing sparse information in EXP from a complete set may destroy its completeness. Previously, these properties were known only for exponential time complexity classes.

We use hypotheses about pseudorandom generators and secure one-way permutations to derive consequences for longstanding open questions about whether NP-complete sets are immune. For example, assuming that pseudorandom generators and secure one-way permutations exist, it follows easily that NP-complete sets are not p-immune. Assuming only that secure one-way permutations exist, we prove that no NP-complete set is  $\text{DTIME}(2^{n^\epsilon})$ -immune. Also, using these hypotheses we show that no NP-complete set is quasipolynomial-close to P.

We introduce a strong but reasonable hypothesis and infer from it that disjoint Turing-complete sets for NP are not closed under union. Our hypothesis asserts existence of a UP-machine  $M$  that accepts  $0^*$  such that for some  $0 < \epsilon < 1$ , no  $2^{n^\epsilon}$  time-bounded machine can correctly compute infinitely many accepting computations of  $M$ . We show that if  $\text{UP} \cap \text{coUP}$  contains  $\text{DTIME}(2^{n^\epsilon})$ -bi-immune sets, then this hypothesis is true.

**Key words.** NP-completeness, Robustness, Immunity, One-way permutations, Disjoint unions.

**AMS subject classifications.** 68Q15

**1. Introduction.** This paper continues the long tradition of investigating the structure of complete sets under various kinds of reductions. Concerning the most interesting complexity class, NP, almost every question has remained open. While researchers have always been interested primarily in the structure of complete sets for NP, for the most part, success, where there has been success, has come from studying the exponential time classes. In this paper we focus entirely on the complexity class NP.

The first topic we study concerns the question of how robust are complete sets. Schöning [Sch86] raised the following question: If a small amount of information is removed from a complete set, does the set remain hard? Tang, Fu, and Liu [TFL93] proved existence of a sparse set  $S$  such that for every  $\leq_m^p$ -complete set  $L$  for EXP,  $L - S$  is not hard. Their proof depends on the fact that for any exponential time computable set  $B$  and any exponential time complete set  $A$ , there exists a length-increasing, one-one reduction from  $B$  to  $A$  [Ber76]. We do not know that about NP. Buhrman, Hoene, and Torenvliet [BHT98] proved that  $L - S$  still remains hard for EXP, if  $S$  is any p-selective sparse set.

Here, we prove these results unconditionally for sets that are NP-complete. We prove that if  $L$  is an NP-complete set and  $S \not\geq L$  is a p-selective sparse set, then  $L - S$  is  $\leq_m^p$ -hard for NP. We use the left-set technique of Ogihara and Watanabe [OW91] to prove this result, and we use this technique elsewhere in the paper also. We demonstrate existence of a sparse set  $S \in \text{DTIME}(2^{2^n})$  such that for every  $L \in \text{NP} - \text{P}$ ,  $L - S$  is not  $\leq_m^p$ -hard for NP. Moreover, we prove for every  $L \in \text{NP} - \text{P}$ , that there exists a sparse  $S \in \text{EXP}$  such that

---

<sup>†</sup>Lehrstuhl für Informatik IV, Universität Würzburg, 97074 Würzburg, Germany. Email: glasser@informatik.uni-wuerzburg.de.

<sup>‡</sup>Department of Computer Science, Iowa State University, Ames, IA 50011. Research supported in part by NSF grants CCR-0344187 and CCF-0430807. Email: pavan@cs.iastate.edu.

<sup>§</sup>Department of Computer Science and Engineering, 201 Bell Hall, University at Buffalo, Buffalo, NY 14260. Research partially supported by NSF grant CCR-0307077. Email: selman@cse.buffalo.edu.

<sup>¶</sup>Department of Computer Science and Engineering, 201 Bell Hall, University at Buffalo, Buffalo, NY 14260. Email: samik@cse.buffalo.edu.

$L - S$  is not  $\leq_m^p$ -hard for NP. Hence, removing sparse information in P from a complete set leaves the set complete, while removing sparse information in EXP from a complete set may destroy its completeness.

In the fourth section of this paper we build on results of Agrawal [Agr02], who demonstrated that pseudorandom generators can be used to prove structural theorems on complete degrees. We use hypotheses about pseudorandom generators to answer the longstanding open question of whether NP-complete sets can be immune. Assuming the existence of pseudorandom generators and secure one-way permutations, we prove easily that no NP-complete set is p-immune. (This too is a well-known property of the EXP-complete sets.) Assuming only that secure one-way permutations exist, we prove that no NP-complete set is  $\text{DTIME}(2^{n^\epsilon})$ -immune. Also, we use this hypothesis to show that no NP-complete set is quasipolynomial-close to P. It is already known [Ogi91, Fu93] that no NP-complete set is p-close to a set in P unless  $P = \text{NP}$ .

The fifth section studies the question of whether the union of disjoint Turing-complete sets for NP is Turing-complete. Here is the background. If  $A$  and  $B$  are two disjoint computably enumerable (c.e.) sets, then  $A \leq_T A \cup B$ ,  $B \leq_T A \cup B$ , and it follows that if either  $A$  or  $B$  is Turing-complete for the c.e. sets, then so is  $A \cup B$  [Sho76]. The proofs are straightforward: To demonstrate that  $A \leq_T A \cup B$ , on input  $x$ , ask whether  $x \in A \cup B$ . If not, then  $x \notin A$ . Otherwise, simultaneously enumerate  $A$  and  $B$  until  $x$  is output. The proof suggests that these properties may not hold for  $\leq_T^p$ -complete sets for NP. In particular Selman [Sel88] raised the question of whether the union of two disjoint  $\leq_T^p$ -complete sets for NP is  $\leq_T^p$ -complete. It is unlikely that  $A \leq_T^p A \cup B$ , for every two disjoint sets  $A$  and  $B$  in NP, for if this holds then  $\text{NP} \cap \text{coNP} = \text{P}$ : Take  $A \in \text{NP} \cap \text{coNP}$ ; then  $\bar{A} \in \text{NP} \cap \text{coNP}$  as well, and  $A \leq_T^p (A \cup \bar{A}) \implies A \in \text{P}$ .

First, we will prove that if  $\text{UEE} \neq \text{EE}$ , then there exist two disjoint languages  $A$  and  $B$  in NP such that  $A \not\leq_T^p A \cup B$ . Second, we introduce the following reasonable but strong hypothesis: There is a UP-machine  $M$  that accepts  $0^*$  such that for some  $0 < \epsilon < 1$ , no  $2^{n^\epsilon}$  time-bounded machine can correctly compute infinitely many accepting computations of  $M$ . This hypothesis is similar to hypotheses used in several earlier papers [FFNR96, HRW97, FPS01, PS01]. We prove, assuming this hypothesis, that there exist disjoint Turing-complete sets for NP whose union is not Turing-complete. Also, we show that if  $\text{UP} \cap \text{coUP}$  contains  $\text{DTIME}(2^{n^\epsilon})$ -bi-immune sets, then this hypothesis is true. Finally, we make several observations about the question of whether the union of two disjoint NP-complete sets is NP-complete. It would be difficult to obtain results about these questions without introducing hypotheses about complexity classes, because there are oracles relative to which the answers to these questions are both positive and negative. Proofs that would settle these questions would not relativize to all oracles.

**2. Preliminaries.** We use standard notation and assume familiarity with standard resource-bounded reducibilities. Given a complexity class  $\mathcal{C}$  and a reducibility  $\leq_r$ , a set  $A$  is  $\leq_r$ -hard for  $\mathcal{C}$  if for every set  $L \in \mathcal{C}$ ,  $L \leq_r A$ . The set  $A$  is  $\leq_r$ -complete if, in addition,  $A \in \mathcal{C}$ . We use the phrase “NP-complete” to mean  $\leq_m^p$ -complete for NP.

A set  $S$  is *sparse* if there exists a polynomial  $p$  such that for all positive integers  $n$ ,  $|\{S \cap \Sigma^n\}| \leq p(n)$ . We use polynomial-time invertible pairing functions  $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ .

A set  $S$  is *p-selective* [Sel79] if there is a polynomial-time-computable function  $f : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$  such that for all words  $x$  and  $y$ , (i)  $f(x, y) = x$  or  $f(x, y) = y$  and (ii)  $x \in A$  or  $y \in A$  implies  $f(x, y) \in A$ .

A set  $L$  is *immune* to a complexity class  $\mathcal{C}$ , or  $\mathcal{C}$ -immune, if  $L$  is infinite and no infinite subset of  $L$  belongs to  $\mathcal{C}$ . A set  $L$  is *bi-immune* to a complexity class  $\mathcal{C}$ , or  $\mathcal{C}$ -bi-immune, if both  $L$  and  $\bar{L}$  are  $\mathcal{C}$ -immune.

**3. Robustness.** In this section we consider the following question: If  $L$  is NP-complete and  $S$  is a sparse set then does  $L - S$  remain complete? This question was studied for exponential time complexity classes by Tang, Fu, and Liu [TFL93] and by Buhrman, Hoene, and Torenvliet [BHT98]. The basic result [TFL93] is that there exists a subexponential-time computable sparse set  $S$  such that for every  $\leq_m^p$ -complete set  $L$  for EXP,  $L - S$  is not EXP-complete. On the other hand, for any p-selective sparse set  $S$ ,  $L - S$  still remains hard [BHT98]. The theorems of Tang, Fu, and Liu depend on the fact that for any exponential time computable set  $B$  and any exponential time complete set  $A$ , there exists a length-increasing, one-one reduction from  $B$  to  $A$  [BH77]. We don't know that about NP. Nevertheless, here we prove the analogues of these results for NP. Observe that our first result, Theorem 3.1, holds unconditionally.

**THEOREM 3.1.** *Let  $L$  be an NP-complete set and let  $S$  be a p-selective sparse set such that  $L \not\subseteq S$ . Then  $L - S$  is  $\leq_m^p$ -hard for NP.*

*Proof.* Note that  $L - S \neq \emptyset$ . If  $L - S$  is finite, then  $L$  is sparse as well. Since  $L$  is  $\leq_m^p$ -complete for NP,  $\text{NP} = \text{P}$  [Mah82]. Therefore,  $L - S$  is also NP-complete. So we assume that  $L - S$  is infinite in the rest of the proof.

We use the left set technique of Ogiwara and Watanabe [OW91]. Assume that  $M$  is a nondeterministic machine that accepts  $L$ . Let  $T_x$  be the computation tree of  $M$  on any string  $x$ . Without loss of generality, assume that  $T_x$  is a complete binary tree, and let  $d$  be the depth of  $T_x$ . Given two nodes  $u$  and  $v$  in  $T_x$ , we say that  $u < v$  if the path from the root to  $u$  lies to the left of the path from the root to  $v$ , and  $u \leq v$  if either  $u < v$  or  $u$  lies on the path from the root to  $v$ . Let

$$\text{Left}(L) = \{ \langle x, u \rangle \mid \exists v, u \leq v, u, v \in T_x, \text{ an accepting computation of } M \text{ on } x \text{ passes through } v \}.$$

Since  $L$  is NP-complete and  $\text{Left}(L)$  is in NP,  $\text{Left}(L) \leq_m^p L$  via some  $f \in \text{PF}$ . When it is understood that  $v \in T_x$ , we will write  $v$  as an abbreviation for  $\langle x, v \rangle$  and  $f(v)$  as an abbreviation for  $f(\langle x, v \rangle)$ . Given  $x$  of length  $n$ , the length of every node of  $T_x$  is bounded by a polynomial in  $n$ . Since  $f$  is polynomial-time computable, the length of  $f(v)$ , where  $v \in T_x$ , is bounded by  $p(n)$ , for some polynomial  $p(\cdot)$ . We call  $f(v)$  the *label* of  $v$ . Since  $S$  is sparse, there is a polynomial bound  $q(n)$  on the number of strings in  $S$  of length at most  $p(n)$ . Let  $g(\cdot, \cdot)$  be the selector function for  $S$ . Consider the following total preorder [Tod91] on some  $Q \subseteq \Sigma^{\leq p(n)}$ .

$$\begin{aligned} x \leq_g y &\iff \exists z_1, z_2, \dots, z_m \in Q, \\ &g(x, z_1) = x, g(z_1, z_2) = z_1, \dots, \\ &g(z_{m-1}, z_m) = z_{m-1}, g(z_m, y) = z_m. \end{aligned}$$

Observe that if  $x \leq_g y$  and  $y \in S$ , then  $x \in S$  also. Given the selector  $g$ , the strings in  $Q$  can be ordered by  $\leq_g$  in time polynomial in the sum of the lengths of the strings in  $Q$ . Therefore, if  $\|Q\|$  is polynomial in  $n$ , then the strings in  $Q$  can be ordered by  $\leq_g$  in time polynomial in  $n$  as well.

We first make a few simple observations.

**OBSERVATION 1.** *If  $u < v$ , and  $w$  is a descendant of  $u$ , then  $w < v$ .*

**OBSERVATION 2.** *Let  $v$  be the left most node of  $T_x$  at some level. Then*

$$x \in L \iff v \in \text{Left}(L) \iff f(v) \in L.$$

**OBSERVATION 3.** Let  $X = \{x_1, x_2, \dots\} \subseteq \Sigma^{\leq p(n)}$  be a set of more than  $q(n)$  distinct strings. Then there exists a procedure that runs in time polynomial in  $n$  and outputs  $x_i \notin S, i \leq q(n) + 1$ .

*Proof.* Order the first  $q(n) + 1$  strings in  $X$  by  $\leq_g$ , and output a highest string as  $x_i$ . Since there can be at most  $q(n)$  strings of length  $\leq p(n)$  in  $S$ ,  $x_i$  cannot be in  $S$ .  $\square$

We now define a reduction from  $L$  to  $L - S$ . Before we give the formal algorithm, we present the idea behind the reduction. At any time during the reduction our reduction maintains a list of nodes in  $T_x$  and a node called *special*. Given  $x$ , we perform a breadth first search on  $T_x$ . If at any level of the search, we find two nodes  $u$  and  $v$  such that  $f(u) = f(v)$ , then we do an Ogiwara–Watanabe pruning. If at any stage, more than  $q(n)$  nodes remain after the pruning, then we can find the left most node *special* such that  $f(\text{special})$  is not in  $S$ . The list contains all the nodes that are to the left of *special*. At this point, we would like stop the search and output  $f(\text{special})$ . However, it is possible that the rightmost accepting computation passes through a node in the list and so *special* is not in  $\text{Left}(L)$ , thus the reduction is wrong. Thus before we output  $f(\text{special})$ , we have to verify that the right most accepting computation does not pass through a node in the list. However, it is not clear that this can be done in polynomial time.

