On Polynomial Time One-Truth-Table Reducibility to a Sparse Set*

OSAMU WATANABE

Department of Computer Science, Tokyo Institute of Technology, Meguro-ku Ookayama, Tokyo 152, Japan

Received November 12, 1987; revised March 13, 1989

In this paper, we measure "intractability" of complexity classes by considering polynomial time 1-truth-table reducibility (in short, \( \leq p_{1-it} \)-reducibility) to a sparse set. We mainly investigate nondeterministic complexity classes that are defined in relation to one-way functions: UP, FewP, UBPP, and \#P. We show that if UP (resp., UBPP and \#P) has a polynomial time unsolvable problem, then it indeed has a problem that is "intractable" not only by being polynomial time unsolvable, but also by being \( \leq p_{1-it} \)-reducible to no sparse set. As an immediate consequence of our observation, we can also prove that if R \( \neq \) NP (resp., P \( \neq \) FewP and \#P \( \neq \) \#P), then no NP-complete set is \( \leq p_{1-it} \)-reducible to a sparse set, and thus no NP-complete set has a p-close approximation; this provides a partial answer to a question asked by Schöning (Math. Systems Theory 19 (1986), 29-41).

1. INTRODUCTION

The intention of this paper is to measure "intractability" of complexity classes by considering polynomial time 1-truth-table reducibility (in short, \( \leq p_{1-it} \)-reducibility) to a sparse set. We mainly investigate complexity classes that are closely related to one-way functions.

For any reduction type \( r \), a set is regarded as "intractable" if it is \( \leq p_r \)-reducible to no sparse set. Note that this intractability notion is stronger than polynomial time noncomputability; i.e., a set that is \( \leq p \)-reducible to no sparse set is obviously not in P. Thus, for any complexity class \( \mathcal{C} \supseteq P \), it is natural to ask the following question.

Question. Suppose that P \( \subseteq \mathcal{C} \). Does \( \mathcal{C} \) have a set that is intractable not only by being not in P but also by being \( \leq p \)-reducible to no sparse set?

This question has been studied by several authors. In particular, interesting results have been obtained having to do with \( \leq p_m \)-reducibility and polynomial time complexity classes such as NP; for example, Mahaney [Ma82] proved that if

* This research was performed while the author was visiting the Department of Mathematics, University of California, Santa Barbara, and was supported in part by the National Science Foundation under Grants CCR-8611980. This is the revised version of the paper "On \( \leq p_{1-it} \)-Sparseness of Nondeterministic Complexity Classes," which was presented at the 15th International Colloquium on Automata, Languages and Programming.
ON $\leq_{1,tt}^P$-REDUCIBILITY TO A SPARSE SET 501

If $P \neq \text{NP}$, then every NP-complete set is $\leq_{m}^P$-reducible to no sparse set. Note that the notion of "polynomial time reducible to no sparse set" yields stronger intractability when we consider more general reducibilities than $\leq_{m}^P$-reducibility [BKSS]. However, only partial results are known for such general reducibilities. Yesha [Ye83] proved that if $P \neq \text{NP}$, then every co-NP-complete set is positive-$\leq_{bt}^P$-reducible to no sparse set; but the question is left open for nonpositive reducibilities. Ukkonen [Uk83] obtained a partial answer by considering tally sets instead of sparse sets; he showed that if $P \neq \text{NP}$, then every NP-complete set is $\leq_{bt}^P$-reducible to no tally set. In this paper we investigate the above question by considering $\leq_{1,tt}^P$-reducibility. From our observations, we can show, for example, that if $R \neq \text{NP}$, then every NP-complete set is $\leq_{1,tt}^P$-reducible to no sparse set.

(Note: Quite recently, Ogiwara and Watanabe [OW 90] proved that if $P \neq \text{NP}$, then no NP-complete set is $\leq_{bt}^P$-reducible to a sparse set.)

The concept of "$\leq_{1,tt}^P$-reducibility to no sparse set" not only generalizes "$\leq_{m}^P$-reducibility to no sparse set," but also yields a way to discuss the nonexistence of a certain type of "approximation algorithm." For any set $A$, we say that $A$ has a (polynomial time) p-close approximation [Sc86, Ye83] if there is a polynomial time algorithm $M$, which is called a p-close approximation of $A$, such that the symmetric difference between $A$ and $L(M)$ is sparse. It is easy to show that a set which is $\leq_{1,tt}^P$-reducible to no sparse set is $\leq_{1,tt}^P$-reducible to no set with a p-close approximation [Sc86]. In other words, if a set is not $\leq_{1,tt}^P$-reducible to a sparse set, it is intractable not only by not having a p-close approximation but also by not being reducible to a set with a p-close approximation.

In this paper, we mainly investigate nondeterministic complexity classes that characterize (polynomial time) one-way functions. Roughly speaking, a function is called "one-way" if the function itself is easy to compute, but its inverse is hard to compute. Recently, one-way functions have been studied because they are interesting from both theoretical and practical points of view; in particular, one-way functions play an important role in cryptography [GS84]. Note that there are several types of one-way functions. Here we study four types of one-way functions: strictly one-to-one one-way functions, strictly poly-to-one one-way functions, randomized one-way functions, and extensible one-way functions. Four complexity classes are investigated in relation to these types of one-way functions: UP[Va76], FewP [Al86], UBPP, and $\mathcal{U}$. 

The most important complexity issue concerning one-way functions is the difficulty of computing their inverses. The above four complexity classes characterize the computation necessary to invert each type of one-way function; intuitively, their intractability indicates the difficulty of inverting the corresponding type of one-way function. For example, one can show that a strictly one-to-one one-way function exists (i.e., some candidate for strictly one-to-one one-way functions is in fact polynomial time non-invertible) if and only if $P \subseteq \mathcal{UP}$. Then the question of interest is how difficult the class UP could be if $P \not\subseteq \mathcal{UP}$. We prove that UP (resp., UBPP and $\mathcal{U}$) contains a set $\leq_{1,tt}^P$-reducible to no sparse set if $P \not\subseteq \mathcal{UP}$ (resp., BPP $\not\subseteq \mathcal{UBPP}$ and $\mathcal{P} \not\subseteq \mathcal{UP}$). We also show a similar result for FewP.
The basic strategy used to prove our results is a variation of the "tree-pruning" methods, which are often used to obtain similar results. (The reader will find a good expository survey on these methods in [Ma86].) We generalize this strategy by considering a "prefix set" [Se88, Wa87] of nondeterministic computation, and by using the notion of "partial complement" [Wa90] instead of "complement" [Fo79] or "pseudo complement" [Ma82]. A conventional tree-pruning method does not work for nonpositive reducibilities such as \( \leq^p_{\text{it}} \)-reducibility. Here we use a property that is particular to \( \leq^p_{\text{it}} \)-reducibility and thereby prove the following lemma: for every set \( L \) in NP, if \( L \) is not in P, then \( PC-\text{Pre}(M) \) is \( \leq^p_{\text{it}} \)-reducible to no sparse set, where \( M \) is a polynomial time nondeterministic machine that accepts \( L \), and \( PC-\text{Pre}(M) \) is the partial complement of the prefix set for \( M \). Our results are proved by investigating the complexity of \( PC-\text{Pre}(M) \). For example, we have that \( L \in \text{UP} \) implies \( PC-\text{Pre}(M) \in \text{UP} \); thus, we can prove that if there exists a set \( L \) in \( \text{UP} - \text{P} \), then \( \text{UP} \) has a set, i.e., \( PC-\text{Pre}(M) \), that is \( \leq^p_{\text{it}} \)-reducible to no sparse set.

