On Sets with Efficient Implicit Membership Tests\textsuperscript{1}

\textit{Lane A. Hemachandra}\textsuperscript{2}

University of Rochester
Department of Computer Science
Rochester, NY 14627

\textit{Albrecht Hoene}

Technische Universität Berlin
Fachbereich Informatik
D-1000 Berlin 10

\textsuperscript{1}This report appears simultaneously as Forschungsberichte des Fachbereichs Informatik 89-22, Technische Universität Berlin.

\textsuperscript{2}Research supported in part by the National Science Foundation under grant CCR-8996198 and a Presidential Young Investigator Award.
Abstract

This paper completely characterizes the complexity of implicit membership testing in terms of the well-known complexity class OptP, optimization polynomial time, and concludes that many complex sets have polynomial-time implicit membership tests.

1 Introduction

Deterministic polynomial time, P, is one notion that has been proposed as loosely defining sets that are efficiently computable [Cob64,Edm65]. By definition, for sets in P one can quickly check whether a given element is a member. Unfortunately, not all sets of interest are in P.

For sets outside of P, it seems natural to ask which properties can be quickly computed, and, conversely, for properties slightly less revealing than membership, it seems natural to characterize the sets for which that property can be quickly computed. Indeed, the study of exactly which sets are simple under various operations other than membership testing is a newly emergent theme in theoretical computer science. Though it has long been known that many complex problems can be efficiently approximately solved, current research efforts show that fundamental algorithmic operations—data compression, perfect hashing, and enumeration—can be efficiently performed on many complex sets [Hem90, GHK,HHS89,HHSY]. The present paper shows that even sets of extremely high complexity may have polynomial-time algorithms for implicitly testing membership.

The recent work of Goldsmith, Hemachandra, Joseph, and Young [GHJY87] can be viewed from the perspective of this paradigm of finding efficient operations for complex sets. Goldsmith, Joseph, and Young defined and extensively studied the behavior of near-testable (NT) sets—those sets $L$ for which on input $x$ one can quickly compute which of (a) $(x \in L) \oplus (x_- \in L)$ or (b) NOT[$(x \in L) \oplus (x_- \in L)$] holds. [GHJY87] showed that NT is essentially the same as the class $\oplus P$, “parity polynomial time” [PZ83,GP86].

In this paper, we consider a property that in some sense combines aspects of P and NT. Fix a set $L$. It is clear that for each $x \neq \epsilon$ exactly two of the following four statements hold:

1. $x \in L$,

\[ \oplus \text{ represents "exclusive or," and } x_- \text{ represents the predecessor of } x \text{ in standard lexicographical order.} \]
2. \( x \notin L \),

3. \( (x \in L) \oplus (x_{-} \in L) \),

4. \( \text{NOT}[(x \in L) \oplus (x_{-} \in L)] \).

If we could determine in polynomial time which of 3/4 holds, \( L \) would be in \( \text{NT} \). If we could determine in polynomial time which of 1/2 holds, \( L \) would be in \( \text{P} \). Clearly, of 1/2 exactly one holds, and of 3/4 exactly one holds. Thus, even though we know that for each \( x \) exactly two of 1/2/3/4 hold, if we could compute which two hold in polynomial time then \( L \) would be in \( \text{P} \).

Nearly near-testable sets (NNT) allow us to capture a more modest amount of information about a given element \( x \). We define a set to be nearly near-testable if for each \( x \) we can find, in polynomial time, one of 1/2/3/4 that holds.

**Definition 1.1** A language \( L \) is in NNT if there is a polynomial-time computable function \( f \) such that for each \( x \) either:

- \( (f(x) = "x \in L") \) and \( (x \in L) \), OR
- \( (f(x) = "x \notin L") \) and \( (x \notin L) \), OR
- \( (f(x) = "(x \in L) \oplus (x_{-} \in L)") \) and \( ((x \in L) \oplus (x_{-} \in L)) \), OR
- \( (f(x) = "\text{NOT}[(x \in L) \oplus (x_{-} \in L)]") \) and \( (\text{NOT}[(x \in L) \oplus (x_{-} \in L)]) \).