To get around this, we further expand the nodes in the list and do an Ogiwara–Watanabe pruning again. This process may redefine the node *special*. We stop the search when every node in the list is a leaf node. If any of the leaf nodes is an accepting node, then we know that  $x$  is in  $L$ , and we output a fixed string not in  $L - S$ . Otherwise, we output  $f(\text{special})$  and argue that the rightmost accepting computation (if exists) must either pass through *special* or lies to the right of it.

Now, we give a formal description of the reduction.

On input  $x, |x| = n$ , the reduction traverses  $T_x$  in stages. During Stage  $k$ , the reduction maintains a list  $\text{list}_k$  of nodes in  $T_x$  at level  $k$ . The reduction procedure has a variable called “special” which holds some node of  $T_x$ . At Stage 1,  $\text{list}_1$  contains the root of  $T_x$  and the value of special is undefined. Now we define Stage  $k > 1$ .

**Step 1** Let  $\text{list}_{k-1} = \langle v_1, v_2, \dots, v_t \rangle$ .

**Step 2** Let  $u'_1 < u'_2 < \dots < u'_{2t}$  be the children of nodes in  $\text{list}_{k-1}$ . This ordering is possible since all nodes in  $\text{list}_{k-1}$  are at depth  $k - 1$  of the tree  $T_x$ , and therefore  $u'_1, \dots, u'_t$  are at level  $k$ . Put all these nodes in  $\text{list}_k$ .

**Step 3: Pruning** If there exist two nodes  $u'_i$  and  $u'_l, i < l$ , in  $\text{list}_k$  such that  $f(u'_i) = f(u'_l)$ , then remove  $u'_i, \dots, u'_{l-1}$  from  $\text{list}_k$ . Now let  $u_1 < \dots < u_m$  be the nodes in  $\text{list}_k$ , where every  $u_i$  has distinct labels. If  $m \leq q(n)$ , go to the next stage.

**Step 4** It must be the case that  $m > q(n)$ . Therefore, by Observation 3, there must be some  $j \leq q(n) + 1$  such that  $f(u_j) \notin S$ . Set  $\text{special} = u_j$ .

**Step 5** If special is the leftmost node of  $T_x$  at level  $k$ , then output special and halt.

**Step 6** Otherwise, place  $u_1, \dots, u_{j-1}$  in  $\text{list}_k$  and go to the next stage.

The following algorithm  $h$  defines the reduction from  $L$  to  $L - S$ :

```

for  $k = 1$  to  $d$ 
  run Stage  $k$ 
if any stage halts and outputs  $v$ , then
  output  $f(v)$ 
else /*  $\text{list}_d$  contains some leaf nodes of  $T_x$  */
  if any of the leaf nodes is an accepting computation
    of  $M$  on  $x$ , then output a predetermined fixed
    string  $w \in L - S$ 
  else

```

```

        output  $f(\text{special})$ 
    endif
endif

```

We prove that the above reduction is correct by the following series of claims:

CLAIM 3.1. *For any  $k < d$ , if Stage  $k$  outputs a string  $v$ , then*

$$x \in L \Leftrightarrow f(v) \in L - S.$$

*Proof.* If Stage  $k$  outputs  $v$ , then  $v$  is the leftmost node of  $T_x$  at level  $k$  and  $f(v) \notin S$ . By Observation 2, the claim follows.  $\square$

From now assume that for no  $k$ , Stage  $k$  halts in Step 5. First we make some observations.

OBSERVATION 4. *During Stage  $k \geq 1$ ,  $\|\text{list}_k\| \leq q(n)$ .*

*Proof.* For any Stage  $k$ , assume that  $\text{list}_{k-1}$  has  $t \leq q(n)$  nodes. The number of nodes in  $\text{list}_k$  before pruning is at most  $2t$ . After the pruning step, every  $v \in \text{list}_k$  has a different label. If there are  $\leq q(n)$  nodes in  $\text{list}_k$ , then the procedure goes to the next stage. Otherwise, the node  $u_j$  where  $j \leq q(n) + 1$  has a label outside  $S$ . Since we assume that Stage  $k$  does not halt in Step 5, the procedure goes to Stage  $k$  with  $\|\text{list}_k\| = j - 1 \leq q(n)$ .  $\square$

OBSERVATION 5. *Suppose  $\text{special} = v$  at the end of Stage  $k$ . Then for  $l \geq k$ ,  $\forall u \in \text{list}_l, u < v$ .*

*Proof.* At the end of Stage  $k$ , let  $v = \text{special} = u_j$ . After Step 6,  $\text{list}_k$  is a subset of  $\{u_1, \dots, u_{j-1}\}$ . Thus  $\forall u \in \text{list}_k, u < v$ . Note that in any subsequent Stage  $l > k$ , the nodes that belong to  $\text{list}_l$  are the descendants of nodes in  $\text{list}_k$ . By Observation 1, we obtain the proof.  $\square$

OBSERVATION 6. *No node that is pruned in Step 3 can be on the path containing the rightmost accepting computation.*

*Proof.* If  $x \notin L$ , no node in  $T_x$  is on the path containing any accepting computation. Therefore, let us assume that  $x \in L$ . If two nodes  $u'_i$  and  $u'_i$  at the same depth have the identical label  $w$ , then  $f(u'_i) \in \text{Left}(L) \iff f(u'_i) \in \text{Left}(L)$ . Therefore, if any  $u'_k$  at the same depth is on the path of the rightmost accepting computation, then either  $k < i$  or  $k \geq l$ . Since only the nodes  $u'_i, \dots, u'_{i-1}$  are pruned,  $u'_k$  cannot be pruned.  $\square$

CLAIM 3.2. *Assume that  $x \in L$  and Stage  $k \geq 1$  does not halt in Step 5. If  $\exists v \in \text{list}_k$  that is on the path containing the rightmost accepting computation, then either  $\exists u \in \text{list}_{k+1}$  that is on the path containing the rightmost accepting computation, or  $\text{special} \in \text{Left}(L)$ .*

*Proof.* Since there is a node  $v$  in  $\text{list}_k$  that is on the path containing the rightmost accepting computation, let  $u'_r$  be the node that is generated at Step 2 of Stage  $k + 1$  that is on the path containing the rightmost accepting computation. By Observation 6,  $u'_r$  cannot get pruned in Step 3, and therefore, it is in  $\text{list}_k$  at Step 4. Let us denote this node by  $u_r$ . If a node  $u_j$  is assigned  $\text{special}$  in Step 4, then either  $j \leq r$ , in which case  $\text{special} \in \text{Left}(L)$ . Otherwise,  $r < j$ , and therefore,  $u_r$  is in  $\text{list}_{k+1}$  after Step 6.  $\square$

CLAIM 3.3. *If for every  $k$ , Stage  $k$  does not halt in Step 5, then  $x \in L$  if and only if  $\text{list}_d$  contains a leaf node that is an accepting computation or  $\text{special} \in \text{Left}(L)$ .*

*Proof.* Note that if  $x$  is not in  $L$ , then no leaf node can be accepting, and no node of  $T_x$  can be in  $\text{Left}(L)$ . Therefore, the *if* direction is trivial. We show the *only if* direction. We prove the following by induction on the number of stages: If  $x \in L$ , then after Stage  $k$ , either the rightmost accepting computation passes through a node in  $\text{list}_k$  or  $\text{special} \in \text{Left}(L)$ .

After Stage 1,  $\text{list}_1$  contains the root of the tree. Thus the claim is true after Stage 1. Assume that the claim is true after Stage  $k - 1$ . Thus either the rightmost accepting computation passes through a node in  $\text{list}_{k-1}$  or  $\text{special} \in \text{Left}(L)$ . We consider two cases.

**Case 1:** The rightmost accepting computation passes through a node in  $\text{list}_{k-1}$ . By Claim 3.2, either there is a node in  $\text{list}_k$  that is on the path of the rightmost accepting computation, or the node that is assigned `special` during Stage  $k$  is in  $\text{Left}(L)$ .

**Case 2:** `special`  $\in \text{Left}(L)$ . Let  $s$  be the node that is currently assigned to `special`. It suffices to show that if a node  $u$  is assigned to `special` at Stage  $k$ , then  $u$  will also be in  $\text{Left}(L)$ . By Observation 5, for every node  $v \in \text{list}_{k-1}$ ,  $v < s$ . Since  $u$  is a descendant of some node  $v$  in  $\text{list}_{k-1}$ ,  $u < s$  as well. Therefore,  $s \in \text{Left}(L) \implies u \in \text{Left}(L)$ .

Therefore, after Stage  $k$ ,  $k \geq 1$ , the rightmost accepting computation of  $M$  either passes through a node in  $\text{list}_k$  or `special`  $\in \text{Left}(L)$ . When  $k = d$ , this implies that either the rightmost accepting computation is a node in  $\text{list}_d$ , or `special`  $\in \text{Left}(L)$ . This completes the proof.  $\square$  The correctness of the reduction now follows.

CLAIM 3.4. *The reduction  $h(\cdot)$  is correct, and it runs in polynomial time.*

*Proof.* If the the reduction halts at Step 5 during any stage, then by Claim 3.1  $x \in L \Leftrightarrow h(x) \in L - S$ . Assume that no stage halts in Step 5. Assume  $x \in L$ . By Claim 3.3, either  $\text{list}_d$  contains an accepting leaf or `special`  $\in \text{Left}(L)$ . If  $\text{list}_d$  contains an accepting computation, then  $h(x) = w \in L - S$ . Otherwise, if `special`  $\in \text{Left}(L)$ , then  $f(\text{special}) \in L$ . However, by the definition of `special`,  $f(\text{special}) \notin S$ . Therefore,  $f(\text{special}) \in L - S$ . On the other hand, if  $x \notin L$ , then no node of  $T_x$  can be in  $\text{Left}(L)$ , and so, in particular, `special`  $\notin \text{Left}(L)$ . Therefore,  $h(x) = f(\text{special}) \notin L$ .

By Observation 4, the number of nodes in  $\text{list}_k$  for any  $k \geq 1$  is bounded by  $q(n)$ . Therefore, the number of nodes visited by the reduction is at most  $d \times 2q(n)$ . Since  $d$  is bounded above by the running time of  $M$  on  $x$ , the total time required by the reduction is at most polynomial in  $n$ .  $\square$

Therefore,  $L \leq_m^p L - S$ . So  $L - S$  is  $\leq_m^p$ -hard for NP.  $\square$

COROLLARY 3.2. *Let  $L$  be a  $\leq_m^p$ -complete set for NP, and  $S \in \text{P}$  be sparse. Then  $L - S$  is  $\leq_m^p$ -complete for NP.*

Observe that the reduction  $h$  that is constructed in the proof of Theorem 3.1 actually satisfies the following property.

$$\begin{aligned} x \in L &\implies h(x) \in L - S \\ x \notin L &\implies h(x) \in \overline{L} \end{aligned}$$

So for any  $S' \subseteq S$  it holds that  $L \leq_m^p L - S'$  via  $h$ . This shows the following generalization of Theorem 3.1.

THEOREM 3.3. *Let  $L$  be a  $\leq_m^p$ -complete set of NP, and let  $S$  be a subset of a sparse  $p$ -selective set. Then  $L - S$  is  $\leq_m^p$ -hard for NP.*

In contrast to the theorem we just proved, in Theorem 3.5, we construct a sparse set  $S \in \text{DTIME}(2^{2^n})$  such that for any set  $L \in \text{NP} - \text{P}$ ,  $L - S$  is not  $\leq_m^p$ -hard for NP. Again, as in Theorem 3.1, we cannot assert that  $L - S \in \text{NP}$ . In Corollary 3.6, we obtain that for every  $L \in \text{NP} - \text{P}$ , there is a sparse  $S \in \text{EXP}$  such that  $L - S$  is not  $\leq_m^p$ -hard for NP.

The following lemma shows a collapse to P for a restricted form of truth-table reduction from  $\overline{\text{SAT}}$  to a sublogarithmically-dense set. In other words, we show that if  $\overline{\text{SAT}}$  disjunctively reduces to some sublogarithmically-dense set where the reduction machine makes logarithmically many nonadaptive queries, then  $\text{NP} = \text{P}$ . We exploit this strong consequence in Theorem 3.5 below.

LEMMA 3.4. *If there exist  $f \in \text{FP}$ ,  $S \subseteq \Sigma^*$ , and a real number  $\alpha < 1$  such that*

1. *for all  $n \geq 0$ ,  $\|S^{\leq n}\| \leq O(\log^\alpha n)$ , and*
2. *for all  $x$ ,  $f(x)$  is a set of words such that  $\|f(x)\| \leq O(\log |x|)$  and*

$$x \in \text{SAT} \Leftrightarrow f(x) \cap S = \emptyset,$$

then  $P = NP$ .

*Proof.* Assume  $f$ ,  $S$ , and  $\alpha$  exist. Let

$$\text{LeftSAT} \stackrel{\text{df}}{=} \{ \langle x, z \rangle \mid \text{formula } x \text{ has a satisfying} \\ \text{assignment } y \geq z \}.$$

Note that for a formula  $x$  with  $n$  variables,  $\langle x, 0^n \rangle \in \text{LeftSAT} \iff x \in \text{SAT}$ . Also, LeftSAT is in NP. Let us assume that  $\text{LeftSAT} \leq_m^p \text{SAT}$  via reduction  $g \in \text{PF}$ . Let  $h(w) \stackrel{\text{df}}{=} f(g(w))$  and let  $p(\cdot)$  be the computation time of  $h$ . Therefore, by assumption, for all  $w$ ,  $h(w)$  is a set of words such that  $\|h(w)\| \leq O(\log |w|)$  and

$$w \in \text{LeftSAT} \iff g(w) \in \text{SAT} \iff h(w) \cap S = \emptyset.$$

Therefore, for every  $S' \subseteq S$ , and for all  $x, y$ ,

$$\langle x, y \rangle \in \text{LeftSAT} \implies h(\langle x, y \rangle) \cap S' = \emptyset.$$

Choose constants  $c$  and  $d$  such that  $\|S^{\leq n}\| \leq c \log^\alpha n$  and  $\|h(w)\| \leq d \log |w|$ . Below we describe a *nondeterministic* polynomial-time-bounded algorithm that accepts SAT. We will see that this algorithm can be simulated in *deterministic* polynomial time. The input is a formula  $x$ .