Since the complexity classes considered here are included in more general nondeterministic complexity classes such as NP, our observations also reveal structural properties of such classes. We show that either \( \text{P} \neq \text{FewP} \), \( \text{P} \neq \text{NP} \), or \( \text{R} \neq \text{NP} \) implies that no sparse \( \leq^p_{\text{it}} \)-hard set exists for NP. Thus, if \( \text{R} \neq \text{NP} \), then no \( \leq^p_{\text{it}} \)-complete set in NP has a p-close approximation; this is a partial answer to a question raised by Schöning [Sc86]. Our main lemma yields that if \( \text{P} \neq \text{NP} \), then no sparse \( \leq^p_{\text{it}} \)-hard set exists for the class \( \text{D}^p \).

The paper is organized as follows. The next section explains notation and notions that are used in this paper. In Section 3, we define four types of one-way functions and discuss their relation to the corresponding complexity classes. Our main lemma is proved in Section 4. Finally, in Section 5, we investigate "\( \leq^p_{\text{it}} \)-reducibility to a sparse set" for four complexity classes related to one-way functions and for more general nondeterministic complexity classes.

2. Preliminaries

In this paper, we use standard notions and notation in computational complexity theory (see, e.g., [BDG88]). Let \( \Sigma \) denote a finite alphabet that includes \( \{0, 1\} \). We use \( \epsilon \) to denote the null string. By a string we mean an element of \( \Sigma^* \). For any string \( x \), let \( |x| \) denote the length of \( x \). For any set of strings \( A \), we use \( A \leq^n \) and \( A \leq^m \) to denote \( \{x \in A : |x| \leq n\} \) and \( \{x \in A : |x| = n\} \) respectively. Let \( |A| \) denote the number of elements in \( A \). A set \( S \) is sparse if there is a polynomial \( p \) such that for every \( n \geq 0 \), \( |S \leq^n| \leq p(n) \). We assume some fixed total bijective pairing function that is polynomial time computable and invertible; let \( \lambda xy \cdot \langle x, y \rangle \) denote it. We use \( A \times B \) to denote the set \( \{ \langle a, b \rangle : a \in A \land b \in B \} \).

By a function we mean a partial function from \( \Sigma^* \) to \( \Sigma^* \) unless otherwise indicated. For any function \( f \), we use \( \text{Dom}(f) \) to denote the domain of \( f \) and \( \text{Range}(f) \) to denote the range of \( f \). For any string \( y \in \Sigma^* \), let \( f^{-1}(y) \) denote \( \{x : f(x) = y\} \).
A function $f$ is one-to-one if $\|f^{-1}(y)\| = 1$ for every $y \in \text{Range}(f)$; $f$ is const-to-one (resp., poly-to-one) if there exists a constant $c > 0$ (resp., a polynomial $p$) such that for every $y \in \text{Range}(f)$, $\|f^{-1}(y)\| \leq c$ (resp., $\|f^{-1}(y)\| \leq p(|y|)$). By an inverse of $f$ we mean a function defined on $\text{Range}(f)$ that maps every $y$ in $\text{Range}(f)$ to some element in $f^{-1}(y)$. Note that every one-to-one function has a unique inverse; when $f$ is one-to-one, we use $f^{-1}$ to denote the inverse of $f$.

Our basic computation model is the standard deterministic, randomized, and nondeterministic Turing machines. We use the symbol $M$ to denote those machines, and use $L(M)$ to denote the set of strings accepted by $M$. We assume a natural encoding of nondeterministic computation by $\{0, 1\}^*$ that satisfies several reasonable conditions. For example, we assume that the length of a string encoding one computation path of a polynomial time bounded nondeterministic machine is bounded by some polynomial; some deterministic machine can easily simulate a computation encoded by a given string; etc. In this paper, every randomized machine $M$ is assumed to have bounded error probability: that is, $M$ satisfies

$$(\exists \varepsilon > 0)(\forall x)\left[ \Pr\{M \text{ accepts } x\} > \frac{1}{2} \rightarrow \Pr\{M \text{ accepts } x\} > \frac{1}{2} + \varepsilon \right].$$

An unambiguous machine [Va76] is a nondeterministic machine that has at most one accepting path for every input; in general, we say that for any string $x$, $M$ on $x$ (i.e., $M$’s computation on $x$) is unambiguous if it has at most one accepting path, and for any set of strings $A$, $M$ on $A$ is unambiguous if $M$ is unambiguous on every $x \in A$. We extend this notion as follows [Al86]: for any function $t$, a $t(n)$-accepting path bounded machine is a nondeterministic machine that has at most $t(|x|)$ many accepting paths for every input $x$. Thus, an unambiguous machine is a 1-accepting path bounded machine. We also introduce another extension of “unambiguous computation.” For any nondeterministic machine $M$, any set $C$, and any input $x$, an accepting path of $M$ on $x$ w.r.t. $C$ is an accepting path $w$ of $M$ on $x$ such that $\langle x, w \rangle \in C$; if such $w$ exists, then we say that $M$ accepts $x$ w.r.t. $C$. Let $L(M; C)$ denote the set of strings accepted by $M$ w.r.t. $C$. Intuitively, $M$ is used to make a nondeterministic computation tree on a given input, and $C$ is used to determine which of the computation paths in the tree are accepting path. A machine $M$ is unambiguous w.r.t. $C$ if for every $x \in \Sigma^*$, $M$ on $x$ has at most one accepting path w.r.t. $C$.