When one nearly near-tests an element of a nearly near-testable set, one obtains partial information that does not in general suffice to immediately compute membership; nonetheless, with an exponential number of tests, one can (trivially) recover membership information. Thus, NNT sets have implicit membership tests. The information is there, but the cost of extraction is high. It follows immediately from the brute force testing just referred to that NNT is contained in \( \text{PSPACE} \).

Clearly, \( \text{NT} \subseteq \text{NNT} \). This paper shows that, just as NT has been shown to be related to \( \oplus \text{P} \) [GHJY87], so also is NNT related to the optimization complexity class \( \text{OptP} \) [Kre88, BJY89]. In particular, Theorem 1.5 says that NNT is the same (within the flexibility of \( \leq_{\oplus}^{\text{NPC}} \) reductions) as \( \oplus \text{P} \) altered by allowing an \( \text{OptP} \) function as a second argument to the underlying nondeterministic TM.\(^2\)

\(^2\) Throughout this paper, the maximum function is always applied to sets of strings (which are in fact integers encoded in binary), and returns the integer value that the lexicographically largest string encodes; we adopt the convention that \( \text{max}(\emptyset) = 0 \).
Definition 1.2 [Kre86]

1. An NP metric Turing Machine, $\hat{N}$, is a nondeterministic polynomial-time Turing machine such that every branch writes a binary number and accepts; and for $x \in \Sigma^*$ we write $\text{opt}\hat{N}(x)$ for the largest value on any branch of $\hat{N}$ on input $x$.

2. A function $f$ is in OptP (optimization polynomial time) if there is an NP metric Turing machine $\hat{N}$ such that $f(x) = \text{opt}\hat{N}(x)$ for all $x \in \Sigma^*$.

Definition 1.3 [LS86] $\text{count}_N(w_1) (\text{count}_N(w_1, w_2))$ represents the number of accepting paths of machine $N$ running on input $w_1$ (on input $w_1, w_2$).

Definition 1.4 $L$ is in $\oplus\text{OptP}$ if and only if there is a nondeterministic polynomial-time Turing machine $N$, and a polynomial $r(\cdot)$, and a function $f \in \text{OptP}$ such that:

$$x \in L \iff \text{count}_N(f(x)) \text{ is odd}.$$  

Theorem 1.5

1. $\text{NNT} \subseteq \oplus\text{OptP}$.

2. $\oplus\text{OptP} \leq^p_{m} \text{NNT}$; indeed, $\oplus\text{OptP} \leq^P_{1-1} \text{NNT}$. That is, for each language $L$ in $\oplus\text{OptP}$, there is a language $L'$ in NNT such that $L$ reduces to $L'$ via a one-to-one reduction computable in deterministic polynomial time.

This paper's proof of Theorem 1.5 substantially extends the techniques of [GJY87] and [GHJY87] by showing that one can construct a way for the maximization performed by the OptP functions within $\oplus\text{OptP}$ sets to be encoded in the instant jackpot options (options 1/2) of some nearly near-testable set.

Finally, by proving that with probability one relative to a random oracle $A$, $\text{NT}^A \neq \text{NNT}^A$, we suggest that NT may differ from NNT. Probability one separations of pairs of classes in this range of complexity are usually attempted via circuit techniques, or via the techniques, now believed to be invalid [Bei88], used in the Bennett-Gill probability one separation of $\oplus\text{P}$ and PP [BG81]. In contrast, we prove our new result by a novel and simple approach: we "reduce" the probability one separation of these two complex classes to the a well understood task: that of separating a parity-like language from P with probability one.
2 Results

This section shows that implicit membership testing is closely related to the optimization class OptP.

We start with a preliminary lemma that both shows the robustness of \(\oplus\text{OptP} - \oplus\text{OptP}\) can be defined in terms of OptP, \(\Delta^p_2\), or maximization—and also translates \(\oplus\text{OptP}\) into a form that will be used in the proof of Theorem 1.5. In the following, parts 2 and 3 can be shown equivalent using the notions developed in ([Kre88], see also [PY84]). The equivalence of parts 1 and 4 establishes the version of \(\oplus\text{OptP}\) to be used in the proof of Theorem 1.5. Just as parts 1 and 2 are related by showing that the "x" argument is superfluous, so also could one add to the lemma below new parts 3' and 4', by replacing "\(\text{count}_N(x, \ldots)\)" with "\(\text{count}_N(\ldots)\)."