- 1  $S' := \emptyset$
- 2  $n :=$  number of variables in  $x$
- 3 if  $1^n$  satisfies  $x$  then accept  $x$   
/\* Otherwise,  $\langle x, 1^n \rangle \notin \text{LeftSAT}$ , and so  $h(\langle x, 1^n \rangle) \cap S \neq \emptyset$ . \*/
- 4 choose some  $s \in h(\langle x, 1^n \rangle)$  nondeterministically
- 5  $S' := S' \cup \{s\}$
- 6 for  $i = 1$  to  $c \log^\alpha p(|x| + n)$
- 7 if  $h(\langle x, 0^n \rangle) \cap S' \neq \emptyset$  then reject  
/\* At this point,  $h(\langle x, 0^n \rangle) \cap S' = \emptyset$ , and  $h(\langle x, 1^n \rangle) \cap S' \neq \emptyset$ . \*/
- 8 Use binary search to determine a word  $y \in \Sigma^n - \{1^n\}$   
such that  $h(\langle x, y \rangle) \cap S' = \emptyset$  and  $h(\langle x, y + 1 \rangle) \cap S' \neq \emptyset$ .
- 9 if  $y$  satisfies  $x$  then accept
- 10 choose some  $s \in h(\langle x, y \rangle)$  nondeterministically
- 11  $S' := S' \cup \{s\}$
- 12 increment  $i$
- 13 reject

**We argue that the algorithm runs in nondeterministic polynomial time:** The loop in steps 6 – 12 runs at most  $c \log^\alpha p(|x| + n)$  times, and the binary search takes at most  $O(n)$  steps for a formula of  $n$  variables. Therefore, the runtime is bounded by a polynomial in  $(n + |x|)$ .

**We argue that the algorithm accepts SAT:** The algorithm accepts only if we find a satisfying assignment (step 3 or step 9). So all unsatisfiable formulas are rejected. We now show that all satisfiable formulas are accepted by at least one computation path.

Let  $x$  be a satisfiable formula; we describe an accepting computation path. On this path,  $S'$  will always be a subset of  $S$ . If  $x$  is accepted in step 3, then we are done. Otherwise,  $\langle x, 1^n \rangle \notin \text{LeftSAT}$  and therefore,  $h(\langle x, 1^n \rangle) \cap S \neq \emptyset$ . So in step 4 at least one computation path chooses some  $s \in S$ .

Since  $x \in \text{SAT}$ ,  $\langle x, 0^n \rangle \in \text{LeftSAT}$ . Hence  $h(\langle x, 0^n \rangle) \cap S = \emptyset$ . Since  $S' \subseteq S$ , it follows that  $h(\langle x, 0^n \rangle) \cap S' = \emptyset$ . Therefore, if  $x \in \text{SAT}$ , the nondeterministic path that makes the correct choice for  $s$  in step 4 cannot reject  $x$  in step 7. Now we have

$$h(\langle x, 0^n \rangle) \cap S' = \emptyset, \text{ and } h(\langle x, 1^n \rangle) \cap S' \neq \emptyset.$$

Therefore, there must be some  $y$  as required by the algorithm, which can be obtained by binary search as follows. Initially, the algorithm considers the interval  $[0^n, 1^n]$  and choose the middle element  $10^{n-1}$ . If  $h(\langle x, 10^{n-1} \rangle) \cap S' \neq \emptyset$ , then we proceed with the interval  $[0^n, 10^{n-1}]$ . Otherwise, we proceed with the interval  $[10^{n-1}, 1^n]$ . By continuing this procedure, we obtain intervals  $[a, b]$  of decreasing size such that  $a < b$  and

$$h(\langle x, a \rangle) \cap S' = \emptyset, \text{ and } h(\langle x, b \rangle) \cap S' \neq \emptyset.$$

If we accept in step 9, then we are done. Otherwise we can argue as follows: By step 8, we have  $h(\langle x, y+1 \rangle) \cap S' \neq \emptyset$  and therefore,  $h(\langle x, y+1 \rangle) \cap S \neq \emptyset$ . Hence  $\langle x, y+1 \rangle \notin \text{LeftSAT}$ . Together with the fact that  $y$  does not satisfy  $x$  in step 9, we obtain  $\langle x, y \rangle \notin \text{LeftSAT}$ . Therefore,  $h(\langle x, y \rangle) \cap S \neq \emptyset$ . On the other hand,  $h(\langle x, y \rangle) \cap S' = \emptyset$ . Therefore, the correct nondeterministic path can choose an  $s \in S - S'$  and continues with the next iteration of the loop. Along this path,  $S'$  is always a subset of  $S \cap \Sigma^{\leq p(|x|+n)}$ . By assumption,

$$\|S^{\leq p(|x|+n)}\| \leq c \cdot \log^\alpha p(|x| + n).$$

We enter the loop with  $\|S'\| = 1$ , and in each iteration we add a new element to  $S'$ . Hence at the beginning of the  $(c \cdot \log^\alpha p(|x| + n))$ -th iteration it holds that  $S' = S \cap \Sigma^{\leq p(|x|+n)}$ . Now consider this iteration at step 8. Elements of  $h(\langle x, y \rangle)$  and elements of  $h(\langle x, y+1 \rangle)$  are of length  $\leq p(|x| + n)$ . So in this iteration we obtain a word  $y$  such that

$$h(\langle x, y \rangle) \cap S = \emptyset,$$

and

$$h(\langle x, y+1 \rangle) \cap S \neq \emptyset.$$

It follows that  $\langle x, y \rangle \in \text{LeftSAT}$  and  $\langle x, y+1 \rangle \notin \text{LeftSAT}$ . So  $y$  is the lexicographically largest satisfying assignment of  $x$ . Therefore, we accept in step 9. It follows that our algorithm accepts SAT.

**We argue that the algorithm can be simulated in deterministic polynomial time:**

Clearly, each path of the nondeterministic computation is polynomially bounded. We estimate the total number of paths as follows. Each path has at most

$$c \cdot \log^\alpha p(|x| + n) + 1$$

nondeterministic choices, where  $\alpha < 1$ . Each such nondeterministic choice guesses an  $s \in h(\langle x, y \rangle)$  for some  $y \in \Sigma^n$ . By assumption,  $\|h(\langle x, y \rangle)\| \leq d \cdot \log(|x| + n)$ . Hence the total number of paths is

$$\begin{aligned} & (d \cdot \log(|x| + n))^{c \cdot \log^\alpha p(|x|+n)+1} \\ & \leq 2^{O(\log \log(|x|+n)) \cdot O(\log^\alpha(|x|+n))} \\ & \leq 2^{O(\log^{1-\alpha}(|x|+n)) \cdot O(\log^\alpha(|x|+n))} \\ & \leq 2^{O(\log(|x|+n))} \\ & \leq (|x| + n)^{O(1)}. \end{aligned}$$

Hence there is only a polynomial number of nondeterministic paths. Therefore, the algorithm can be simulated in deterministic polynomial time.  $\square$

**THEOREM 3.5.** *There exists a sparse  $S \in \text{DTIME}(2^{2^n})$  such that for every  $L \in \text{NP} - \text{P}$ ,  $L - S$  is not  $\leq_m^p$ -hard for NP.*

*Proof.* Let  $\{N_i\}_{i \geq 0}$  be an enumeration of all nondeterministic polynomial-time-bounded Turing machines such that for all  $i$ , the running time of  $N_i$  is bounded by the polynomial  $p_i(n) = n^i + i$ . Similarly, let  $\{f_j\}_{j \geq 0}$  be an enumeration of all polynomial-time computable functions such that for all  $j$ , the running time of  $f_j$  is bounded by the polynomial  $p_j(n) = n^j + j$ . We use a polynomial-time computable and polynomial-time invertible pairing function  $\langle \cdot, \cdot \rangle$  such that  $r = \langle i, j \rangle$  implies  $i \leq r$  and  $j \leq r$ .

A *requirement* is a natural number  $r$ . If  $r = \langle i, j \rangle$ , then we interpret this as the requirement that  $L(N_i)$  does not many-one reduce to  $L(N_i) - S$  via reduction function  $f_j$ .

Let  $t(m) = 2^{2^m}$ . We describe a decision algorithm for  $S$ . Let  $w$  be the input and let  $n = |w|$ .

```

1  if  $|w| < 4$  then reject
2   $n := |w|$ ,  $m :=$  greatest number such that  $t(m) \leq n$ 
3  for  $k = 1$  to  $m$ 
4       $S_k := \emptyset$ ,  $L_k := \emptyset$ 
5      for  $r = 1$  to  $k$ 
6          if  $r \notin L_1 \cup L_2 \cup \dots \cup L_{k-1}$  then
7              determine  $i$  and  $j$  such that  $r = \langle i, j \rangle$ ,
8              for all  $z \in \Sigma^{<t(k+2)}$  in increasing lexicographic order
9                   $y := f_j(z)$ 
10                 if  $t(k) \leq |y| < t(k+1)$  and  $N_i(z)$  accepts then
11                      $S_k := \{y\}$ 
12                      $L_k := \{r\}$ 
13                 exit the loops for  $z$  and  $r$ , and consider next  $k$ 
14             endif
15         increment  $z$ 
16     endif
17     increment  $r$ 
18     increment  $k$ 
19     accept if and only if  $S_m = \{w\}$ 
    
```

The algorithm works in stages  $1, \dots, m$  where  $m$  is the greatest natural number such that  $t(m) \leq n$ . In stage  $k$ , we construct a set  $S_k$  such that  $\|S_k\| \leq 1$  and

$$S_k = \{w \in S \mid t(k) \leq |w| < t(k+1)\}.$$

Hence  $S$  can be written as  $S_1 \cup S_2 \cup \dots$ . The input  $w$  is accepted if and only if it belongs to  $S_m$ . The aim of stage  $k$  is to satisfy a requirement  $r \in \{1, \dots, k\}$  that has not been satisfied so far (i.e.,  $r \notin L_1 \cup \dots \cup L_{k-1}$ ). If more than one requirement is satisfiable, then we choose the smallest one. Requirement  $r = \langle i, j \rangle$  is satisfied by placing a string  $y$  into  $S_k$  and therefore into  $S$  such that  $f_j(z) = y$  for a suitable  $z \in L(N_i)$ . This makes sure that  $L(N_i)$  does not many-one reduce to  $L(N_i) - S$  via reduction  $f_j$ .

Whenever we refer to (the value of) a program variable without mentioning the time when we consider this variable, then we mean the value of the variable when the algorithm stops. Variables  $L_k$  represent sets of requirements. If requirement  $i$  is satisfied in stage  $k$ , then  $i$  is added to the set  $L_k$ . The algorithm ensures that  $\|L_k\| \leq 1$  for every  $k$ .

**We observe that  $S$  is sparse:** For all inputs of length  $\geq t(k)$  the algorithm constructs the same sets  $S_1, S_2, \dots, S_m$  and  $L_1, L_2, \dots, L_m$ . Therefore, it is unambiguous to refer to  $S_k$  and  $L_k$ . Moreover, it is immediately clear that any  $S_k$  contains at most one word, and this word, if it exists, has a length that belongs to the interval  $[t(k), t(k+1))$ . By definition of

$t(k)$ , for every  $n \geq 4$ ,

$$t(\lfloor \log \log n \rfloor) \leq n < t(\lfloor \log \log n \rfloor + 1).$$

Hence on input of some word of length  $n \geq 4$ , we have

$$m = \lfloor \log \log n \rfloor \tag{3.1}$$

in step 2. So the algorithm computes singletons  $S_1, S_2, \dots, S_m$  such that  $S^{\leq n} \subseteq S_1 \cup S_2 \cup \dots \cup S_m$ . It follows that

$$\|S^{\leq n}\| \leq \lfloor \log \log n \rfloor. \tag{3.2}$$

In particular,  $S$  is sparse.

**We observe that  $S \in \text{DTIME}(2^{2^n})$ :** Note that in step 10,  $|z| < t(m+2) = t(m)^4 \leq n^4$  and  $i \leq r \leq m = \lfloor \log \log n \rfloor$ . So a single path of the nondeterministic computation  $N_i(z)$  has length

$$\leq n^{4i} + i \leq 2^{O(\log^2 n)}.$$

Hence the simulation of the complete computation takes

$$2^{2^{O(\log^2 n)}} \cdot 2^{O(\log^2 n)} = 2^{2^{O(\log^2 n)}}$$

steps. Similarly, step 9 takes  $2^{O(\log^2 n)}$  steps. So the loop at steps 8–15 takes at most

$$2^{n^4} \cdot 2^{2^{O(\log^2 n)}} = 2^{2^{O(\log^2 n)}}$$

steps. The loops 5–17 and 3–18 multiply this number of steps at most by factor

$$m^2 \leq \lfloor \log \log n \rfloor^2.$$

Therefore, the overall running-time is  $2^{2^{O(\log^2 n)}}$ . This shows

$$S \in \text{DTIME}(2^{2^{O(\log^2 n)}}) \subseteq \text{DTIME}(2^{2^n}).$$

**We observe that no  $L - S$  is many-one hard:** Let  $L \in \text{NP} - \text{P}$  and choose a machine  $N_i$  such that  $L = L(N_i)$ . Assume that  $L - S$  is  $\leq_m^p$ -hard for NP. Therefore, there exists  $j$  such that  $L \leq_m^p L - S$  via reduction function  $f_j$ . We consider two cases and show that both cases lead to contradiction. This will complete the proof.

*Case 1:* Assume there exists an  $e \geq 1$  such that for all  $x \in L^{\geq e}$ ,  $|f_j(x)| < \sqrt{|x|}$ . Using Lemma 3.4, we show that this implies  $L \in \text{P}$ , thereby obtaining a contradiction.