Polynomial time bounded versions of the above machine models yield well-known complexity classes such as $P$, $BPP$, and $NP$. In particular, we will investigate the following language classes:

$$\text{UP} = \{ L(M): M \text{ is a polynomial time bounded unambiguous machine} \};$$

$$\text{FewP} = \{ L(M): M \text{ is a polynomial time and } t(n)\text{-accepting path bounded machine, for some polynomial } t \};$$

and

$$\text{UBPP} = \{ L(M; C): M \text{ is a polynomial time bounded machine that is unambiguous w.r.t. } C \in BPP \}.$$
We have the following relations.

**Proposition 2.1.** (1) $P \subseteq \text{UP} \subseteq \text{FewP} \subseteq \text{NP}$.

(2) $\text{UP} \subseteq \text{UBPP}, \text{BPP} \subseteq \text{UBPP}$, and $\text{UBPP} = \text{UP}^{\text{BPP}}$.

*Proof.* (1) Immediate from the definitions.

(2) It suffices to show that $\text{UP}^{\text{BPP}} \subseteq \text{UBPP}$, since the other relations are clear from the definitions. The proof follows from the facts that $\text{BPP} = \text{BPP}^{\text{BPP}}$ and that we can reduce a sequence of queries to one query by using nondeterministic guesses [Wr77]. The details are left to the reader.

Even and Yacobi [EY80] introduced the concept of "promise problem." A promise problem is a pair of sets $(Q, R)$, where $Q$ and $R$ are regarded as a promise and a property, respectively. A set $X$ is a solution of a promise problem $(Q, R)$ if $(\forall x \in Q)[x \in R \leftrightarrow x \in X]$. A promise problem $(Q, R)$ is polynomial time solvable if there exists a solution of $(Q, R)$ that is in $P$. In this paper we investigate only "unambiguous" promise problems. A promise problem $(Q, R)$ is unambiguous if there exist a solution $X$ in NP and a polynomial time nondeterministic machine $M$ accepting $X$ that is unambiguous on $Q$. We use $\mathcal{P}$ and $\mathcal{UP}$ to denote the class of polynomial time solvable promise problems and the class of unambiguous promise problems respectively.

The following type of promise problem $(Q, R)$ is unambiguous: $R$ is in NP and some polynomial time nondeterministic machine $M$ for $R$ is unambiguous on $Q$. (Note that $R$ is one of the solutions of $(Q, R)$.) One interesting unambiguous promise problem of this type is $(\text{1SAT}, \text{SAT})$ [VV86], where 1SAT is the set of Boolean formulas that have at most one satisfying assignment, and SAT is the set of satisfying Boolean formulas. It is clear that (1SAT, SAT) is unambiguous; the natural nondeterministic machine for SAT witnesses it. Furthermore it is easy to show that (1SAT, SAT) is the "hardest" among all unambiguous promise problems in the following sense.

**Proposition 2.2.** If $(\text{1SAT}, \text{SAT})$ is polynomial time solvable, then $\mathcal{P} = \mathcal{UP}$.

*Remark.* It is not known whether (1SAT, SAT) is hard for $\mathcal{UP}$ under natural reducibilities such as $\leq_{\text{P}}^\text{T}$-reducibility [GS84].

*Proof.* Consider any promise unambiguous problem $(Q, R)$. By definition, there exist a solution $X$ and a polynomial time nondeterministic machine $M$ accepting $X$ that is unambiguous on $Q$.

Since SAT is $\leq_{\text{P}}^\text{m}$-complete in NP, there is a $\leq_{\text{m}}^\text{P}$-reduction from $X$ to SAT. Recall Cook's way of constructing such a reduction (see, e.g., [GJ79]); a reduction $f$ is designed so that for each $x \in X$, a satisfying assignment of $f(x)$ encodes some accepting computation of $M$ on $x$. Such a reduction $f$ satisfies the following properties: (i) for every $x$, $x \in X \leftrightarrow f(x) \in \text{SAT}$, and (ii) for every $x \in Q$, $f(x) \in 1\text{SAT}$.
Hence, for every solution $Y$ of (1SAT, SAT), the set $\{x : f(x) \in Y\}$ is a solution of $(Q, R)$. Therefore, if (1SAT, SAT) has some solution in $P$, then so does $(Q, R)$.

There are several types of polynomial time reducibilities [LLS76]; we consider "$\leq_{1,tt}^P$-reducibility." A function is a polynomial time 1-truth-table function if it is a total and polynomial time computable function whose range is $\Sigma^* \times \{\text{id}, \neg\}$, where $\text{id}$ and $\neg$ denote the Boolean identity and the Boolean negation. For any pair of sets $A$ and $B$, $A$ is polynomial time 1-truth-table reducible to $B$ ($A \leq_{1,tt}^P B$) if there exists a polynomial time 1-truth-table function $f$ such that for every $u \in \Sigma^*$, $(f(u) = \langle v, x \rangle, u \in A \leftrightarrow x(b \in B) = \text{true}$.

A polynomial time 1-truth-table function witnessing $A \leq_{1,tt}^P B$ is called a $\leq_{1,tt}^P$-reduction from $A$ to $B$.

3. Classification of Polynomial Time One-Way Functions

Our main motivation for studying UP, FewP, UBPP, and $\#P$ is their relation to polynomial time one-way functions. Roughly speaking, a function is regarded as "polynomial time one-way" if it is one-to-one (or poly-to-one) and polynomial time computable, but its inverse is not polynomial time computable. Here we make this notion more precise by defining four classes of one-way functions, then we show their relation to the corresponding complexity classes.

A function $f$ is called honest if there exists a polynomial $p$ such that for all $x \in \text{Dom}(f)$, $p(|f(x)|) \geq |x|$. This notion is necessary to avoid the case where the polynomial time noninvertibility is trivial. A function $f$ is called polynomial time computable if there exist a deterministic Turing machine transducer $M$ and a polynomial $p$ such that

(i) $(\forall x \in \text{Dom}(f))[M \text{ outputs } f(x) \text{ within } p(|x|) \text{ steps }];$ and

(ii) $(\forall x \notin \text{Dom}(f))[M \text{ does not halt }].$

Although this is a standard definition, the following weaker notion is sometimes referred as "polynomial time computability": a function $f$ is pseudo polynomial time computable if it has a deterministic transducer $M$ that satisfies the above condition (i). Note the following properties.

**Proposition 3.1.** (1) A function is pseudo polynomial time computable if and only if it has a polynomial time computable extension.

(2) For every pseudo polynomial time computable function $f$, $f$ is polynomial time computable if and only if $\text{Dom}(f)$ is in $P$.

(The proof is straightforward and thus omitted.)