Lemma 2.1 The following are equivalent:

1. \(L\) is in \(\oplus\text{OptP}\).
2. There is a nondeterministic polynomial-time Turing machine \(N\), and a polynomial \(r(\cdot)\), and a function \(f \in \text{OptP}\) such that:
   \[ x \in L \iff \text{count}_N(x, f(x)) \text{ is odd.} \]
3. There is a nondeterministic polynomial-time Turing machine \(N\), a polynomial \(r(\cdot)\), and a function \(f\) computable by a \(\text{P}^{\text{NP}}\) machine (i.e., in \(\text{F}^{\text{P}^{\text{NP}}}\), in the notation of [Kre88]) such that:
   \[ x \in L \iff \text{count}_N(x, f(x)) \text{ is odd.} \]
4. There is a nondeterministic polynomial-time Turing machine \(N\), a polynomial \(r(\cdot)\), and a polynomial-time computable predicate \(R(\cdot, \cdot)\), such that:
   \[ x \in L \iff \text{count}_N(x, \max_{|z|=r(|x|)} \{z \mid R(x, z)\}) \text{ is odd.} \]

We defer the proof of Lemma 2.1 until immediately after that of Theorem 1.5.

Theorem 1.5

1. \(\text{NNT} \subseteq \oplus\text{OptP}\).
2. \(\oplus\text{OptP} \leq^p_m \text{NNT};\) indeed, \(\oplus\text{OptP} \leq^p_{1-1} \text{NNT}\). That is, for each language \(L\) in \(\oplus\text{OptP}\), there is a language \(L'\) in \(\text{NNT}\) such that \(L\) reduces to \(L'\) via a one-to-one reduction computable in deterministic polynomial time.

Due to the fact that \(\oplus\text{OptP}\) (but not necessarily \(\text{NNT}\)) is closed downwards under \(\leq^p_{1-1}\),
we immediately obtain the following corollary.

**Corollary 2.2**

1. The downward closures of NNT and ⊕OptP under \( \leq_{m}^{p} \) are identical. That is, 
   \[
   \{ L \mid (\exists L' \in \text{NNT})[L \leq_{m}^{p} L'] \} = \{ L \mid (\exists L' \in \oplus\text{OptP})[L \leq_{m}^{p} L'] \}.
   \]

2. The downward closures of NNT and ⊕OptP under \( \leq_{1}^{p} \) are identical.

It follows immediately from Theorem 1.5, the obvious inclusions of Proposition 2.3, and Toda's [Tod89] recent results,\(^3\) that it is extremely unlikely that nearly near-testable sets have efficient membership tests.

**Proposition 2.3** \( \oplus P \subseteq \oplus \text{OptP} \) and \( NP \subseteq \oplus \text{OptP} \).

**Corollary 2.4**

1. If \( \text{NNT} = P \), then \( P = NP = PH = \oplus P \).

2. If \( \text{NNT} \subseteq \Sigma_{k}^{p} \), then \( \Sigma_{k+1}^{p} = PH \).

3. \( PH \subseteq BP \cdot NNT \).\(^4\)

**Proof of Theorem 1.5** Part (1) is straightforward.\(^5\)

Now, let us show that for each \( L \in \oplus \text{OptP} \), there is an \( L' \in \text{NNT} \) such that \( L \leq_{1}^{p} L' \). Consider an arbitrary set \( L \in \oplus \text{OptP} \). By Lemma 2.1, let \( N, r(\cdot), \) and \( R(\cdot, \cdot) \) be a machine, polynomial, and predicate for \( L \), in the sense of Lemma 2.1, part 4. W.l.o.g., for all \( x, m \) let \( N(x, m) \) not have any member of \( 0^* \) as an accepting computation path. W.l.o.g., let \( r(\cdot) \) be monotonically increasing. W.l.o.g., there is an integer \( k \geq 2 \) such that for all \( x, m \), all computation paths of \( N(x, m) \) are of length exactly \( |x|^k + k \), and all paths of this length are present.