Consider an arbitrary formula  $x$ . Let  $s_0 \stackrel{\text{def}}{=} x$  and let  $s_{l+1} \stackrel{\text{def}}{=} f_j(s_l)$  for  $l \geq 0$ . By assumption, for all  $y \in L^{\geq e}$  it holds that

$$y \in L \iff f_j(y) \notin S \wedge f_j(y) \in L. \tag{3.3}$$

Hence

$$x \in L \iff s_1 \notin S \wedge s_1 \in L. \tag{3.4}$$

If  $|s_1| \geq e$ , we use equivalence (3.3) for  $y = s_1$ . We obtain

$$s_1 \in L \iff s_2 \notin S \wedge s_2 \in L. \quad (3.5)$$

By equivalences (3.5) and (3.4), we have

$$x \in L \iff s_1 \notin S \wedge s_2 \notin S \wedge s_2 \in L \quad (3.6)$$

Now we use equivalence (3.3) again, this time for  $y = s_2$ . We proceed in this way until we reach an  $s_k$  such that either  $|s_k| < e$  or  $|s_k| \geq \sqrt{|s_{k-1}|}$ . The following equivalence holds:

$$x \in L \iff \bigwedge_{l=1}^k s_l \notin S \wedge s_k \in L. \quad (3.7)$$

Note that if  $|s_k| < e$ , then it is easy to verify whether  $s_k$  belongs to  $L$ . So in polynomial time we can determine a string  $s$  which is defined as follows. If  $s_k \in L^{<e}$ , then let  $s$  be a fixed element from  $\overline{S}$ . Otherwise, let  $s$  be a fixed element from  $S$ . We show

$$x \in L \iff \{s_1, \dots, s_k, s\} \cap S = \emptyset. \quad (3.8)$$

“ $\Rightarrow$ ” Assume  $x \in L$ . Therefore,  $s_1, \dots, s_k \in L$ . If  $|x| < e$ , then  $k = 0$ ,  $s \in \overline{S}$ , and we are done. Otherwise,  $|x| \geq e$  and  $k \geq 1$ . If  $|s_k| < e$ , then, by equivalence (3.7),  $s \in \overline{S}$ . So from equivalence (3.7) it follows that  $\{s_1, \dots, s_k, s\} \cap S = \emptyset$ , and we are done. We show that the remaining case, i.e.,  $|s_k| \geq e$  is impossible. Since the algorithm terminated, it must be the case that  $|s_k| \geq \sqrt{|s_{k-1}|}$ . However, since  $x \in L$ ,  $s_k \in L$  by equivalence (3.7). By our assumption, it cannot happen that  $s_k \in L^{\geq e}$  and  $|s_k| \geq \sqrt{|s_{k-1}|}$ . Therefore,  $|s_k| < e$ .

“ $\Leftarrow$ ” Assume  $\{s_1, \dots, s_k, s\} \cap S = \emptyset$ . Hence  $s \in \overline{S}$  and therefore,  $s_k \in L^{<e}$ . From equivalence (3.7) we obtain  $x \in L$ . This shows equivalence (3.8).

For  $1 \leq l \leq k-1$  it holds that  $|s_l| < \sqrt{|s_{l-1}|}$ . Therefore,  $k \leq |x|$  and so the strings  $s$  and  $s_i$  can be constructed in polynomial time in  $|x|$ . Let  $g \in \text{FP}$  be the function that on input  $x$  computes the set  $\{s_1, \dots, s_k, s\}$ . So for all  $x$ ,

$$x \in L \iff g(x) \cap S = \emptyset. \quad (3.9)$$

Observe that for all  $x$ ,

$$\|g(x)\| \leq \lfloor \log \log |x| \rfloor + 1.$$

By assumption,  $L - S$  is many-one hard for NP. So there exists a reduction function  $h \in \text{FP}$  such that  $\text{SAT} \leq_m^p L - S$  via  $h$ . By equation (3.9), for all  $x$ ,

$$\begin{aligned} x \in \text{SAT} &\iff h(x) \in L \wedge h(x) \notin S \\ &\iff (g(h(x)) \cup \{h(x)\}) \cap S = \emptyset. \end{aligned}$$

With  $h'(x) \stackrel{\text{def}}{=} g(h(x)) \cup \{h(x)\}$  it holds that for all  $x$ ,

$$x \in \text{SAT} \iff h'(x) \cap S = \emptyset. \quad (3.10)$$

Clearly,  $h'$  belongs to FP and for all  $x$ ,

$$\|h'(x)\| \leq \lfloor \log \log |h(x)| \rfloor + 2 \leq \lfloor \log \log |x| \rfloor + c \quad (3.11)$$

for a suitable constant  $c$ . By equations (3.2), (3.10), and (3.11), we satisfy the assumptions of Lemma 3.4 (take  $h'$ ,  $S$ , and  $\alpha = 1/2$ ). It follows that  $L \in P$ . This contradicts our assumption.

*Case 2:* Assume there exist infinitely many  $x \in L$  such that  $|f_j(x)| \geq \sqrt{|x|}$ . We show that in this case  $L$  does not many-one reduce to  $L - S$  via  $f_j$ . This will give us the necessary contradiction.

Recall that  $L = L(N_i)$ . Let  $\bar{r} \stackrel{\text{def}}{=} \langle i, j \rangle$ . Since every non-empty  $L_k$  contains a unique requirement, we can choose a number  $m' \geq \bar{r}$  such that for all  $k \geq m'$ ,

$$L_k \cap \{0, \dots, \bar{r} - 1\} = \emptyset. \quad (3.12)$$

Note that for infinitely many strings  $x \in L$ ,  $|f_j(x)| \geq \sqrt{|x|}$ . Therefore, we can choose some string  $\bar{z} \in L$  such that

$$\sqrt{|\bar{z}|} \geq t(m') \quad (3.13)$$

and

$$|f_j(\bar{z})| \geq \sqrt{|\bar{z}|}. \quad (3.14)$$

Let  $w \stackrel{\text{def}}{=} f_j(\bar{z})$  and  $n \stackrel{\text{def}}{=} |w|$ . Let  $\bar{m}$  be such that  $t(\bar{m}) \leq n < t(\bar{m} + 1)$ . By the choice of  $\bar{z}$ ,  $|\bar{z}| < t(\bar{m} + 2)$ . We will show that if  $\bar{r}$  is not in  $L_1 \cup \dots \cup L_{\bar{m}-1}$ , then  $L_{\bar{m}} = \{\bar{r}\}$ .

Consider the algorithm on input  $w$ . By the choice of  $\bar{m}$ ,  $t(\bar{m}) \leq |w| = n < t(\bar{m} + 1)$ . Therefore, by the choice of  $m$  in step 2 of the algorithm,  $m = \bar{m}$ . Consider step 6 when  $k = \bar{m}$  and  $r = \bar{r}$ . We note that as a consequence of (3.12), for any  $k \geq \bar{m} \geq m'$ ,  $L_k \cap \{0, \dots, \bar{r} - 1\} = \emptyset$ . Therefore, the loop (step 5 – step 17) cannot exit with some  $r < \bar{r}$ . On the other hand, for  $\bar{r}$ , the condition in step 6 must be true, since we assumed that  $\bar{r} \notin L_1 \cup \dots \cup L_{\bar{m}-1}$ . Therefore, we reach step 7. By the choice of  $\bar{m}$ ,  $|\bar{z}| \leq t(\bar{m} + 2)$ . Therefore, either we reach step 9 such that  $z = \bar{z}$ , or  $\bar{r}$  is put in  $L_{\bar{m}}$  with some other  $z$  and  $S_{\bar{m}} = \{f_j(z)\}$ . If  $z = \bar{z}$ , then after step 9,  $y = f_j(\bar{z}) = w$  and therefore,  $|y| = n$ . Therefore, it must hold in step 10 that  $t(\bar{m}) \leq |y| < t(\bar{m} + 1)$ . Moreover,  $N_i(\bar{z})$  accepts since  $\bar{z} \in L$ . Therefore, we reach the steps 11 and 12 where we obtain that  $L_{\bar{m}} = \{\bar{r}\}$ . As a consequence,

$$\bar{r} \notin L_1 \cup \dots \cup L_{\bar{m}-1} \implies \bar{r} \in L_{\bar{m}}.$$

It follows that  $\bar{r} \in L_1 \cup L_2 \cup \dots \cup L_{\bar{m}}$ . Let  $k$ ,  $1 \leq k \leq \bar{m}$  be such that  $L_k = \{\bar{r}\}$ . Let  $S_k = \{y\}$ . By steps 9 and 10, there exists a  $z \in L$  such that  $y = f_j(z)$ . From  $y \in S$  it follows that  $y \notin L - S$ . Therefore,  $L$  does not many-one reduce to  $L - S$  via reduction function  $f_j$ . This contradicts our assumption.  $\square$

**COROLLARY 3.6.** *For every  $L \in \text{NP} - P$  there exists a sparse  $S \in \text{EXP}$  such that  $L - S$  is not  $\leq_m^p$ -hard for NP.*

*Proof.* Choose  $i$  such that  $L = L(N_i)$ . We recycle the proof of theorem 3.5. Here we only have to do the diagonalization against the machine  $N_i$ . So we interpret  $r$  as the requirement that  $L$  does not many-one reduce to  $L - S$  via reduction function  $f_r$ . We modify the algorithm in the proof of Theorem 3.5 by replacing step 7 with “ $j := r$ ”.

Analogously to the proof of Theorem 3.5 we observe that  $S$  is sparse. Because we modified the algorithm, now  $S$  belongs to EXP. This is seen as follows: Again in step 10,

$$|z| < t(m + 2) = t(m)^4 \leq n^4.$$

But now  $i$  is a constant. So a single path of the nondeterministic computation  $N_i(z)$  now has length  $\leq n^{4i} + i$ . Hence the simulation of the complete computation takes the following number of steps:

$$2^{n^{4i} + i} \cdot (n^{4i} + i) \leq 2^{n^{O(1)}}$$

Note that

$$j = r \leq m \leq \lceil \log \log n \rceil.$$

So step 9 takes

$$n^{4j} + j \leq 2^{O(\log^2 n)} \leq 2^{O(n)}$$

steps. Therefore, the loop at steps 8–15 takes at most

$$2^{n^4} \cdot 2^{n^{O(1)}} = 2^{n^{O(1)}}$$

steps. The loops 5–17 and 3–18 multiply this number of steps at most by factor

$$m^2 \leq \lceil \log \log n \rceil^2.$$

Therefore, the overall running-time remains  $2^{n^{O(1)}}$ . This shows  $S \in \text{EXP}$ .

Analogously to the proof of Theorem 3.5 we argue that  $L - S$  is not  $\leq_m^p$ -hard for NP. Here we have to define  $\bar{r} \stackrel{\text{def}}{=} j$  in Case 2.  $\square$

In view of Corollary 3.6 we would like to minimize the complexity of  $S$ . For this it suffices to consider  $L = \text{SAT}$ : Given a sparse set  $S$  such that  $\text{SAT} - S$  is not  $\leq_m^p$ -hard for NP, for every  $L \in \text{NP}$ , it is easy to describe a sparse set  $S'$  such that  $S' \leq_m^p S$  and  $L - S'$  is not  $\leq_m^p$ -hard for NP. Let  $f \in \text{PF}$  be the one-one function that reduces  $L$  to SAT. Then  $S' = \{x \mid f(x) \in S\}$  is sparse and  $L - S'$  reduces to  $\text{SAT} - S$  via  $f$ . So  $L - S'$  cannot be  $\leq_m^p$ -hard for NP.

**4. Immunity and Closeness.** Agrawal [Agr02] demonstrated that pseudorandom generators can be used to prove structural theorems on complete degrees of NP. Here we build on his results and show that hypotheses about pseudorandom generators and secure one-way permutations answer the longstanding open question of whether NP-complete sets can be immune. Also, using these hypothesis we show that no NP-complete set is quasipolynomial-close to P.

It is well-known that no EXP-complete set is p-immune. To see this, consider  $L \in \text{EXP}$  that is  $\leq_m^p$ -complete. Then  $0^* \leq_m^p L$  via some length-increasing reduction  $f$ . Since  $f$  is length-increasing,  $\{f(0^n) \mid n \geq 0\}$  is an infinite subset of  $L$ . However, while for any EXP-complete set  $L$  and any  $A \in \text{EXP}$ , there is a length-increasing reduction from  $A$  to  $L$  [BH77], this is not known to hold for NP.

We begin with the following definitions. In particular it is important to distinguish pseudorandom generators, as defined by Nisan and Wigderson [NW94], for derandomization purposes, from cryptographic pseudorandom generators [Yao82, BM84].

**DEFINITION 4.1.** A function  $G = \{G_n\}_n$ ,  $G_n : \Sigma^n \mapsto \Sigma^{m(n)}$  is an  $s(n)$ -secure cryptographic pseudo-random generator (crypto-prg in short) if  $G$  is computable in polynomial time in the input length,  $m(n) > n$ , for every  $\delta(\cdot)$  such that  $\delta(n) < 1$ , for every  $t(\cdot)$  such that  $t(n) \leq \delta(n) \cdot s(n)$ , and for every circuit  $C$  of size  $t(n)$ , for all sufficiently large  $n$ ,

$$\left| \Pr_{x \in \Sigma^{m(n)}} [C(x) = 1] - \Pr_{y \in \Sigma^n} [C(G_n(y)) = 1] \right| \leq \delta(n).$$

**DEFINITION 4.2.** A function  $G = \{G_n\}_n$ ,  $G_n : \Sigma^l \mapsto \Sigma^n$  is a pseudorandom generator (prg in short) if  $l = O(\log n)$ ,  $G$  is computable in time polynomial in  $n$ , and for any linear-size circuit  $C$ ,

$$\left| \Pr_{x \in \Sigma^n} [C(x) = 1] - \Pr_{y \in \Sigma^l} [C(G_n(y)) = 1] \right| \leq \frac{1}{n}.$$

**DEFINITION 4.3.** A function  $f = \{f_n\}_n$ ,  $f_n : \Sigma^{=n} \mapsto \Sigma^{=m(n)}$ , is  $s(n)$ -secure if for every  $\delta(\cdot)$  such that  $\delta(n) < 1$ , for every  $t(\cdot)$  such that  $t(n) \leq \delta(n) \cdot s(n)$ , and for every non-uniform circuit family  $\{C_n\}_n$  of size  $t(n)$ , for all sufficiently large  $n$ ,

$$\Pr_{x \in \Sigma^{=n}} [C_n(x) = f_n(x)] \leq \frac{1}{2^{m(n)}} + \delta(n).$$

**Hypothesis A.** Pseudorandom generators exist.

**Hypothesis B.** There is a secure one-way permutation. Technically, there is a permutation  $\pi \in \text{PF}$  and  $0 < \epsilon < 1$  such that  $\pi^{-1}$  is  $2^{n^\epsilon}$ -secure.