The first two types of one-way functions are "strictly one-to-one one-way" and "strictly poly-to-one one-way". A function $f$ is called a strictly one-to-one one-way (resp., strictly const-to-one one-way, strictly poly-to-one one-way) if it is honest, one-to-one (resp., const-to-one, poly-to-one), and polynomial time computable, but no
inverse of $f$ is polynomial time computable. These sorts of one-way functions have been studied by several authors and were shown to have a deep relation to the structure of complexity classes [Al86, KLD86, KMR88]. Their existence is characterized by the polynomial time computability of a corresponding complexity class.

**Proposition 3.2.**

1. $P \neq \text{UP}$ if and only if strictly one-to-one one-way functions exist [Va76, GS84, Ko85] if and only if strictly const-to-one one-way functions exist [Wa88].

2. $P \neq \text{FewP}$ if and only if strictly poly-to-one one-way functions exist [Ao86].

In the field of cryptography (e.g., [GS84]), the existence of one-way functions is assumed and, indeed, several candidates for one-way functions have been proposed. However, even if we assume that such functions are not polynomial time invertible, we have been unable to prove that they yield a "strictly one-to-one" one-way function. This is because such cryptographic one-way functions are pseudo polynomial time computable, and it is unknown whether they are polynomial time computable, or whether one can extend them to polynomial time computable functions while keeping them one-to-one (or even poly-to-one). For example, consider the following function $\text{ptimes}$:

$$\text{ptimes}(\langle p, q \rangle) = p \cdot q.$$ 

(Note: $\text{Dom}(\text{ptimes}) = \{\langle p, q \rangle : 0 < p \leq q \land p \text{ and } q \text{ are primes}\}$.) This function is honest, one-to-one, and pseudo polynomial time computable, and is conjectured to be polynomial time noninvertible. On the other hand, the problem of whether it is polynomial time computable is still open. Note that one can easily extend it to a total function, thereby making it polynomial time computable; nevertheless, we have been unable to extend it while keeping it one-to-one (or even poly-to-one).

(Note: In [GS88] the reader will find a more elaborate explanation of this issue with several interesting examples.)

Here we introduce two types of one-way functions, "randomized one-way" and "extensible one-way," that are more appropriate when studying the well-known candidates for cryptographic one-way functions.

A function $f$ is *randomized polynomial time computable* if there exist a randomized Turing transducer $M$, a polynomial $p$, and a constant $\epsilon < \frac{1}{2}$ such that

(i) $(\forall x \in \text{Dom}(f))[\Pr\{M \text{ outputs } f(x) \text{ within } p(|x|) \text{ steps}\} > 1 - \epsilon]$; and

(ii) $(\forall x \notin \text{Dom}(f))[\Pr\{M \text{ does not halt}\} > 1 - \epsilon]$.

A function is called *randomized one-way* if it is honest, one-to-one, randomized polynomial time computable, and its inverse is not randomized polynomial time computable. For example, $\text{ptimes}$ is honest, one-to-one, and randomized polynomial time computable, since its domain $\{\langle p, q \rangle : 0 < p \leq q \land p \text{ and } q \text{ are primes}\}$
is in BPP; hence, \( p\times \) is randomized one-way if its inverse is not randomized polynomial time computable. The existence of randomized one-way functions is characterized by the following relation between BPP and UBPP.

**Proposition 3.3.** \( \text{BPP} \neq \text{UBPP} \) if and only if randomized one-way functions exist.

**Proof.** (If part) Valiant proved [Va76] that is strictly one-to-one one-way functions exist, then \( P \neq UP \). Following a similar argument, we prove \( \text{BPP} \neq \text{UBPP} \) from the existence of randomized one-way functions. Let \( f \) be a randomized one-way function: i.e., \( f \) is honest, one-to-one, randomized polynomial time computable, and \( f^{-1} \) is not randomized polynomial time computable. Consider the set \( L = \{ (y, w) : w \text{ is a prefix of } f^{-1}(y) \} \). Note that \( L \) is in UBPP. We show that \( L \) is not in BPP; hence, \( L \) witnesses \( \text{BPP} \neq \text{UBPP} \). Suppose otherwise, i.e., \( L \) is in BPP. Note that \( f^{-1} \) is polynomial time computable relative to the set \( L \) and that \( P^{BPP} = \text{BPP} \). Hence, \( f^{-1} \) is randomized computable, a contradiction.

(Only-if part) The proof is similar to the one in [GS84, Ko85], where they constructed a strictly one-to-one one-way function assuming that \( P \neq UP \).  

A function \( f \) is called **extensible one-way** if it satisfies the following conditions:

(i) \( f \) is honest and one-to-one,

(ii) there exists an honest and polynomial time computable extension \( g \) of \( f \) such that for every \( y \in \text{Range}(f) \), \( g^{-1}(y) = \{ f^{-1}(y) \} \) (i.e., \( f \) has an honest and polynomial time computable extension that keeps the one-to-one-ness of \( f \) on \( \text{Dom}(f) \)), and

(iii) \( f^{-1} \) is not pseudo polynomial time computable.

Note that it follows from condition (ii) that \( f \) is pseudo polynomial time computable (see Proposition 3.1).

It is easy to see that most well-known candidates for cryptographic one-way functions satisfy (i) and (ii); hence, they are extensible one-way if their inverses are indeed as hard as expected. For example, consider the function \( p\times \) again. Define \( \text{times} \) by

\[
\text{times}(\langle n, m \rangle) = n \cdot m.
\]

(Note: \( \text{Dom}(\text{times}) = \{ \langle n, m \rangle : 0 < n \leq m \} \).) Then \( \text{times} \) is a natural extension of \( p\times \); furthermore it is an extension that witnesses condition (ii) for \( p\times \). Hence, \( p\times \) is extensible one-way if \( p\times^{-1} \) is not pseudo polynomial time computable.

Grollmann and Selman [GS84] considered a certain type of extensible one-way function in relation to public-key cryptography: they proved [GS88, Theorem 11] that if there exists an extensible one-way function \( f \) such that \( \text{Dom}(f) \in \text{NP} \), then one can design a public-key cryptosystem that cannot be cracked in polynomial time.
We have the following relation between extensible one-way functions and unambiguous promise problems.