We define a (nonstandard but) very simple pairing function. Let \( x, m, p \in \{ 0, 1 \}^* \), and let \( l \) be an integer greater than zero; define \( \text{tweak}_{l}(x, m, p) \) to be \( xmp \) (the concatenation, without any separation characters, of \( x, m, \) and \( p \)) if \( |m| = r(|x|) \) and \( |p| = |x|^l + l \), and let \( \text{tweak}_{l}(x, m, p) \) be undefined otherwise. \( \text{tweak}_{l}(x, m, p) \) is one-to-one everywhere it is not

---

\(^3\)Namely, that (a) if \( \oplus P \subseteq \Sigma_{k}^{p} \), then the polynomial hierarchy collapses to \( \Sigma_{k+1}^{p} \), and (b) \( PH \subseteq BP \cdot \oplus P \).

\(^4\)In fact, for the same reason, the stronger result holds that \( PH \subseteq BP \cdot NT \), where \( NT \) is defined in [GJY87]. The BP operator is defined in [Sch87].

\(^5\)That is via using the maximum to find the largest element less than the current element that yields absolute membership information, and then counting the parity of the number of changes in membership status between it and the input value.
undefined, and given a string $z$, we can determine in polynomial time whether $z$ is in the range of $\text{tweak}_{i}$, and, if so, what its inverse is.

We define the set $L'$ as follows, using $\text{lex}$ as a subscript to denote operations performed with respect to the standard lexicographical order:

If there are no $x$, $m$, $p$ such that $z = \text{tweak}_{k}(x, m, p)$, then $z \notin L'$. Otherwise, let $x$, $m$, $p$ be the unique strings such that $\text{tweak}_{k}(x, m, p) = z$. Let $z$ be in $L'$ if:

1. $p = 0|z|^{k+k}$ and $m = 0^{|z|}$, or
2. $p = 0|z|^{k+k}$ and $R(x, m)$, or
3. there are an odd number of strings $p'$ such that:
   
   (a) $p' \leq_{\text{lex}} p$, and
   (b) $p' >_{\text{lex}} 0|z|^{k+k}$, and
   (c) $N(x, \hat{m})$ has $p'$ as an accepting path, where $\hat{m}$ is
   \[ \max_{|j|=|m| \text{ and } j \leq_{\text{lex}} m} \{j \mid R(x, j)\} \]  

All strings not granted membership in $L'$ by the above rules are not members of $L'$.

We claim that $L' \in \text{NNT}$, and $L \leq_{1-1} L'$. Let us show that $L'$ is nearly near-testable by explicitly presenting the polynomial-time procedure for nearly near-testing $L'$ (see Definition 1.1).

Given an input $z$, check if there are $x$, $m$, and $p$, such that $\text{tweak}_{k}(x, m, p) = z$. If not, print "$z \notin L'$." If such $x$, $m$, and $p$ exist (and thus are, perforce, unique), then:

If $m = 0^{|z|}$ and $p = 0|z|^{k+k}$ then print "$z \in L'$";
else if $R(x, m)$ and $p = 0|z|^{k+k-1}$ then print "$z \in L'$" if $p$ is an accepting path of $N(x, m)$ and print "$z \not\in L'$" if $p$ is a rejecting path of $N(x, m)$;
else if $p$ is an accepting path of $N(x, m)$ and $R(x, m)$ then print "$(z \in L') \oplus (z_{-} \in L')$";
else print "NOT[$(z \in L') \oplus (z_{-} \in L')$]."

\[ ^6 \text{Recall footnote 2.} \]
It is easily seen (by looking at the definition of $L'$) that this polynomial-time procedure nearly near-tests $L'$. Finally, $x \in L$ if and only if $\text{tweak}_k(x, 1^{\lceil |x| \rceil}, 1^{\lceil |x| + k \rceil})$ is in $L'$; thus, $L \leq_{p-1}^p L'$; this is because $\text{tweak}_k(x, 1^{\lceil |x| \rceil}, 1^{\lceil |x| + k \rceil})$ counts the parity of the number of paths of $N$ when its first input is $x$ and its second input is the true maximum.

It is clear from the proof that $L$ reduces to $L'$ by a $\leq_{1-1}^p$ reduction that is trivially invertible, and whose range is in P (and in fact, can be made to be $\Sigma^*$ by squashing out the empty lengths from the range of the reduction of the given proof).