Hypothesis B implies the existence of cryptographic pseudorandom generators [Yao82]. Agrawal [Agr02] showed that if Hypothesis B holds, then every  $\leq_m^p$ -complete set for NP is hard also for one-one, length-increasing, non-uniform reductions. The following theorem is implicit in the proof of his result:

**THEOREM 4.4.** *If Hypotheses A and B hold, then every set  $A$  that is  $\leq_m^p$ -hard for NP is hard for NP under length-increasing reductions.*

By Theorem 4.4, Hypotheses A and B imply that for every NP-complete set  $A$ , there is a length-increasing reduction  $f$  from  $0^*$  to  $A$ . This immediately implies that the set

$$\{f(0^n) \mid n \geq 0\}$$

is an infinite subset of  $A$  that belongs to P, i.e.,  $A$  cannot be p-immune.

**THEOREM 4.5.** *If Hypotheses A and B hold, then no  $\leq_m^p$ -complete set for NP can be p-immune.*

We consider immunity with respect to classes that are larger than P. Similar questions have been studied for EXP. For example, Homer and Wang [HW94] showed that EXP-complete sets have dense UP subsets.

**THEOREM 4.6.** *Let  $\mathcal{C} \subseteq \text{NP}$  be a complexity class closed under  $\leq_m^p$ -reductions such that for some  $\epsilon > 0$ , there is a tally set  $T \in \mathcal{C}$  that is not in  $\text{DTIME}(2^{n^\epsilon})$ . Then no  $\leq_m^p$ -complete set for NP is  $\mathcal{C}$ -immune.*

**COROLLARY 4.7.** *If there is a tally set in UP that is not in  $\text{DTIME}(2^{n^\epsilon})$ , then no  $\leq_m^p$ -complete set for NP is UP-immune.*

*Proof.* [of Theorem 4.6] Let  $T$  be a tally set in  $\mathcal{C}$  that does not belong to  $\text{DTIME}(2^{n^\epsilon})$ . We will show that no NP-complete set is  $\mathcal{C}$ -immune.

Let  $L$  be an NP-complete set and let  $k > 0$  such that  $L \in \text{DTIME}(2^{n^k})$ . Let  $f$  be a  $\leq_m^p$ -reduction from  $T$  to  $L$ . We claim that the set

$$X = \{f(0^n) \mid 0^n \in T \text{ and } |f(0^n)| > n^{\epsilon/k}\}$$

is infinite. Assume otherwise: Then, for all but finitely many  $n$ ,  $0^n \in T \implies |f(0^n)| \leq n^{\epsilon/k}$ . Consider the following algorithm that accepts a finite variation of  $T$ : On input  $0^n$ , if  $|f(0^n)| \leq n^{\epsilon/k}$ , then accept  $0^n$  if and only if  $f(0^n) \in L$ . Otherwise, reject  $0^n$ . This algorithm takes time at most  $2^{|f(0^n)|^k} \leq 2^{(n^{\epsilon/k})^k} = 2^{n^\epsilon}$ . This contradicts the assumption that  $T \notin \text{DTIME}(2^{n^\epsilon})$ . Therefore,  $X$  is infinite. Also,  $X \subseteq f(T) \subseteq L$ . Now we will show that  $X \leq_m^p T$ . Since  $T$  belongs to  $\mathcal{C}$  and  $\mathcal{C}$  is closed under  $\leq_m^p$ -reductions, that will demonstrate that  $L$  is not  $\mathcal{C}$ -immune.

To see that  $X \leq_m^p T$ , we apply the following reduction: On input  $y$ ,  $|y| = m$ , determine whether  $f(0^i) = y$  for some  $i < m^{k/\epsilon}$ . If there is such an  $i$ , then output the first such  $0^i$ .

Otherwise,  $y \notin X$ . In this case, output some fixed string not in  $T$ . We need to show that  $y \in X$  if and only if the output of this reduction belongs to  $T$ . If  $y \in X$ , then there exists  $i$  such that  $i < m^{k/\epsilon}$ ,  $0^i \in T$ , and  $f(0^i) = y$ . Let  $0^{i_0}$  be the output of the reduction. In this case,  $y = f(0^i) = f(0^{i_0})$ . Now recall that  $f$  is a reduction from  $T$  to  $L$ . For this reason,  $0^i \in T$  if and only if  $0^{i_0} \in T$ . The converse case, that  $y \notin X$ , is straightforward.  $\square$

Agrawal [Agr02] defined a function  $g \in \text{PF}$  to be  $\gamma$ -sparsely many-one on  $S \subseteq \{0, 1\}^n$  if

$$\forall x \in S, \|g^{-1}(g(x)) \cap \{0, 1\}^n\| \leq \frac{2^n}{2^{n^\gamma}}.$$

Here  $g^{-1}(z) = \{x \mid g(x) = z\}$ . The function  $g$  is *sparsely many-one* on  $S \subseteq \{0, 1\}^n$  if it is  $\gamma$ -sparsely many-one on  $S \subseteq \{0, 1\}^n$  for some  $\gamma > 0$ .

Given a  $2^{n^\epsilon}$ -secure one-way permutation, Goldreich and Levin [GL89] construct a  $2^{n^\alpha}$ -secure crypto-prg,  $0 < \alpha < \epsilon$ . This crypto-prg  $G$  is defined only on strings of even length, i.e.,  $G$  is a partial function. However, Agrawal [Agr02] notes that  $G$  can be extended to be total, and the security remains the same. This crypto-prg has a nice property, namely it is a one-one function.

Let  $S$  be any set in NP and  $L$  be any NP-complete language. Let  $S' = G(S)$ . Since  $S'$  is in NP, there is a many-one reduction  $f$  from  $S'$  to  $L$ . Let  $h \stackrel{\text{def}}{=} f \circ G$ . Since  $G$  is one-one,  $h$  is a many-one reduction from  $S$  to  $L$ .

LEMMA 4.8 ([Agr02]). *For every  $n$ ,  $h \stackrel{\text{def}}{=} f \circ G$  is a  $\alpha/2$ -sparsely many-one on  $S \cap \Sigma^{=n}$ , where  $\alpha$  is the security parameter of  $G$ .*

LEMMA 4.9. *Let  $f$  be a  $\gamma$ -sparsely many-one function on  $S = 0^* \times \Sigma^* \cap \{0, 1\}^n$  for every  $n$ , and let  $l = n^{2/\gamma}$ . Then, for sufficiently large  $n$ ,*

$$\|\{w \in 0^n \times \Sigma^{=l} \mid |f(w)| > n\}\| \geq \frac{3}{4}2^l.$$

*Proof.* Let  $S_n = 0^n \times \Sigma^{=l}$ . Every string in  $S_n$  has length  $m = n + l$ . For every  $w \in S_n$ , there are at most  $\frac{2^m}{2^{m\tau}}$  strings of length  $m$  that can map to  $f(w)$ . Therefore,  $\|f(S_n)\| \geq 2^l / (\frac{2^m}{2^{m\tau}})$ . Taking  $l = n^{\frac{2}{\gamma}}$ , we obtain that at least  $\frac{3}{4}$  of the strings in  $S_n$  have image of length  $> n$ .  $\square$

THEOREM 4.10. *If Hypothesis B holds, then for every  $\epsilon > 0$ , no  $\leq_m^p$ -complete set for NP can be  $\text{DTIME}(2^{n^\epsilon})$ -immune.*

*Proof.* The hypothesis implies the existence of a  $2^{n^\epsilon}$ -secure one-way permutation. Let  $G$  be the  $2^{n^\alpha}$ -secure crypto-prg,  $0 < \alpha < \epsilon$ , constructed from this secure one-way function. Let  $S = 0^* \times \Sigma^*$ , and  $S' = G(S)$ . Since  $L$  is NP-complete  $S' \leq_m^p L$  via  $f$ . Thus  $S \leq_m^p L$  via  $h = f \circ G$ . By Lemma 4.8,  $h$  is  $\alpha/2$ -sparsely many-one on  $S \cap \Sigma^{=n}$  for every  $n$ . For any  $n$ , take  $l = n^{4/\alpha}$ . Then, by Lemma 4.9, we know that for large enough  $n$ , at least  $\frac{3}{4}$  of the strings in  $0^n \times \Sigma^{=l}$  map via  $h$  to a string of length  $> n$ .

Let  $k = \frac{4}{\epsilon\alpha}$ . Assume  $G$  maps strings of length  $n$  to strings of length  $n^r$ ,  $r > 0$ . It is well known that from  $G$  we can construct a crypto-prg  $G'$  that expands  $n$  bits to  $n^k$  bits [Gol01, page 115]. Thus for any string  $w$  of length  $n^\epsilon$ ,  $G'(w)$  is of length  $l = n^{4/\alpha}$ . Consider the following circuit that on input  $(0^n, y)$ ,  $|y| = l$  accepts if and only if  $|h(0^n, y)| > n$ . This circuit accepts at least  $\frac{3}{4}$  of the inputs  $(0^n, y)$ ,  $|y| = l$ , if the input is chosen according to uniform distribution. Therefore, there must be some  $w$ ,  $|w| = n^\epsilon$ , such that this circuit accepts  $G'(w)$ . Therefore, for this  $w$ ,  $|h(0^n, G'(w))| > n$ . Now, the following  $\text{DTIME}(2^{n^\epsilon})$ -algorithm outputs infinitely many strings of  $L$ :

Input  $0^n$   
 Let  $m = n^\epsilon$   
 for  $w \in \Sigma^{=m}$   
 If  $|h(0^n, G'(w))| > n$ , then output  $h(G'(w))$   
 $\square$

**4.1. Closeness.** In general, Yesha [Yes83] considered two sets  $A$  and  $B$  to be close if the census of their symmetric difference,  $A\Delta B$ , is a slowly increasing function. For example,  $A$  and  $B$  are  $p$ -close if there is a polynomial  $p$  such that for every  $n$ ,  $\|(A\Delta B)^{=n}\| \leq p(n)$ . Ogiwara [Ogi91] and Fu [Fu93] observed that if  $A$  is NP-complete, then  $A$  is not  $p$ -close to any set  $B \in \mathsf{P}$ , unless  $\mathsf{P} = \mathsf{NP}$ . Define  $A$  and  $B$  to be *quasipolynomial-close* if there exists a constant  $k$  such that for every  $n$ ,  $\|(A\Delta B)^{=n}\| \leq 2^{\log^k n}$ . We show that if Hypothesis B holds, then no NP-complete set is quasipolynomial-close to a set in  $\mathsf{P}$ . We show unconditionally that if  $L$  is paddable and quasipolynomial-close to a set in  $\mathsf{P}$ , then  $L$  belongs to  $\mathsf{BPP}$ . As a corollary, if Hypothesis A holds, then no paddable NP-complete set is quasipolynomial-close to a set in  $\mathsf{P}$ .

We recall the following definitions, and recall that all known, natural NP-complete sets are paddable [BH77]:

**DEFINITION 4.11.** *A set  $A$  is paddable if there exists  $p(\cdot, \cdot)$ , a polynomial-time computable, polynomial-time invertible (i.e., there is a  $g \in \mathsf{PF}$  such that for all  $x$  and  $y$ ,  $g(p(x, y)) = \langle x, y \rangle$ ) function, such that for all  $a$  and  $x$ ,*

$$a \in A \iff p(a, x) \in A.$$

Recall that a set  $A$  is  $p$ -isomorphic to  $B$  if there exists  $f$ , a polynomial-time computable, polynomial-time invertible permutation on  $\Sigma^*$ , such that  $A \leq_m^p B$  via  $f$ . Mahaney and Young [MY85] proved that two paddable sets are many-one equivalent if and only if they are  $p$ -isomorphic.

**THEOREM 4.12.** *If  $L$  is paddable and quasipolynomial-close to a set in  $\mathsf{P}$ , then  $L \in \mathsf{BPP}$ .*

*Proof.* Assume that  $L$  is a paddable set and there is a set  $B \in \mathsf{P}$  such that  $L$  is quasipolynomial-close to  $B$ . Let  $p(\cdot, \cdot)$  be a padding function for  $L$ . Given a string  $x$ ,  $|x| = n$ , consider the following set.

$$P_x = \{p(x, y) \mid |x| = |y|\}.$$

Let  $q$  be a polynomial such that all strings in  $P_x$  have length  $\leq q(n)$ . Let  $k$  be a constant such that  $\|(L\Delta B)^{\leq q(n)}\| \leq 2^{\log^k n}$ . Note that  $\|P_x\| = 2^n$ .

If  $x \in L$ , then  $P_x \subseteq L$ . Therefore, at least  $2^n - 2^{\log^k n}$  strings from  $P_x$  belong to  $B$ . On the other hand, if  $x \notin L$ , then  $P_x \cap L = \emptyset$ , and so at least  $2^n - 2^{\log^k n}$  strings from  $P_x$  are not in  $B$ . Therefore,

$$x \in L \Rightarrow \Pr_{y \in \Sigma^n} [p(x, y) \in B] \geq 1 - \frac{2^{\log^k n}}{2^n},$$

$$x \notin L \Rightarrow \Pr_{y \in \Sigma^n} [p(x, y) \in B] \leq \frac{2^{\log^k n}}{2^n}.$$

Therefore,  $L \in \mathsf{BPP}$ .  $\square$

**COROLLARY 4.13.** *If SAT is quasipolynomial-close to a set in  $\mathsf{P}$ , then  $\mathsf{NP} = \mathsf{RP}$ .*

This follows immediately from the result of Ko [Ko82] that  $\text{NP} \subseteq \text{BPP}$  implies  $\text{NP} = \text{RP}$ . Hypothesis A implies that  $\text{BPP} = \text{P}$ . Therefore, we have the following corollaries.

**COROLLARY 4.14.** *If Hypothesis A holds, then no paddable set  $L \notin \text{P}$  can be quasipolynomial-close to any set in  $\text{P}$ .*

**COROLLARY 4.15.** *If Hypothesis A holds, then no set  $p$ -isomorphic to  $\text{SAT}$  can be quasipolynomial-close to any set in  $\text{P}$ , unless  $\text{P} = \text{NP}$ .*

Next we are interested primarily in the following Theorems 4.16 and 4.18, and their immediate consequence, Corollary 4.19. Theorem 4.16 follows directly from the statement of Hypothesis B.