**Proposition 3.4.** \( P \neq \mathcal{NP} \) if and only if extensible one-way functions exist.

**Proof.** (If part) The proof is similar to the one for Proposition 3.3. Let \( f \) be an extensible one-way function: that is, \( f \) satisfies conditions (i)-(iii). In particular, \( f \) has an extension satisfying (ii); let \( g \) be such an extension. Define a promise problem \((D, L)\) as follows:

\[
D = \{ \langle y, w \rangle : w \in \Sigma^* \land y \in \text{Range}(f) \},
\]

and

\[
L = \{ \langle y, w \rangle : w \text{ is a prefix of } g^{-1}(y) \}.
\]

We can easily define a polynomial time nondeterministic machine accepting \( L \) that is unambiguous on \( D \). Thus, \((D, L)\) is an unambiguous promise problem. (Note that \( L \) itself is a solution of \((D, L)\).)

Let \( X \) be any solution of \((D, L)\). It follows from the definition of "solution" that \((\forall \langle y, w \rangle \in D) [\langle y, w \rangle \in X \iff \langle y, w \rangle \in L]\); in other words, \((\forall y \in \text{Range}(f)) [\langle y, w \rangle \in X \iff w \text{ is a prefix of } f^{-1}(y)]\). Thus, \( f^{-1} \) is pseudo polynomial time computable relative to \( X \). This proves that no solution of \((D, L)\) is in \( P \), since \( f^{-1} \) is not pseudo polynomial time computable. Therefore, \((D, L)\) witnesses \( P \neq \mathcal{NP} \).

(Only-if part) The proof is clear from the above discussion and the one in [GS84, Ko85]; thus, it is omitted. \( \square \)

4. **Breadth First Search Algorithm**

In this section we construct a polynomial time deterministic algorithm for a given set in \( NP \) from a technical assumption; this algorithm will play an important role in later discussion.

We first define two concepts, "prefix set" (see, e.g., [Se88, Wa87]) and "partial complement" [Wa90], which are necessary to state our results, and which are key notions in this paper. For any polynomial time nondeterministic acceptor \( M \), the following sets \( \text{Pre}(M) \) and \( \text{PC-Pre}(M) \) are called a prefix set for \( M \) and its partial complement:

\[
\text{Pre}(M) = \{ \langle x, w \rangle : w \text{ is a prefix of an accepting path of } M \text{ on } x \},
\]

\[
\text{PC-Pre}(M) = \{ \langle x, w \rangle : x \in L(M) \land \langle x, w \rangle \notin \text{Pre}(M) \}.
\]

One might think that both \( \text{Pre}(M) \) and \( \text{PC-Pre}(M) \) have more information than \( L(M) \), and thus they are harder than \( L(M) \). Indeed, we can easily show that \( L(M) \leq_p \text{Pre}(M) \) and that \( L(M) \leq_p \text{PC-Pre}(M) \).
In [Wa90] it is shown that if $\text{PC-Pre}(M)$ is $\leq_{\text{m}}^P$-reducible to a sparse set, then \( L(M) \) is in \( P \). Here we extend this by considering a more general reduction type.

**Lemma 4.1.** For every polynomial time nondeterministic acceptor \( M \), if $\text{PC-Pre}(M)$ is $\leq_{\text{t}}^P$-reducible to a sparse set, then \( L(M) \) is in \( P \).

**Proof.** Let \( M \) be any polynomial time nondeterministic acceptor. Assume that $\text{PC-Pre}(M)$ is $\leq_{\text{t}}^P$-reducible to a sparse set \( S \) via a $\leq_{\text{t}}^P$-reduction \( f \); from this assumption we will prove that \( L(M) \in P \), by giving a polynomial time deterministic algorithm that accepts \( L(M) \). We lose no generality by assuming that there is a fixed polynomial \( p_M \) such that every computation path of \( M \) on an input of length \( n \) is encoded by a string in \( \{0, 1\}^{p_M(n)} \).

We first give an outline of the algorithm we will construct. For a given input \( x \), the algorithm searches for an accepting path of \( M \) on \( x \); this search is done in a “breadth first” manner. That is, the algorithm has the following outline.

```plaintext
program Beam Search;
input(x); n ← |x|;
W ← {ε}; (recall that ε denotes the null string)
repeat \( p_M(n) \) times do {
  V ← \{w0, w1 : w ∈ W\};
  (⋆) W ← choose some elements from \( V \) so that \( W \) keeps
    at least one prefix of an accepting path if one exists in \( V \);
  if \( W \) has an accepting path of \( M \) on \( x \) then ACCEPT else REJECT
end.
```

The correctness of this outline, i.e., the algorithm accepts \( x \) iff \( x \in L \), is clear. Note that \( V \) keeps a prefix of an accepting path if the previous \( W \)—the set \( W \) obtained in the previous iteration of the loop—has one. However, we cannot simply do \( "W ← V" \) at (⋆): if we did, \( W \) would grow exponentially. Hence, the algorithm must select some elements from \( V \) without losing all of the accepting paths.

Consider the selection strategy in more detail. The following property is clear from the assumption that \( f \) is a reduction from $\text{PC-Pre}(M)$ to \( S \).

**Claim 1.** Let \( x \) be any string. For every \( w \) and \( w' \), if \( f(⟨x, w⟩) = f(⟨x, w'⟩) \), then we have either (i) both of \( w \) and \( w' \) are prefixes of accepting paths on \( x \) or (ii) neither of them is a prefix of an accepting path on \( x \).

For any \( x \) and \( V \), define \( U \) by

\[
U = \{w \in V : \text{no } w' < w \text{ exists in } V \text{ such that } f(⟨x, w⟩) = f(⟨x, w'⟩)\},
\]

where “<” is the lexicographic ordering of \( \Sigma^* \). In other words, \( U \) is the set of the lexicographicaly first elements of \( V \) that have different images under the function \( \lambda w . f(⟨x, w⟩) \). Claim 1 guarantees that \( U \) inherits at least one prefix of an accepting path from \( V \). Thus, implementing (⋆) by “\( W ← U \)” we still get a correct
algorithm. However, this is not a polynomial time algorithm in general, because
$W$ may grow exponentially. From our assumption, we will be able to define a
polynomial, say $q$, such that if $\|U\|$ exceeds $q(n)$, then one can either (i) choose
the first $q(n)$ elements of $U$ for $W$ without losing all of the prefixes of the accepting
paths in $U$, or (ii) conclude that $x \in L$.