**Proof of Lemma 2.1** The equivalence of parts 2 and 3 follows immediately from Theorem 4 of [Kre86] (3 $\Rightarrow$ 2 is instant; 2 $\Rightarrow$ 3 by having $N$ assume the role of computing the $h$ function of Krentel's Theorem 4 [Kre86]). It is also clear that 4 $\Rightarrow$ 2, as OptP functions can easily find the maximum value on which a polynomial predicate is true, by having each path guess a value and if the predicate is true, print the value. And it is immediate that 1 $\Rightarrow$ 2. It is also clear that 2 $\Rightarrow$ 1. To see this, let $L_2$ be a language satisfying part 2, via OptP function $f_2$ and nondeterministic polynomial-time Turing machine $N_2$. Then $L \in \oplus \text{OptP}$ via OptP function $f_1$ and nondeterministic polynomial-time Turing machine $N_1$, where $f_1(x) = (x, f_2(x))$ and $N_1(y)$ starts by decoding its input into $y = (x, z)$ and simulates $N_2(x, z)$. The pairing function must be chosen with care, so as not to interfere with the optimization; the venerable function $(x, z) = x_10x_20...x_r1z_1...z_r$, is fine, where $x_i$ ($z_i$) is the $i$th bit of $x$ ($z$).

We now turn to showing that 2 $\Rightarrow$ 4. Let $L$ be an arbitrary set satisfying part 2 of Lemma 2.1.

Choose machines that certify this; in particular, we'll use $N_2$ to denote the machine $N$ of part 2 of Lemma 2.1, and we'll use $\hat{N}$ to denote the (metric) Turing machine (see Definition 1.2) for the OptP function $f$ of part 2 of Lemma 2.1. W.l.o.g., for some integer $k$, $\hat{N}$ has, on inputs of length $n$, exactly $2^{n^k + k}$ paths, each of length exactly $n^k + k$. We will now define $N_4$, $r_4$, and $R_4$, as in part 4 of the lemma now being proven, in order to show that $L$ satisfies part 4.

Set $r_4(n) = 2(n^k + k)$. Let the predicate $R_4(x, z)$ accept if and only if:

1. $|z| = 2(|x|^k + k)$ and,
2. if we view $z$ as $z = \text{concatenation}(z_\text{start}, z_\text{end})$, $|z_\text{start}| = |z_\text{end}|$, then path $z_\text{end}$ of the computation tree of $\hat{N}(x)$ prints the integer value that $z_\text{start}$, viewed as a binary string, encodes.

Let $N_4(x, y)$ reject if $|y| \neq 2(|x|^k + k)$. Otherwise, $N_4(x, y)$ computes $\hat{p}$, the value of the
first $|x|^k + k$ bits of $y$ viewed as an integer, and then simulates $N_2(x, \bar{p})$.

Why does this work? The first and second halves of $x$ hold, respectively, the values and paths of the OptP function. The maximization finds the largest value and the path on which it is obtained. $N_4$ starts by throwing away the superfluous path information (which, in allowing the max to find the value of the OptP function, has already served its purpose), and proceeds to exactly simulate the machine $N_2$ of part 2 of our lemma.

Finally, it should be noted that nearly near-testable sets are related to 2-for-1 P-cheatable sets [Bei87,GJY87]. It follows from the definitions that NNT is contained in the class of 2-for-1 P-cheatable sets. However, as an immediate corollary of the first part of Theorem 1.5 and the result of Beigel, Gasarch, Gill, and Owings [BGGO87] that there are 2-for-1 P-cheatable sets of arbitrarily high time complexity, we have:

**Corollary 2.5** NNT is a strict subset of the class of 2-for-1 P-cheatable sets.

Thus, $\text{NP} \cup \oplus \text{P} \subseteq \oplus \text{OptP} \subseteq \text{PSPACE} \cap 2\text{-for-1-P-cheatable}$.

This paper has studied the class NNT. However, the class NT has been extensively investigated in earlier papers [GJY87,GHJY87]. If NT = NNT, there would be no need for a separate study of NNT; and indeed, at first NNT seems very closely related to NT. In fact, if one looks not at the classes of polynomial-time near-testable and nearly near-testable sets, but rather at the classes of sets with exponential-time near-testing and nearly near-testing functions, it is immediate that they are the same, and both are equal to exponential time. Similar results hold for many other classes, such as for recursive near-testers and recursive nearly near-testers.