**THEOREM 4.16.** *Hypothesis B implies that*

$$\text{NP} \not\subseteq \bigcup_{k>0} \text{DTIME}(2^{\log^k n}).$$

*Proof.* Hypothesis B asserts the existence of a  $2^{n^\epsilon}$ -secure one-way permutation  $\pi$ , for some  $0 < \epsilon < 1$ . No  $2^{n^\epsilon}$ -size circuit can compute the inverse of  $\pi$ . So the set

$$B = \{\langle y, i \rangle \mid \text{ith bit of } \pi^{-1}(y) = 0\}$$

belongs to  $\text{NP}$  and cannot have a quasipolynomial-size family of circuits. However, if  $B \in \text{DTIME}(2^{\log^k n})$ , for some  $k > 0$ , then  $B$  has a family of circuits of size  $(2^{\log^k n})^2 < 2^{\log^{2k} n}$ , which is a contradiction.  $\square$

We require the following proposition, which follows from Homer and Longpré's study of Ogihara–Watanabe pruning [HL94].

**PROPOSITION 4.17.** *If there exists a set  $S$  that has a quasipolynomially-bounded census function and that is  $\leq_{\text{btt}}^p$ -hard for  $\text{NP}$ , then  $\text{NP} \subseteq \bigcup_{k>0} \text{DTIME}(2^{\log^k n})$ .*

**THEOREM 4.18.** *If  $\text{NP} \not\subseteq \bigcup_{k>0} \text{DTIME}(2^{\log^k n})$ , then no  $\text{NP}$ -complete set is quasipolynomial-close to a set in  $\text{P}$ .*

*Proof.* Assume there exists an  $\text{NP}$ -complete  $A$  that is quasipolynomial-close to some  $B \in \text{P}$ . Let  $S \stackrel{\text{def}}{=} A\Delta B$ . So  $S$  has a quasipolynomially-bounded census function.  $A \leq_{1\text{-tt}}^p S$  and therefore,  $S$  is  $\leq_{1\text{-tt}}^p$ -hard for  $\text{NP}$ . By Proposition 4.17,  $\text{NP} \subseteq \bigcup_{k>0} \text{DTIME}(2^{\log^k n})$ .  $\square$

As an immediate consequence, we have the following corollary, which has a stronger consequence than Corollary 4.15.

**COROLLARY 4.19.** *If Hypothesis B holds, then no  $\text{NP}$ -complete set is quasipolynomial-close to any set in  $\text{P}$ .*

It is interesting to note that Corollary 4.19 has a short proof that does not depend on Theorems 4.16 and 4.18. We present that now:

*Proof.* We begin as the proof of Theorem 4.16 begins: Hypothesis B asserts the existence of a  $2^{n^\epsilon}$ -secure one-way permutation  $\pi$ . No  $2^{n^\epsilon}$ -size circuit can compute the inverse of  $\pi$ . So the set  $B = \{\langle y, i \rangle \mid \text{ith bit of } \pi^{-1}(y) = 0\}$  belongs to  $\text{NP}$  and cannot have quasipolynomial-size family of circuits.

Let us assume that  $L$  is an  $\text{NP}$ -complete set such that there is some set  $S \in \text{P}$  and some  $k > 0$  such that for every  $n$ ,  $\|L\Delta S\| \leq 2^{\log^k n}$ . This implies that  $L \in \text{P}/(2^{\log^k n})$ , where the advice for any length  $n$  is the set of strings in  $L\Delta S$ . On an input  $x$ , accept  $x$  if and only if  $x \in S$  and  $x$  is not in the advice set, or  $x \notin S$  and  $x$  belongs to the advice set.

Therefore,  $L$  has a family of quasipolynomial-size circuits. Since  $L$  is  $\text{NP}$ -complete, it follows that every set in  $\text{NP}$  has quasipolynomial-size family of circuits. By the above discussion, this contradicts Hypothesis B.  $\square$

Finally, we state another consequence of Theorem 4.18, that has the same conclusion as does Corollary 4.19. Buhrman and Homer [BH92] proved that NP-complete sets do not have circuits of quasipolynomial size unless the exponential hierarchy collapses to  $\text{NEXP}^{\text{NP}}$ , the second level.

**COROLLARY 4.20.** *If the exponential hierarchy does not collapse to its second level, then no NP-complete set is quasipolynomial-close to any set in P.*

**5. Disjoint Pairs.** Recall that if  $\text{NP} \cap \text{coNP} \neq \text{P}$ , then there exist disjoint sets  $A$  and  $B$  in NP such that  $A \not\leq_T^p A \cup B$ . Our first result derives the same consequence under the assumption that  $\text{UEE} \neq \text{EE}$ .

**THEOREM 5.1.** *If  $\text{UEE} \neq \text{EE}$ , then there exist two disjoint sets  $A$  and  $B$  in UP such that  $A \not\leq_T^p A \cup B$ .*

*Proof.* Beigel et al. [BBFG91] showed that if  $\text{NEE} \neq \text{EE}$ , then there exists a language in  $\text{NP} - \text{P}$  for which search does not reduce to decision. Their proof also shows that if  $\text{UEE} \neq \text{EE}$ , then there exists a language  $S$  in  $\text{UP} - \text{P}$  for which search does not reduce to decision. Let  $M$  be an unambiguous Turing machine that accepts  $S$ , and for every word  $x \in S$ , let  $a_x$  be the unique accepting computation of  $M$  on  $x$ . Let  $p$  be a polynomial such that for all  $x \in S$ ,  $|a_x| = p(|x|)$ . Define

$$A = \{ \langle x, y \rangle \mid x \in S, |y| = p(|x|), \text{ and } y \leq a_x \}$$

and

$$B = \{ \langle x, y \rangle \mid x \in S, |y| = p(|x|), \text{ and } y > a_x \}.$$

Both  $A$  and  $B$  belong to UP and are disjoint. Let

$$A \cup B = S' = \{ \langle x, y \rangle \mid x \in S \text{ and } |y| = p(|x|) \}.$$

Note that  $S'$  is many-one reducible to  $S$ . Now assume  $A \leq_T^p S'$ . Since  $S'$  is many-one reducible to  $S$ , it follows that  $A \leq_T^p S$ . However, we can compute the witness  $a_x$  for  $x \in S$  by using a binary search algorithm with oracle  $A$ . Therefore, replacing  $A$  with  $S$ , we see that search reduces to decision for  $S$ , contradicting our choice of  $S$ .  $\square$

Next, we study the question of whether there exist two disjoint Turing complete sets for NP whose union is not Turing complete. We consider the following hypothesis.

**UP-machine hypothesis.** There is a UP-machine  $M$  that accepts  $0^*$  such that for some  $0 < \epsilon < 1$ , no  $2^{n^\epsilon}$  time-bounded machine can correctly compute infinitely many accepting computations of  $M$ .

The UP-machine hypothesis and its variant the NP-machine hypothesis (obtained by replacing the UP-machine with an NP-machine) have been studied previously and shown to have several believable consequences. For example, Pavan and Selman [PS02] showed that the UP-machine hypothesis implies the existence of a Turing complete language for NP that is not truth-table complete, and the NP-machine hypothesis implies the existence of a Turing complete language for NP that is not many-one complete. Hitchcock and Pavan [HP04] showed that the NP-machine hypothesis implies  $\text{AM} = \text{NP}$ ,  $\text{BPP}^{\text{NP}} = \text{P}^{\text{NP}}$ , and  $\text{NEXP} \not\subseteq \text{P/poly}$ . Hemaspaandra, Rothe and Wechsung [HRW97] showed several equivalent characterizations of variants of the NP and UP-machine hypotheses. Fenner et al. [FFNR96] studied the infinite-often version of the NP-machine hypothesis.

Hitchcock and Pavan showed that if  $\text{UE} \cap \text{coUE}$  is not included in  $\text{DTIME}(2^{2^{\epsilon n}})$  infinitely often, then the UP-machine hypothesis holds. They also related this hypothesis to

the question of  $UE \cap coUE$  having high circuit complexity: Let  $K$  be the standard complete set for EXP. If  $UE \cap coUE$  does not have  $K$ -oracle circuits of size  $2^{\epsilon n}$  infinitely often, then the UP-machine hypothesis holds. They also showed that these hypotheses are weaker than some hypotheses used in the context of uniform derandomization.

We show that if  $UP \cap coUP$  has bi-immune languages, then the UP-machine hypothesis holds.

**THEOREM 5.2.** *If there is a  $DTIME(2^{n^\epsilon})$ -bi-immune language in  $UP \cap coUP$ , then the UP-machine hypothesis is true.*

*Proof.* Let  $L \in UP \cap coUP$  be the  $DTIME(2^{n^\epsilon})$ -bi-immune set, and let  $N$  and  $N'$  be the UP machines for  $L$  and  $\bar{L}$ . Consider the following machine  $M$  that accepts  $0^*$ : On input  $0^n$ ,  $M$  guesses an accepting computation of  $N$  and of  $N'$  on  $0^n$ , and accept  $0^n$  if either guess is right. Note that for every  $0^n$ , exactly one of the guesses will be correct, and therefore,  $L(M) = 0^*$ . If there is a  $2^{n^\epsilon}$  time-bounded machine  $T$  that can correctly compute infinitely many accepting computation of  $M$ , then either  $X = \{0^i \mid T(0^i) \text{ outputs an accepting computation of } N\}$  or  $X' = \{0^i \mid T(0^i) \text{ outputs an accepting computation of } N'\}$  is an infinite subset of  $L$  or  $\bar{L}$ , contradicting the bi-immunity of  $L$ .  $\square$

**THEOREM 5.3.** *If the UP-machine hypothesis is true, then there exist two disjoint Turing complete sets for NP whose union is not Turing complete.*

*Proof.* Let  $a_n$  be the accepting computation of  $M$  on  $0^n$ . Let  $p(n)$  be the polynomial that bounds  $|a_n|$ . Note that a deterministic machine can verify in polynomial time whether a string of length  $p(n)$  is an accepting path of  $M$ . Consider the following sets:

$$A = \{\langle x, a_m + 1 \rangle \mid |x| = n, x \in SAT, m = (2n)^{1/\epsilon}\} \\ \oplus \{\langle 0^n, i \rangle \mid i \leq p(n), \text{ bit } i \text{ of } a_n = 1\},$$

and

$$B = \{\langle x, a_m - 1 \rangle \mid |x| = n, x \in SAT, m = (2n)^{1/\epsilon}\} \\ \oplus \{\langle 0^n, i \rangle \mid i \leq p(n), \text{ bit } i \text{ of } a_n = 0\}.$$

It is easy to see that both  $A$  and  $B$  are Turing-complete for NP. They can be made disjoint by choosing an appropriate pairing function. Note that

$$A \cup B = \{\langle x, a \rangle \mid |x| = n, x \in SAT, a = a_m - 1 \text{ or } \\ a_m + 1, m = (2n)^{1/\epsilon}\} \oplus \{\langle 0^n, i \rangle \mid i \leq p(n)\}.$$

Assume that  $A \cup B$  is Turing complete for NP. Since the set  $\{\langle 0^n, i \rangle \mid i \leq p(n)\}$  is in P, the following set is Turing complete:

$$C = \{\langle x, a \rangle \mid |x| = n, x \in SAT, a = a_m - 1 \text{ or } a_m + 1, \\ m = (2n)^{1/\epsilon}\}$$

Consider the set

$$S = \{\langle 0^n, i \rangle \mid \text{bit } i \text{ of } a_n = 1\}.$$

Since  $S \in NP$ ,  $S \leq_T^p C$  via some oracle Turing machine  $U$ .

We describe the following procedure  $\mathcal{A}$ :

1. input  $0^n$ .
2. Simulate  $U$  on strings  $\langle 0^n, i \rangle$ , where  $1 \leq i \leq p(n)$ .

3. Let  $q = \langle x, y \rangle$  be a query that is generated. If  $y \neq a_t + 1$  or  $y \neq a_t - 1$  for some  $t$ , then continue the simulation with answer “No”.
4. Else,  $q = \langle x, y \rangle$ ,  $|x| = t^\epsilon/2$  and  $y = a_t + 1$  or  $y = a_t - 1$ .
5. If  $t \geq n^\epsilon$ , then output “Unsuccessful”, print  $a_t$  and Halt.
6. Otherwise, check whether  $x \in \text{SAT}$ ; this takes at most  $2^{|x|} \leq 2^{n^{\epsilon^2}/2}$  time. Answer the query appropriately, and continue the simulation of  $U$ .

Now we consider two cases.

CLAIM 5.1. *If  $\mathcal{A}(0^n)$  does not output unsuccessful for infinitely many  $n$ , then there is a  $2^{n^\epsilon}$ -time bounded machine that correctly outputs infinitely many accepting computations of  $M$ .*

*Proof.* Assume  $\mathcal{A}(0^n)$  does not output unsuccessful. This implies that  $\mathcal{A}$  is able to decide membership of  $\langle 0^n, i \rangle$ ,  $1 \leq i \leq p(n)$ , in  $S$ . Therefore,  $\mathcal{A}$  can compute  $a_n$ . The most expensive step of the above procedure is Step 6, where  $\mathcal{A}$  decides the membership of  $x$  in SAT. However, this occurs only if  $|x| \leq n^{\epsilon^2}/2$ , and hence takes at most  $2^{n^{\epsilon^2}/2}$  time. Thus the total time is bounded by  $O(p(n) \times q(n) \times 2^{n^{\epsilon^2}/2})$ , where  $q(n)$  is the running time of  $U$  on  $\langle 0^n, i \rangle$ . Since  $\epsilon < 1$ , this is bounded by  $2^{n^\epsilon}$ .  $\square$

CLAIM 5.2. *If  $\mathcal{A}(0^n)$  outputs “Unsuccessful” for all but finitely many  $n$ , then there is a  $2^{n^\epsilon}$ -time bounded machine that outputs infinitely many accepting computations of  $M$ .*

*Proof.* If  $\mathcal{A}(0^n)$  is unsuccessful, then it outputs a string  $a_t$  such that  $t \geq n^\epsilon$ . Hence, if  $\mathcal{A}(0^n)$  is unsuccessful for all but finitely many strings, then for infinitely many  $t$  there exist an  $n$ , where  $n \leq t^{1/\epsilon}$ , and  $\mathcal{A}(0^n)$  outputs  $a_t$ . Thus the following procedure computes infinitely many accepting computations of  $M$ :

```

input  $0^t$ 
for  $i = 1$  to  $t^{1/\epsilon}$  do
  if  $\mathcal{A}(0^i)$  outputs  $a_t$ 
    output  $a_t$  and halt.
  endif
end for

```

Note that  $\mathcal{A}(0^i)$  runs in time  $O(p(i) \times q(i) \times 2^{i^{\epsilon^2}/2})$ . Thus the total running time of the above procedure is  $O(2^{t^\epsilon})$ .  $\square$

Claims 5.1 and 5.2 show that if  $C$  is Turing complete for NP, then there is a  $2^{n^\epsilon}$ -time bounded Turing machine that computes infinitely many accepting computations of  $M$ . This contradicts the UP-machine hypothesis, and therefore,  $A \cup B$  cannot be Turing complete for NP.  $\square$

**5.1. Many-One Complete Languages.** Here we consider the analogous questions for many-one reductions. We first show under two different hypotheses that there exist disjoint sets  $A$  and  $B$  in NP such that  $A \not\leq_m^p A \cup B$ . Also we study the question for NP-complete sets. One of our results will show a relation between our question and propositional proof systems. We refer the reader to Glaßer et al. [GSS04] for definitions about proof systems and reductions between disjoint NP-pairs.