Let us present the algorithm precisely. We first define several polynomials. Let $p_f$
be a polynomial time bound for computing $f$; note that $|f(u)| \leq p_f(|u|)$ for every
$u \in \Sigma^*$. Let $q_\delta$ be a polynomial such that $\|S^{\leq n}\| \leq q_\delta(n)$ for every $n \geq 0$. Without
loss of generality, we can assume that there exists a constant $c_0$ such that
$(\forall x, w)[|w| \leq p_M(|x|) \rightarrow |\langle x, w \rangle| \leq c_0 p_M(|x|)]$. Define a polynomial $t$ by
t(n) = q_\delta \circ p_f(c_0 p_M(n))$. Now our algorithm is as follows.

program Beam Search;
  input(x); n + - |x|;
  $W$ + - \{\epsilon\};
repeat $p_M(n)$ times do {
    $V$ + - \{w0, w1 : w \in $W$\};
    $U$ + - \{w \in $V$ : no w' < w exists in $V$ such that $f(\langle x, w \rangle) = f(\langle x, w' \rangle)$\};
    $U_{id}$ + - \{w \in $U$ : $f(\langle x, w \rangle) = \langle y, id \rangle$ for some $y$\};
    (***) $U_{-}$ + - \{w \in $U$ : $f(\langle x, w \rangle) = \langle y, \neg \rangle$ for some $y$\};
    if $\|U_{id}\| > t(n)$ then $U_{id}$ + - the first $t(n)$ + 1 elements from $U_{id}$;
    if $\|U_{-}\| > t(n)$ then ACCEPT (and halt);
    $W$ + - $U_{id} \cup U_{-}$;
if $W$ has an accepting path of $M$ on $x$ then ACCEPT else REJECT
end.

We first show the validity of the algorithm: namely, we prove that the selection
procedure used above is correct. For a given input $x$, consider any point when the
algorithm has just finished statement (**). Let $x, n, V, U, U_{id}$, and $U_{-}$ be fixed in
the following discussion. Define the following sets:

$A = \{w \in U : w$ is a prefix of an accepting path of $M$ on $x\}$;

$R = \{w \in U : w$ is not a prefix of an accepting path of $M$ on $x\}$;

$U_{id,A} = U_{id} \cap A$; $U_{-,A} = U_{-} \cap A$; $U_{id,R} = U_{id} \cap R$; and $U_{-,R} = U_{-} \cap R$.

Define $f_x(w) = f(\langle x, w \rangle)$. Note that $f_x$ is one-to-one on $U$ (from the definition of
$U$) and thus one-to-one on every subset of $U$. Then the correctness of the selection
procedure immediately follows from the discussion above and Claim 2(c) and (d)
below.

Claim 2. (a) If $x \in L$, then $\|U_{-,A}\| \leq t(n)$ and $\|U_{id,R}\| \leq t(n)$.
(b) If $x \notin L$, then $U_{-,A} = \emptyset$ and $\|U_{-,R}\| \leq t(n)$.
(c) For any $U' \subseteq U_{id}$, if $x \in L$ and $\|U'\| > t(n)$, then $U'$ has an element in $A$,
i.e., $U'$ has a prefix of an accepting path.
(d) If $\|U_{-}\| > t(n)$, then $x \in L$. 
Proof of Claim 2. (a) Suppose that \( x \in L \). It follows from the definition of \( \text{PC-Pre}(M) \) that \( \langle x, w \rangle \notin \text{PC-Pre}(M) \) for every \( w \in U_{\neg A} (\subseteq A) \). Thus \( f_x(U_{\neg A}) \subseteq S \), because \( f \) is a \( \leq^p_{1\text{tt}} \)-reduction from \( \text{PC-Pre}(M) \) to \( S \). For every \( w \) in \( U_{\neg A} (\subseteq V) \), we have \( |w| \leq p_M(n) \), and thus \( |f_x(w)| \leq p_f(c_0 p_M(n)) \). Hence, \( f_x(U_{\neg A}) \subseteq S^{\leq p_f(c_0 p_M(n))} \), from which we have \( \|f_x(U_{\neg A})\| \leq q_s(p_f(c_0 p_M(n))) = t(n) \). This proves that \( \|U_{\neg A}\| \leq t(n) \), since \( f_x \) is one-to-one on \( U_{\neg A} \). A similar argument shows that \( \|U_{id,R}\| \leq t(n) \).

(b) Suppose that \( x \notin L \). It is clear that \( U_{\neg A} = \emptyset \), because no accepting path exists for \( x \). Following an argument similar to (a), we have \( f_x(U_{\neg A}) \subseteq S^{\leq p_f(c_0 p_M(n))} \), and thus, \( \|U_{\neg A}\| \leq t(n) \).

(c) Note that \( U_{id} = U_{id,A} \cup U_{id,R} \). Then it is clear from (a) that if \( U' \subseteq U_{id} \) and \( \|U'\| > t(n) \), then \( U' \cap U_{id,A} \neq \emptyset \), i.e., \( U' \) has an element in \( U_{id,A} (\subseteq A) \).

(d) Note that \( U_{\neg} = U_{\neg,A} \cup U_{\neg,R} \); then immediate from (b). \( \Box \) Claim 2

We conclude the proof by showing that the algorithm halts within polynomial time. Consider the execution of the algorithm on an input of length \( n \). Note that each set variable has no more than \( 4r(n) + 2 \) elements, where the length of each element is bounded by \( p_M(n) \). Thus, clearly the algorithm halts within polynomial time. \( \Box \)

We have proved that if \( \text{PC-Pre}(M) \leq^p_{1\text{tt}} S \) via \( f \), then the above beam-search algorithm accepts \( L \) correctly. However, even if \( f \) is not a \( \leq^p_{1\text{tt}} \)-reduction from \( \text{PC-Pre}(M) \) to \( S \), the algorithm may still accept some input strings correctly. Here we state a condition, for a given \( x \), such that the algorithm accepts \( x \) iff \( x \in L \).

Let \( M \) be any polynomial time nondeterministic acceptor. One can find all the information concerning \( M \)'s computation in \( \text{PC-Pre}(M) \). We define the subset of \( \text{PC-Pre}(M) \) that concerns the computation of \( M \) on a specific input: for any \( x \), define \( \text{PC-Pre}(M)[x] = \{ \langle x, w \rangle : \langle x, w \rangle \in \text{PC-Pre}(M) \} \). For any sets \( A, B, \) and \( C \), a polynomial time \( 1\text{-tt} \)-function \( f \) is called a partial \( \leq^p_{1\text{tt}} \)-reduction from \( A \) to \( B \) consistent on \( C \) if for every \( u \in C \) (let \( f(u) = \langle v, x \rangle \), \( u \in A \iff x(v \in B) = \text{true} \).

Lemma 4.2. Let \( M \) be any polynomial time nondeterministic acceptor. Let \( f \) be any polynomial time \( \text{1-truth-table} \) function, and let \( S \) be any sparse set. For every \( x \in \Sigma^* \), if \( f \) is a partial \( \leq^p_{1\text{tt}} \)-reduction from \( \text{PC-Pre}(M)[x] \) to \( S \) that is consistent on \( \{ x \} \times \Sigma^* \), then the beam-search algorithm in Lemma 4.1 yields a correct answer on input \( x \), i.e., it accepts \( x \) iff \( x \in L(M) \). (The proof is straightforward modification of the previous proof and thus omitted.)