Nonetheless, we give evidence that NT and NNT are not the same.

One traditional way of opening the possibility that classes differ is to present a relativized world in which they do differ [BGS75]. Somewhat stronger evidence can be presented by showing that classes differ relative to almost every oracle ([BG81], see also [Cai89,Bab87,KMR89]), though even this level of evidence does not ensure that the classes are different in the unrelativized world [Kur83]. Our proof proceeds by “reducing” to a simpler task the problem of separating NT from NNT.

**Theorem 2.6** $\text{NT}^A \not\subseteq \text{NNT}^A$ with probability one relative to a random oracle $A$.

---

7The claim holds for either of the standard definitions of exponential time, $E = \cup_{c>0} \text{DTIME}[2^{cn}]$ and $\text{EXP} = \cup_{c>0} \text{DTIME}[2^n]$.

8To be conservative about such claims, one should only make the claim that conflicting relativizations show that a problem will not be resolved by relativizable techniques; then one can discuss in detail the extent to which present and possible future techniques are or are not relativizable [Har85].
Proof of Theorem 2.6  By $N^A$ ($N^A$), we mean the sets that have a near-testing (nearly near-testing) function computable in $P^A$. Consider the following nearly near-testing function, which will implicitly define the language, $L_A$, that it nearly near-tests.

$$f_A(x) = \begin{cases} 
\chi_A(x) & \text{if } x \in 0^* \\
(z \in L_A) \oplus (z^- \in L_A) & \text{if } x \not\in 0^* \text{ and } z \in A \\
\text{NOT}[(z \in L_A) \oplus (z^- \in L_A)] & \text{if } x \not\in 0^* \text{ and } z \not\in A 
\end{cases}$$

Note that (trivially) for all $A$, $L_A \in N^A$, since there is a nearly near-testing function (namely $f_A$).

Now let us ask how likely it is that, relative to random oracle $A$, the set $L_A$ is in $N^A$. Consider an oracle $A'$ such that $L_{A'} \in N^A$, and let the near-testing function be called $f'$ ($\in P^{A'}$). Clearly, $A'$ will have an odd number of strings of length $n$ if and only if $1^n \in L_{A'}$. (This is because, in the case where $0^n \not\in L_{A'}$, we'll have $1^n \in L_{A'}$ only if the second line of the definition of $f$ occurs an odd number of times; in the case where $0^n \in L_{A'}$, we'll have $1^n \in L_{A'}$ only if the second line of the definition of $f$ occurs an even number of times, but $0^n$ itself will contribute to the parity to make the parity odd.) However, $1^n$ is in $L_{A'}$ if and only if:

$$(0^{n+1} \in A') \oplus (f'(0^{n+1}) \text{ prints } "(0^{n+1} \in L_{A'}) \oplus (1^n \in L_{A'})")$$

since $f'$ is a near-tester. The test just given is a $P^{A'}$ test that computes the parity of the number of strings of a given length in $A'$. That is, we've shown that if $L_{A'} \in N^A$, then there is a polynomial-time deterministic Turing machine that on input $1^n$ computes of parity of the set $\{w \mid |w| = n \text{ and } w \in A\}$. By standard techniques [BG81], the class of such oracles has measure zero. Thus with probability one relative to a random oracle $A$, $N^A \neq N^A$ (indeed, $N^A \nsubseteq N^A$, since that fact that $N \subseteq N^A$ relativizes).

---

9There has been some confusion in this area recently. Beigel [Bei88] has noted that Bennett and Gill’s proof of [BG81, Theorem 3] is incorrect, and that the result itself may not be true. However, the still-correct techniques of [BG81], in particular their Lemma 1, easily suffice to show that for random oracle $A$, one cannot compute the parity function of $A$ of in polynomial time relative to $A$. 

---
3 Conclusion

This paper shows that NNT is essentially the same as $\oplus$OptP. Thus, the complexity of implicit membership testing is closely related to the complexity of optimization.

Acknowledgements

We are very grateful to Paul Young and Judith Goldsmith for pointing out that NNT sets are 2-for-1 P-cheatable, and to Richard Beigel and Steven Rudich for helpful conversations.

References