THEOREM 5.4. *If  $P \neq \text{NP} \cap \text{coNP}$ , then there exist disjoint  $A, B \in \text{NP}$  such that*

1.  *$A$  and  $B$  are many-one equivalent, and*
2.  *$A \not\leq_m^p A \cup B$ .*

*Proof.* Let  $b \in \{0, 1\}$ , and let  $L \in \text{NP} \cap \text{coNP} - P$ . Define

$$A = \{bw \mid b = \chi_L(w)\},$$

and

$$B = \{bw \mid b \neq \chi_L(w)\}.$$

Both  $A$  and  $B$  belong to  $\text{NP} \cap \text{coNP} - \text{P}$ . Note that  $A \cup B = \{0, 1\} \circ \Sigma^*$ . However, note that  $A \leq_m^p B$  via  $f(bw) = \bar{b}w$ , and the same reduction reduces  $B$  to  $A$ . Also note that  $w \rightarrow 1w$  reduces  $L$  to  $A$ , and hence  $A$  cannot be in  $\text{P}$ . Therefore,  $A \not\leq_m^p A \cup B$ .  $\square$

**THEOREM 5.5.** *If  $\text{UE} \neq \text{E}$ , then there exist disjoint sets  $A$  and  $B$  in  $\text{NP}$  such that  $A \not\leq_m^p A \cup B$ .*

*Proof.* Hemaspaandra et al. [HNOS96] showed that if  $\text{NE} \neq \text{E}$ , then there exists a language  $S$  in  $\text{NP}$  for which search does not reduce to decision nonadaptively. Essentially the same proof shows that if  $\text{UE} \neq \text{E}$ , then there exists a language  $S$  in  $\text{UP}$  for which search does not reduce to decision nonadaptively. Since  $S \in \text{UP}$ , for each  $x \in S$ , there is a unique witness  $v_x$ , where  $|v_x| = p(|x|)$ , for some polynomial  $p$ . Define

$$A = \{\langle x, i \rangle \mid x \in S, i \leq p(|x|), \text{ and the } i\text{th bit of the witness } v_x \text{ of } x \text{ is } 0\}$$

and

$$B = \{\langle x, i \rangle \mid x \in S, i \leq p(|x|), \text{ and the } i\text{th bit of the witness } v_x \text{ of } x \text{ is } 1\}.$$

It is clear that both  $A$  and  $B$  are in  $\text{NP}$  and are disjoint. Then,

$$A \cup B = S' = \{\langle x, i \rangle \mid x \in S, i \leq p(|x|)\}.$$

Observe that  $S' \leq_m^p S$ . Assume  $A \leq_m^p S'$ ; then  $A \leq_m^p S$ . Therefore, we can compute the  $i$ th bit of the witness of  $x$  by making one query to  $S$ . This implies that search nonadaptively reduces to decision for  $S$ , which is a contradiction.  $\square$

Two disjoint sets  $A$  and  $B$  are  $\text{P}$ -separable if there is a set  $S \in \text{P}$  such that  $A \subseteq S \subseteq \bar{B}$ . Otherwise, they are  $\text{P}$ -inseparable. Let us say that  $(A, B)$  is a *disjoint NP-pair* if  $A$  and  $B$  are disjoint sets that belong to  $\text{NP}$ . If  $(A, B)$  is a disjoint NP-pair such that  $A$  and  $B$  are  $\text{P}$ -separable, then  $A \leq_m^p A \cup B$  follows easily: On input  $x$ , the reduction outputs  $x$ , if  $x \in S$ , and outputs some fixed string  $w \notin A \cup B$ , if  $x \notin S$ . This observation might lead one to conjecture that  $A \cup B$  is not  $\leq_m^p$ -complete, if  $A$  and  $B$  are disjoint,  $\text{P}$ -inseparable,  $\leq_m^p$ -complete  $\text{NP}$  sets. The following theorem shows that this would be false, assuming  $\text{P} \neq \text{UP}$ .

**THEOREM 5.6.** *If  $\text{P} \neq \text{UP}$ , then there exist disjoint  $\text{NP}$ -complete sets  $A$  and  $B$  such that*

1.  $(A, B)$  is  $\text{P}$ -inseparable and
2.  $A \cup B$  is many-one complete for  $\text{NP}$ .

*Proof.* Under the assumption that  $\text{P} \neq \text{UP}$ , Grollmann and Selman [GS88] constructed a  $\text{P}$ -inseparable disjoint NP-pair  $(A', B')$  such that  $A'$  and  $B'$  are  $\text{NP}$  complete. Let

$$A \stackrel{\text{def}}{=} 0A' \cup 1\text{SAT},$$

and

$$B \stackrel{\text{def}}{=} 0B'.$$

Therefore,  $A \cap B = \emptyset$ . Also,  $\text{SAT} \leq_m^p A \cup B$  via  $f(\phi) = 1\phi$ . Therefore,  $A \cup B$  is  $\text{NP}$  complete. If  $(A, B)$  is  $\text{P}$ -separable, then so is  $(A', B')$ .  $\square$

Similar arguments show that in Theorem 5.6, we can use hypothesis  $P \neq NP \cap \text{coNP}$  instead of  $P \neq UP$ : If  $P \neq NP \cap \text{coNP}$ , then there exist sets  $A$  and  $B$  with same properties as in Theorem 5.6.

We learn from the next theorem that if there exist disjoint NP-complete sets whose union is not NP-complete, then this happens already for paddable NP-complete sets.

**THEOREM 5.7.** *The following are equivalent:*

1. *There exists an NP-complete set  $A$  and a set  $B \in \text{NP}$  such that  $A \cap B = \emptyset$  and  $A \cup B$  is not NP-complete.*
2. *There exist disjoint, NP-complete sets  $A$  and  $B$  such that  $A \cup B$  is not NP-complete.*
3. *There exist paddable, disjoint, NP-complete sets  $A$  and  $B$  such that  $A \cup B$  is not NP-complete.*
4. *For every paddable NP-complete set  $A$ , there is a paddable NP-complete set  $B$  such that  $A \cap B = \emptyset$  and  $A \cup B$  is not NP-complete. Furthermore, there is a polynomial-time-computable permutation  $\pi$  on  $\Sigma^*$  such that*
  - (a) *for all  $x$ ,  $\pi(\pi(x)) = x$ , and*
  - (b)  *$A \leq_m^p B$  and  $B \leq_m^p A$ , both via  $\pi$ .*
5. *For every NP-complete set  $A$ , there is a set  $B \in \text{NP}$  such that  $A \cap B = \emptyset$  and  $A \cup B$  is not NP-complete.*

By Theorem 5.7, if there exist disjoint, NP-complete sets whose union is not complete, then there is a set  $B$  in NP that is disjoint from SAT such that  $\text{SAT} \cup B$  is not NP-complete. Moreover, in that case, there exists such a set  $B$  so that  $B$  is p-isomorphic to SAT. It is even the case that SAT and  $B$  are  $\leq_m^p$ -reducible to one another via the same polynomial-time computable permutation.

*Proof.* **1  $\Rightarrow$  2:** Let  $A' \stackrel{\text{def}}{=} 0A \cup 1B$  and  $B' \stackrel{\text{def}}{=} 1A \cup 0B$ . Since  $A$  is NP-complete, both sets  $A'$  and  $B'$  are NP-complete. However,  $A' \cup B' = \{0, 1\} \cdot (A \cup B)$ , and hence is not NP-complete.

**2  $\Rightarrow$  3:** Choose  $A$  and  $B$  according to item 2. Let  $A' \stackrel{\text{def}}{=} A \times \Sigma^*$  and  $B' \stackrel{\text{def}}{=} B \times \Sigma^*$ .  $A'$  and  $B'$  are disjoint, paddable, and NP-complete.  $A' \cup B' = (A \cup B) \times \Sigma^*$ . Hence  $A' \cup B' \leq_m^p A \cup B$  and therefore,  $A' \cup B'$  is not NP-complete.

**3  $\Rightarrow$  4:** Choose  $A$  and  $B$  according to item 3. We may assume that there exists a polynomial-time computable permutation  $\pi$  on  $\Sigma^*$  such that

- for all  $x$ ,  $\pi(\pi(x)) = x$ , and
- $A \leq_m^p B$  and  $B \leq_m^p A$ , both via  $\pi$ .

Otherwise, we use  $0A \cup 1B$  and  $1A \cup 0B$  instead of  $A$  and  $B$ ; and  $\pi$  is the permutation on  $\Sigma^*$  that flips the first bit.

Let  $A'$  be any paddable NP-complete set. So  $A'$  and  $A$  are paddable and many-one equivalent. Therefore,  $A'$  and  $A$  are p-isomorphic, i.e., there exists  $f$ , a polynomial-time computable, polynomial-time invertible permutation on  $\Sigma^*$ , such that  $A' \leq_m^p A$  via  $f$ .

Let  $B' \stackrel{\text{def}}{=} f^{-1}(B)$ .  $B' \leq_m^p B$  via  $f$  and therefore,  $B'$  and  $B$  are p-isomorphic. It follows that  $B'$  is paddable and NP-complete.  $A' \cap B' = \emptyset$ , since  $A \cap B = \emptyset$ . Moreover,  $A' \cup B' \leq_m^p A \cup B$  via  $f$  and hence,  $A' \cup B'$  is not NP-complete. Let  $\pi'(x) \stackrel{\text{def}}{=} f^{-1}(\pi(f(x)))$ . So  $\pi'$  is a polynomial-time computable permutation on  $\Sigma^*$ . For all  $x$ ,

$$\pi'(\pi'(x)) = f^{-1}(\pi(f(f^{-1}(\pi(f(x)))))) = x.$$

Moreover, for all  $x$ ,

$$x \in A' \iff f(x) \in A \iff \pi(f(x)) \in B \iff \pi'(x) \in B'.$$

Therefore,  $A' \leq_m^p B'$  via  $\pi'$ , and analogously,  $B' \leq_m^p A'$  via  $\pi'$ .

**4  $\Rightarrow$  1:** Follows immediately, since SAT is paddable and NP-complete.

**1**  $\Rightarrow$  **5**: Choose  $A$  and  $B$  according to item 1, and let  $A'$  be an arbitrary NP-complete set. Let  $f \in \text{PF}$  such that  $A' \leq_m^p A$  via  $f$ .  $B' \stackrel{\text{def}}{=} \{x \mid f(x) \in B\}$ . Clearly,  $B' \in \text{NP}$  and  $A' \cap B' = \emptyset$ , since  $A \cap B = \emptyset$ . For all  $x$ ,

$$x \in A' \cup B' \Leftrightarrow f(x) \in A \vee f(x) \in B \Leftrightarrow f(x) \in A \cup B.$$

So  $A' \cup B' \leq_m^p A \cup B$  via  $f$  and therefore,  $A' \cup B'$  is not NP-complete.

**5**  $\Rightarrow$  **1**: Trivial.  $\square$

Next we state relations between our question and propositional proof systems [CR79]. The recent paper of Glaßer et al. [GSSZ03] contains definitions of the relevant concepts: propositional proof systems (pps), optimal pps, for a pps  $f$ , the canonical disjoint NP-pair  $(\text{SAT}^*, \text{REF}_f)$  of  $f$ , and reductions between disjoint NP-pairs. If a propositional proof system  $f$  is optimal, then Razborov [Raz94] has shown that the canonical disjoint NP-pair of  $f$  is  $\leq_m^{pp}$ -complete. Therefore, it is natural to ask, for any proof system  $f$ , whether the union  $\text{SAT}^* \cup \text{REF}_f$  of the canonical pair is complete for NP. However, this always holds. It holds for trivial reasons, because SAT reduces to  $\text{SAT}^* \cup \text{REF}_f$  by mapping every  $x$  to  $(x, \epsilon)$ . Since  $x$  does not have a proof of size 0, we never map to  $\text{REF}_f$ . However,  $x \in \text{SAT} \Leftrightarrow (x, \epsilon) \in \text{SAT}^*$ . Nevertheless, it is interesting to inquire, as we do in the following theorem, whether some perturbation of the canonical proof system might yield disjoint sets in NP whose union is not complete.

**THEOREM 5.8.** *Assume  $\text{P} \neq \text{NP}$  and there exist disjoint sets  $A$  and  $B$  in NP such that  $A$  is NP-complete but  $A \cup B$  is not NP-complete. Then there exists a pps  $f$  and a set  $X \in \text{P}$  such that*

1.  $\text{SAT}^* \cap X$  is NP-complete and
2.  $(\text{SAT}^* \cap X) \cup (\text{REF}_f \cap X)$  is not NP-complete.

*Proof.* If  $\text{NP} = \text{coNP}$ , then  $\overline{\text{SAT}}$  has a polynomially bounded pps  $f$ . Let  $p$  be the bound and let  $X \stackrel{\text{def}}{=} \{(x, y) \mid y = 0^p(|x|)\}$ . Clearly,  $\text{SAT}^* \cap X$  is NP-complete. Observe that

$$(\text{SAT}^* \cap X) \cup (\text{REF}_f \cap X) = X.$$

Since the latter set is in P and  $\text{P} \neq \text{NP}$ , it cannot be NP-complete. So in this case we are done.