5. \( \leq^p_{1\text{tt}} \)-Reducibility to a Sparse Set

We first investigate \( \leq^p_{1\text{tt}} \)-Reducibility to a sparse set" for complexity classes concerning one-way functions.
THEOREM 5.1. If \( P \neq UP \), then there exists a set in \( UP - P \) that is \( \leq_{1\text{-it}} \)-reducible to no sparse set.

Proof. The outline of the proof is almost the same as the one in [Wa90]. Let \( L \) be any set in \( UP - P \), and let \( M \) be a polynomial time unambiguous acceptor for \( L \). We show that \( PC-Pre(M) \) satisfies the theorem. It follows from Lemma 4.1 that \( PC-Pre(M) \) is \( \leq_{1\text{-it}} \)-reducible to no sparse set; hence, clearly \( PC-Pre(M) \) is not in \( P \). Thus, it suffices to show that \( PC-Pre(M) \in UP \). Define a set \( pc-Pre(M) \) as follows:

\[
\text{pc-Pre}(M) = \{ \langle x, w \rangle : \exists w' [w' \text{ is an accepting path of } M \text{ on } x \land w \text{ is not a prefix of } w'] \}.
\]

Noting that \( M \) is an unambiguous machine, it is easy to show that \( pc-Pre(M) \) is in \( UP \) and that \( PC-Pre(M) = pc-Pre(M) \). Hence, \( PC-Pre(M) \) is in \( UP \).  

We have similar theorems for \( UBPP \) and \( \#P \).

THEOREM 5.2. If \( BPP \neq UBPP \), then there exists a set in \( UBPP - BPP \) that is \( \leq_{1\text{-it}} \)-reducible to no sparse set.

Proof. Let \( L \) be any set in \( UBPP - BPP \); let \( M \) and \( C \) be a machine and a set witnessing \( L \in UBPP \) (see the definition of \( UBPP \)). Here we consider a randomised version of "prefix set" and "partial complement"; that is, we define the set \( PC-Pre(M; C) \) as follows:

\[
PC-Pre(M; C) = \{ \langle x, w \rangle : x \in L(M; C) \land w \text{ is not a prefix of an accepting path of } M \text{ on } x \text{ w.r.t. } C \}.
\]

Following an argument similar to Theorem 5.1, we can prove that \( PC-Pre(M; C) \in UBPP \). Furthermore, since \( L(M) \leq_{P} PC-Pre(M; C) \) and \( L(M) \notin BPP \), we have \( PC-Pre(M; C) \notin BPP \). Thus, it suffices to prove that \( PC-Pre(M; C) \) is not \( \leq_{1\text{-it}} \)-reducible to any sparse set.

By way of contradiction, suppose that \( PC-Pre(M; C) \) is \( \leq_{1\text{-it}} \)-reducible to a sparse set \( S \) via \( f \). We use the beam-search algorithm in Lemma 4.1 with a slight modification: i.e., change the last statement so that the algorithm accepts input \( x \) if and only if \( W \) has an accepting path of \( M \) on \( x \) w.r.t. \( C \). Then the same argument proves that this algorithm correctly accept \( L \). Note that the last statement is achieved by a polynomial time randomized computation; thus \( L \in BPP \). A contradiction.  

THEOREM 5.3. If \( \#P \neq \#P \), then there exists a promise problem in \( \#P - \#P \) for which no solution is \( \leq_{1\text{-it}} \)-reducible to a sparse set.

Proof. Let \( (D, L) \) be any promise problem in \( \#P - \#P \). By definition, we have a polynomial time nondeterministic machine \( M \) that accepts some solution \( X \)
of \((D, L)\) and that is unambiguous on \(D\). Consider a promise problem \((U, pc-Pre(M))\), where \(U = D \times \Sigma^*\) and \(pc-Pre(M)\) is the set defined for \(M\) as in the proof of Theorem 5.1. We prove that \((U, pc-Pre(M))\) satisfies the theorem.

The following claim shows that \((U, pc-Pre(M))\) is an unambiguous promise problem.

**Claim 1.** There exists a polynomial time nondeterministic machine accepting \(pc-Pre(M)\) that is unambiguous on \(U\).

**Proof of Claim 1.** Consider a polynomial time nondeterministic machine that accepts \(pc-Pre(M)\) in a usual way; it satisfies the claim. \(\Box\) Claim 1

Now it suffices to show that \((U, pc-Pre(M))\) has no solution that is \(\leq^P_{1\text{-it}}\)-reducible to a sparse set. (Note that this also implies \((U, pc-Pre(M))\) \(\notin \mathcal{P}\).) By way of contradiction, suppose that a solution \(X\) of \((U, pc-Pre(M))\) is \(\leq^P_{1\text{-it}}\)-reducible to a sparse set \(S\) via \(f\). Consider the beam-search algorithm defined for \(M, S\), and \(f\) in the proof of Lemma 4.1. Although we do not know if the algorithm accepts \(L(M)\) (since we do not know if \(PC-Pre(M) \leq^P_{1\text{-it}} S\) via \(f\) here), we do know that the algorithm is polynomial time bounded. Let \(Y\) be the set of strings accepted by the algorithm. The following claim shows that \(Y\), which is in \(\mathcal{P}\), is a solution of \((D, L)\), thereby leading to a contradiction. \(\Box\) Claim 2

**Claim 2.** The set \(Y\) is a solution of \((D, L)\).

**Proof of Claim 2.** Let any string \(x \in D\) be fixed. Consider sets \(PC-Pre(M)[x] = \{ \langle x, w \rangle : \langle x, w \rangle \in PC-Pre(M) \}\) and \(pc-Pre(M)[x] = \{ \langle x, w \rangle : \langle x, w \rangle \in pc-Pre(M) \}\). Then for every \(w \in \Sigma^*\), we have

\[
\langle x, w \rangle \in PC-Pre(M)[x] \iff \langle x, w \rangle \in pc-Pre(M)[x] \quad \text{(since } M \text{ on } x \text{ is unambiguous)}
\]

\[
\langle x, w \rangle \in X \quad \text{(since } X \text{ is a solution of } (U, pc-Pre(M)))
\]

\[
\alpha(v \in S) = \text{true}, \quad \text{where } f(\langle x, w \rangle) = \langle v, \alpha \rangle \quad \text{(since } X \leq^P_{1\text{-it}} S \text{ via } f). \quad \text{(since } \alpha \text{ has a solution of } (D, L)). \quad \Box \text{ Claim 2}
\]

The above theorems show that if a class \(\mathcal{C}\) is "intractable" from polynomial time computability, then it actually possesses a stronger sense of "intractability": e.g., if \(UP\) has a set not in \(\mathcal{P}\), then it indeed has a set that is not \(\leq^P_{1\text{-it}}\)-reducible to any sparse set. Unfortunately, a similar proof technique does not yield this type of result for \(FewP\); thus, it is left open whether \(\mathcal{P} \neq FewP\) implies \(FewP\) has a set
\leq_{1 \text{-} \text{uni}}$-reducible to no sparse set. On the other hand, we can prove that $P \neq \text{FewP}$ implies that $\text{NP}$ has this type of intractability.

**Theorem 5.4.** If $P \neq \text{FewP}$, then there exists a promise problem in $\text{NP}$ such that no solution of it is $\leq_{1 \text{-} \text{uni}}$-reducible to a sparse set.

**Proof.** From Theorem 5.3, it suffices to show that $P \neq \text{NP}$. Let $A$ be a set in $\text{FewP} - P$; let $q_A$ and $M_A$ be a polynomial and a $q_A(n)$-accepting path bounded nondeterministic acceptor for $A$. Define a set $L$ as follows:

$$L = \{ \langle x, k \rangle : \text{there are at least } k \text{ accepting paths of } M_A \text{ on input } x \}.$$  

It is clear that $L \in \text{NP}$; furthermore, it is easy to see that $L$ is accepted by some polynomial time nondeterministic machine $M$ satisfying

$$(\forall x, k)[\langle x, k \rangle \notin L \implies M \text{ on } \langle x, k \rangle \text{ is unambiguous}].$$

Define $D = \{ \langle x, k \rangle : M \text{ is unambiguous on } \langle x, k \rangle \}$. Then it is clear from the definition that $(D, L)$ is an unambiguous promise problem. We show that $(D, L)$ is not polynomial time solvable.

Assume to the contrary that $(D, L)$ has a solution $X$ in $P$. The following properties are immediate from the definition of $L$, the property of $M$, and the assumption that $X$ is a solution of $(D, L)$.

**Claim 1.**

(a) $(\forall x)[x \in A \leftrightarrow (\exists k : 1 \leq k \leq q_A(|x|))[\langle x, k \rangle \in L]].$

(b) $(\forall \langle x, k \rangle \in D)[\langle x, k \rangle \in L \leftrightarrow \langle x, k \rangle \in X].$

(c) $(\forall x)[\langle x, q_A(|x|) \rangle \in D] \Rightarrow (\forall x, k)[\langle x, k \rangle \notin L \implies \langle x, k \rangle \notin D].$

From these properties, it is easy to see that the following algorithm, which is clearly polynomial time bounded, accepts $A$; this is a contradiction.

**Program**

```
input(x); n \leftarrow |x|;
for k \leftarrow q_A(n) \text{ downto 1 do }
    \{ \text{if } \langle x, k \rangle \in X \text{ then ACCEPT (and halt)}; \}
    \text{REJECT}
end.
```

Immediate corollaries of our results provide some observations on the sparseness of $\leq_{1 \text{-} \text{uni}}$-hard sets for nondeterministic complexity classes.

**Corollary 5.5.** From each one of the following assumptions, it is provable that no sparse $\leq_{1 \text{-} \text{uni}}$-hard set exists for $\text{NP}$.

(a) $P \neq \text{UP}$ (i.e., a strictly one-to-one one-way function exists);
(b) \( P \neq \text{FewP} \) (i.e., a strictly poly-to-one one-way function exists); and
(c) \( P \neq \mathcal{U}P \) (i.e., an extensible one-way function exists).

Proof. Notice that UP is a subclass of NP and that every unambiguous promise problem has a solution in NP. Then the proof is immediate from Theorem 5.1, 5.3, and 5.4 (and Proposition 3.2 and 3.4). \( \square \)

Hence, any one of the above conditions (a), (b), and (c) implies that no \( \leq_{\text{p}} \)-complete set in NP is \( \leq_{\text{1-tt}} \)-reducible to a sparse set and thus implies that no \( \leq_{\text{m}} \)-complete set in NP has a p-close approximation (see the discussion in Section 1).

From the result by Valiant and Vazirani [VV86] we have that \( R \neq \text{NP} \) implies that (1SAT, SAT) has no polynomial time solution. Hence, we have the following corollary.

Corollary 5.6. If \( R \neq \text{NP} \), then NP has no sparse \( \leq_{\text{1-tt}} \)-hard set.

This contrasts with the fact that if \( R = \text{NP} \), then NP has a sparse \( \leq_{\text{1}} \)-hard set [Ad78].

For the randomized nondeterministic class \( \text{NP}^{\text{BPP}} \), we have a result similar to Corollary 5.5.

Corollary 5.7. If \( \text{BPP} \neq \text{UBPP} \) (i.e., a randomized one-way function exists), then no sparse \( \leq_{\text{1-tt}} \)-hard set exists for \( \text{NP}^{\text{BPP}} \).

Proof. Noting that \( \text{UBPP} \subseteq \text{NP}^{\text{BPP}} \), the proof is immediate from Theorem 5.2 (and Proposition 3.3). \( \square \)

The class \( \text{DP} \) [PY82] is the class of languages \( L \) such that \( L = A \cap B^c \) for some \( A \) and \( B \) in NP. From Lemma 4.1, we have the following observation on the sparseness of \( \leq_{\text{1-tt}} \)-hard sets for \( \text{DP} \).

Corollary 5.8. If \( P \neq \text{NP} \), then \( \text{DP} \) has no sparse \( \leq_{\text{1-tt}} \)-hard set.

Proof. We prove the contrapositive. Assume that \( \text{DP} \) has a sparse \( \leq_{\text{1-tt}} \)-hard set \( S \); i.e., every set in \( \text{DP} \) is \( \leq_{\text{1-tt}} \)-reducible to \( S \). Note that for every polynomial time nondeterministic acceptor \( M \), \( PC-\text{Pre}(M) \) is in \( \text{DP} \) and thus is \( \leq_{\text{1-tt}} \)-reducible to \( S \). Hence, from Lemma 4.1 we have \( P = \text{NP} \). \( \square \)

Acknowledgments

I express my deep appreciation to Mr. Richard Huff and the referee who gave me constructive comments for improving the readability of the manuscript. I also thank Professor Ronald Book and Professor Shouwen Tang for various helpful discussions.
REFERENCES