From now on, let us assume that  $\text{NP} \neq \text{coNP}$ . By Theorem 5.7, there exists  $B' \in \text{NP}$  such that  $B' \subseteq \overline{\text{SAT}}$  and  $\text{SAT} \cup B'$  is not NP-complete. Let  $C \in \text{P}$  and  $p$  be a polynomial such that for all  $x$ ,

$$x \in B' \iff \exists y \in \Sigma^{p(|x|)} [(x, y) \in C].$$

Choose a polynomial-time-computable, polynomial-time-invertible pairing function  $\langle \cdot, \cdot \rangle$  such that for all  $x$  and  $y$ ,  $|\langle x, y \rangle| = 2|xy|$ . Define the following pps:

$$f(z) \stackrel{\text{def}}{=} \begin{cases} x & \text{if } z = \langle x, y \rangle, |y| = p(|x|), \text{ and } (x, y) \in C \\ x & \text{if } z = \langle x, 0^{2^{p(|x|)}} \rangle \text{ and } x \in \overline{\text{SAT}} \\ \text{false} & \text{otherwise} \end{cases}$$

Observe that  $f$  is a pps. Define

$$X \stackrel{\text{def}}{=} \{(x, 0^m) \mid m = 2(|x| + p(|x|))\}.$$

$X \in \text{P}$ . Let  $\text{SAT}' \stackrel{\text{def}}{=} \text{SAT}^* \cap X$  and  $\text{REF}' \stackrel{\text{def}}{=} \text{REF}_f \cap X$ .  $\text{SAT}' \in \text{NP}$  and  $\text{REF}' \in \text{NP}$ . Moreover,  $\text{SAT}'$  is NP-complete.

It remains to show that  $\text{SAT}' \cup \text{REF}'$  is not NP-complete. Let  $\alpha$  be a fixed element in  $\overline{\text{SAT} \cup B'}$ . (Such an element exists, because otherwise  $\text{NP} = \text{coNP}$ .) We show  $\text{SAT}' \cup \text{REF}' \leq_m^p \text{SAT} \cup B'$  via the following reduction function:

$$h(x, y) \stackrel{\text{def}}{=} \begin{cases} x & \text{if } (x, y) \in X \\ \alpha & \text{otherwise} \end{cases}$$

Assume  $(x, y) \in \text{SAT}' \cup \text{REF}'$ . Hence  $(x, y) \in X$  and therefore,  $y = 0^{2(|x|+p(|x|))}$  and  $h(x, y) = x$ . If  $(x, y) \in \text{SAT}'$ , then  $h(x, y) = x \in \text{SAT}$ . If  $(x, y) \in \text{REF}'$ , then there exists  $z \in \Sigma^{\leq 2(|x|+p(|x|))}$  such that  $f(z) = x$ . By the definition of  $f$ , there exists  $z \in \Sigma^{2(|x|+p(|x|))}$  such that  $z = \langle x, y \rangle$ ,  $|y| = p(|x|)$ , and  $(x, y) \in C$ . Hence  $h(x, y) = x \in B'$ .

Now assume  $(x, y) \notin \text{SAT}' \cup \text{REF}'$ . If  $(x, y) \notin X$ , then  $h(x, y) = \alpha \notin \text{SAT} \cup B'$  and we are done. Otherwise,  $(x, y) \in X$ . First,  $h(x, y) = x \notin \text{SAT}$ , since  $(x, y) \notin \text{SAT}'$ . Second, if  $x \in B'$ , then there exists  $y \in \Sigma^{p(|x|)}$  such that  $(x, y) \in C$ . Therefore, if  $x \in B'$ , then there exists  $z \in \Sigma^{\leq 2(|x|+p(|x|))}$  such that  $f(z) = x$ . The latter is not possible, since  $(x, y) \notin \text{REF}'$ . It follows that  $h(x, y) = x \notin B'$ .

This shows  $\text{SAT}' \cup \text{REF}' \leq_m^p \text{SAT} \cup B'$  via  $h$ . Hence,  $\text{SAT}' \cup \text{REF}'$  is not NP-complete.

□

In Theorem 5.9 we show that if there are sets  $A$  and  $B$  belonging to NP such that  $A \cap B = \emptyset$  and  $A \cup B$  is not NP-complete, then  $(A, B)$  cannot be a  $\leq_{sm}^{pp}$ -complete disjoint NP-pair.

**THEOREM 5.9.** *If  $(A, B)$  is a  $\leq_{sm}^{pp}$ -complete disjoint NP-pair, then  $A$ ,  $B$ , and  $A \cup B$  are NP-complete.*

*Proof.* Since the disjoint NP-pair  $(\text{SAT}, \{z \wedge \bar{z}\}) \leq_{sm}^{pp}$ -reduces to  $(A, B)$ ,  $\text{SAT} \leq_m^p A$ , i.e.,  $A$  is NP-complete. Similarly,  $B$  is NP-complete as well. Assume that  $(\text{SAT}, \{z \wedge \bar{z}\}) \leq_{sm}^{pp}$ -reduces to  $(A, B)$  via some reduction function  $f$ . Let

$$f'(x) \stackrel{\text{def}}{=} \begin{cases} f(x) & \text{if } x \neq z \wedge \bar{z} \\ f(y \wedge z \wedge \bar{z}) & \text{if } x = z \wedge \bar{z} \end{cases}$$

We obtain  $f'(\text{SAT}) \subseteq A \cup B$  and  $f'(\overline{\text{SAT}}) \subseteq \overline{A \cup B}$ . Hence  $A \cup B$  is NP-complete. □

According to the comments after Theorem 5.6, the converse of Theorem 5.9 does not hold if either  $\text{P} \neq \text{UP}$  or  $\text{P} \neq \text{NP} \cap \text{coNP}$ . Since we know that there exists a  $\leq_{sm}^{pp}$ -complete disjoint NP-pair if and only if there is a  $\leq_m^{pp}$ -complete disjoint NP-pair [GSS04], we obtain the following corollary.

**COROLLARY 5.10.** *If  $\leq_m^{pp}$ -complete disjoint NP-pairs exist, then there is a  $\leq_m^{pp}$ -complete disjoint NP-pair such that both components and their union are NP-complete.*

**5.2. Relativizations.** We have been considering the following questions:

1. Do there exist disjoint sets  $A$  and  $B$  in NP such that both  $A$  and  $B$  are  $\leq_T^p$ -complete, but  $A \cup B$  is not  $\leq_T^p$ -complete?
2. Do there exist disjoint sets  $A$  and  $B$  in NP such that both  $A$  and  $B$  are NP-complete, but  $A \cup B$  is not NP-complete?

We observe here that there exist oracles relative to which both of these questions have both “yes” and “no” answers. This implies that resolving these questions would require nonrelativizable techniques.

**PROPOSITION 5.11.** *If the union of every two disjoint  $\leq_T^p$ -complete sets for NP is  $\leq_T^p$ -complete for NP, then  $\text{P} \neq \text{NP} \implies \text{NP} \neq \text{coNP}$ .*

*Proof.* Let us assume that  $\text{NP} = \text{coNP}$ . Then  $\text{SAT} \cup \overline{\text{SAT}} = \Sigma^*$ , which is  $\leq_T^p$ -complete if and only if  $\text{P} = \text{NP}$ . □

Therefore, relative to an oracle for which  $P \neq NP = \text{coNP}$  holds [BGS75], the answer to question (1) is “yes”. Also, it is obvious that relative to an oracle for which  $P = NP$ , the answer to this question is “no” [BGS75].

Now we consider question (2).

PROPOSITION 5.12. *If the union of every two disjoint NP-complete sets is NP-complete, then  $NP \neq \text{coNP}$ .*

Therefore, an oracle relative to which  $NP = \text{coNP}$  holds will answer “yes” to question (2). We learned already that if  $A$  and  $B$  are disjoint, NP-complete, P-separable sets, then  $A \cup B$  is NP-complete. Homer and Selman [HS92] construct an oracle relative to which all disjoint NP-pairs are P-separable, yet  $P \neq NP$ . Therefore, relative to this oracle, the answer to question (2) is “no.” Indeed, relative to this oracle, the answer to question (1) is “no” also.

**6. Acknowledgments.** The authors are appreciative of enlightening conversation with M. Agrawal. The authors thank H. Buhrman for informing them of his work on quasipolynomial density, and thank M. Ogihara for informing them of his results on polynomial closeness. Also, we received helpful suggestions from L. Hemaspaandra.

#### REFERENCES

- [Agr02] M. Agrawal. Pseudo-random generators and structure of complete degrees. In *Proceedings 17th IEEE Conference on Computational Complexity*, pages 139–147. IEEE Computer Society, 2002.
- [BBFG91] R. Beigel, M. Bellare, J. Feigenbaum, and S. Goldwasser. Languages that are easier than their proofs. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 19–28. IEEE Computer Society Press, 1991.
- [Ber76] L. Berman. On the structure of complete sets: Almost everywhere complexity and infinitely often speedup. *Proc. 17th IEEE Symp. on Foundations of Computing*, pages 76–80, 1976.
- [BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the  $P=NP$  problem. *SIAM Journal on Computing*, 4:431–442, 1975.
- [BH77] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–322, 1977.
- [BH92] H. Buhrman and S. Homer. Superpolynomial circuits, almost sparse oracles, and the exponential hierarchy. In *Foundations of Software Technology and Theoretical Computer Science, 12th Conference, New Delhi, India, December 18-20, 1992, Proceedings*, volume 652 of *Lecture Notes in Computer Science*, pages 116–127. Springer-Verlag, 1992.
- [BHT98] H. Buhrman, A. Hoene, and L. Torenvliet. Splittings, robustness, and structure of complete sets. *SIAM Journal on Computing*, 27(3):637–653, 1998.
- [BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(2):850–864, 1984.
- [CR79] S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36–50, 1979.
- [FFNR96] S. Fenner, L. Fortnow, A. Naik, and J. Rogers. On inverting onto functions. In *Proceedings 11th Conference on Computational Complexity*, pages 213–223. IEEE Computer Society Press, 1996.
- [FPS01] L. Fortnow, A. Pavan, and A. Selman. Distributionally hard languages. *Theory of Computing Systems*, 34:245–261, 2001.
- [Fu93] B. Fu. On lower bounds of the closeness between complexity classes. *Mathematical Systems Theory*, 26(2):187–202, 1993.
- [GL89] O. Goldreich and L. Levin. A hardcore predicate for all one-way functions. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.
- [Gol01] O. Goldreich. *Foundations of Cryptography—Volume 1*. Cambridge University Press, New York, 2001.
- [GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–335, 1988.
- [GSS04] C. Glaßer, A. Selman, and S. Sengupta. Reductions between disjoint NP-pairs. In *Proceedings 19th IEEE Conference on Computational Complexity*. IEEE Computer Society, 2004.
- [GSSZ03] C. Glaßer, A. Selman, S. Sengupta, and L. Zhang. Disjoint NP-pairs. In *Proceedings 18th IEEE Conference on Computational Complexity*. IEEE Computer Society, 2003.
- [HL94] S. Homer and L. Longpré. On reductions of np sets to sparse sets. *Journal of Computer and System Sciences*, 48(2):324–336, 1994.

- [HNOS96] E. Hemaspaandra, A. Naik, M. Ogiwara, and A. Selman. P-selective sets and reducing search to decision vs. self-reducibility. *Journal of Computer and System Sciences*, 53:194–209, 1996. Special Issue of papers selected from the Eighth Annual IEEE Conference on Structure in Complexity Theory.
- [HP04] J. Hitchcock and A. Pavan. Hardness hypotheses, derandomization, and circuit complexity. In *Foundations of Software Technology and Theoretical Computer Science*, number 3328 in LNCS, pages 184–197, 2004.
- [HRW97] L. Hemaspaandra, J. Rothe, and G. Wechsung. Easy sets and hard certificate schemes. *Acta Informatica*, 34:859–879, 97.
- [HS92] S. Homer and A. Selman. Oracles for structural properties: The isomorphism problem and public-key cryptography. *Journal of Computer and System Sciences*, 44(2):287–301, 1992.
- [HW94] S. Homer and J. Wang. Immunity of complete problems. *Information and Computation*, 110(1):119–129, 1994.
- [Ko82] Ker-I Ko. Some observations on probabilistic algorithms and NP-hard problems. *Information Processing Letters*, 14:39–43, 1982.
- [Mah82] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and Systems Sciences*, 25(2):130–143, 1982.
- [MY85] S. Mahaney and P. Young. Reductions among polynomial isomorphism types. *Theoretical Computer Science*, 39:207–224, 1985.
- [NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.
- [Ogi91] M. Ogiwara. On P-closeness of polynomial-time hard sets. manuscript, 1991.
- [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, 1991.
- [PS01] A. Pavan and A. Selman. Separation of NP-completeness notions. In *Proceedings 16th IEEE Conference on Computational Complexity*. IEEE Computer Society, 2001.
- [PS02] A. Pavan and A. Selman. Separation of NP-completeness notions. *SIAM Journal on Computing*, 31(3):906–918, 2002.
- [Raz94] A. Razborov. On provably disjoint NP-pairs. Technical Report TR94-006, Electronic Colloquium on Computational Complexity, 1994.
- [Sch86] U. Schöning. Complete sets and closeness to complexity classes. *Math Systems Theory*, 19:24–41, 1986.
- [Sel79] A. Selman. P-selective sets, tally languages, and the behavior of polynomial-time reducibilities on NP. *Mathematical Systems Theory*, 13:55–65, 1979.
- [Sel88] A. Selman. Natural self-reducible sets. *SIAM Journal of Computing*, 17(5):989–996, 1988.
- [Sho76] J. R. Shoenfield. Degrees of classes of RE sets. *Journal of Symbolic Logic*, 41(3):695–696, 1976.
- [TFL93] S. Tang, B. Fu, and T. Liu. Exponential-time and subexponential-time sets. *Theoretical Computer Science*, 115(2):371–381, 1993.
- [Tod91] S. Toda. On polynomial-time truth-table reducibilities of intractable sets to P-selective sets. *Mathematical Systems Theory*, 24:69–82, 1991.
- [Yao82] A. C. C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91. IEEE Computer Society Press, 1982.
- [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.