# On Sparse Oracles Separating Feasible Complexity Classes

Juris Hartmanis*
Lane Hemachandra**

TR 85-707
October 1985

Department of Computer Science
Cornell University
Ithaca, NY 14853

* Research supported by NSF grant DCR-8301766.
** Research supported by a Fannie and John Hertz Foundation Fellowship and NSF grant DCR-8301766.

# On Sparse Oracles Separating Feasible Complexity Classes

Juris Hartmanis[*]
Lane Hemachandra[†]

Department of Computer Science
Cornell University

ABSTRACT

This note clarifies which oracles separate NP from P and which do not. In essence, we are changing our research paradigm from the study of which problems can be relativized in two conflicting ways to the study and characterization of the class of oracles achieving a specified relativization. Results of this type have the potential to yield deeper insights into the nature of relativization problems and focus our attention on new and interesting classes of languages.

A complete and transparent characterization of oracles that separate NP from P would resolve the long-standing P=?NP question. In this note, we settle a central case. We fully characterize the sparse oracles separating NP from P in worlds where P=NP. We display related results about coNP, E, NE, coNE, and PSPACE.

[*]Research supported by NSF grant DCR-8301766.

[†]Research supported by a Fannie and John Hertz Foundation Fellowship and NSF grant DCR-8301766.

# 1 Introduction and Overview

Structural questions about feasible complexity classes, such as P=?NP, are usually first analyzed in relativized worlds. Researchers prove that, in appropriately relativized worlds, both a statement and its converse can be made to hold. This is interpreted as strong evidence that current proof techniques lack the power to resolve the question.[1]

In this note, we propose a new approach to structural questions that is both richer and more challenging than simply finding oracles relativizing a question in conflicting ways. We suggest, instead, characterizing the class of oracles achieving a specified relativization of the question.

Consider the ubiquitous P=?NP question. It is well known that this problem can be relativized both ways [1]. A complete and transparent characterization of oracles collapsing NP into P would resolve the P=?NP question. Leaving this to the interested reader, we instead find a complete classification for a central case. In worlds where P = NP, we completely characterize the sparse[2] oracles separating NP from P.

**Theorem 1.1** *If* $\mathrm{P}^A = \mathrm{NP}^A$ *and* $S$ *is sparse, then* $[\mathrm{P}^{A,S} = \mathrm{NP}^{A,S} \iff S$ *is* $\mathrm{P}^{A,S}$*-printable*$]$.

Whimsically phrased, this theorem says that the ability of a set to fool the P mechanism while remaining faithful to the NP mechanism depends on the complexity of the set's internal information organization (and not, for example, on the classical complexity of the set).

Next we dissect the theorem, strengthening each direction. While doing this, we note that many complexity properties of sparse sets degenerate into equivalence in worlds where P = NP. Here we have equivalent characterizations via classical uniform complexity, non-uniform complexity, and time-bounded Kolmogorov complexity.

**Theorem 1.2**

1. *If* P = NP *and* $T$ *is self-encodable*[3], *then* $\mathrm{P}^T = \mathrm{NP}^T$.

[1] Some recent results present exceptions to this truism [5][7].

[2] A set $S$ is *sparse* iff $(\exists \text{ polynomial } p(\cdot))(\forall i)[|S \cap (\Sigma + \epsilon)^i| \leq p(i)]$.

[3] Self-encodability is discussed later in this paper. A set $S$ is self-encodable iff there are polynomial time machines $H$ and $J$ so $S = \{x \mid H(x, J^S(1^{|x|}))\text{accepts}\}$.

Intuitively, a set is self-encodable if, given the set, one can quickly distill enough information to allow quick answers to future membership queries. That is, we can make a crib sheet.

2. *[8] If $S$ is sparse and $S$ is not* $\mathrm{P}^S$*-printable, then* $E^S \neq NE^S$.

**Theorem 1.3** *If* $\mathrm{P} = \mathrm{NP}$ *and $S$ is sparse, the following are equivalent:* [4]

1. *$S$ is* $\mathrm{P}^S$*-printable (self-printable).*
2. *$S$ is self-encodable.*
3. $\mathrm{P}^S = \mathrm{NP}^S$.
4. $\mathrm{E}^S = \mathrm{NE}^S$.
5. $S \in \mathrm{PH}//\mathrm{poly}$.
6. $(\exists c)(S \subseteq K^S[c \log n, n^c + c])$.

Finally, we note that the same techniques can be applied to many other classes. We state theorems for coNP, E, NE, coNE, PSPACE, and even the polynomial hierarchy. For example, in worlds where coNP = NP, we completely characterize the sparse sets separating coNP and NP. Interestingly, we note that oracles from the advice hierarchy [14] have no effect on the structure of the polynomial hierarchy. This gives strong evidence that PSPACE is not in the advice hierarchy.

**Theorem 1.4**

1. *If* $\mathrm{coNP}^A = \mathrm{NP}^A$ *and $S$ is a sparse set, then* $[\mathrm{coNP}^{A,S} = \mathrm{NP}^{A,S} \iff S$ *is* $\mathrm{NP}^{A,S}$*-printable*$]$.
2. *If* $\mathrm{P}^A = \mathrm{PSPACE}^A$ *and $S$ is a sparse set, then* $[\mathrm{P}^{A,S} = \mathrm{PSPACE}^{A,S} \iff S$ *is* $\mathrm{P}^{A,S}$*-printable*$]$.

**Theorem 1.5** *If $T$ is in the advice hierarchy then* $[$ PH *collapses* $\iff \mathrm{PH}^T$ *collapses*$]$.

**Corollary 1.6** *If* PSPACE *is in the advice hierarchy, then* PH *collapses.*

## 2 Basic Result

We prove, in worlds where P = NP, that the sparse sets separating NP from P are exactly those that are not *self-printable*. The internal organization of these sets is so complex that even given the set as an oracle, no polynomial time machine can hunt down the strings in the set.

[4] Definitions are stated in the body of this paper.

**Definition 2.1**

1. *A set $T$ is* $\mathrm{P}^S$-printable *iff there is a polynomial time function $f$ so* $f^S(1^n) = (\Sigma + \epsilon)^n \cap T$.
2. *A set $T$ is* self-printable *iff $T$ is $\mathrm{P}^T$-printable.*

**Theorem 2.2**

- *If* $\mathrm{P} = \mathrm{NP}$ *and $S$ is sparse, then* $[\mathrm{P}^S = \mathrm{NP}^S \iff S$ *is* $\mathrm{P}^S$*-printable*$]$.
- *(Relativized Version) If* $\mathrm{P}^A = \mathrm{NP}^A$ *and $S$ is sparse, then* $[\mathrm{P}^{A,S} = \mathrm{NP}^{A,S} \iff S$ *is* $\mathrm{P}^{A,S}$*-printable*$]$.

**Proof of Theorem 2.2**

$\Rightarrow$ Suppose $S$ is sparse and $\mathrm{P}^S = \mathrm{NP}^S$. Let $L_{\text{search}} = \{(a,b) \mid$ there is a string in $S$ lexicographically between $a$ and $b\}$. $L_{\text{search}}$ is in $\mathrm{NP}^S$, so by assumption, $L_{\text{search}} \in \mathrm{P}^S$. Since $S$ is sparse, it is easy to see that, using binary search repeatedly to find the smallest new element of $S$, we can $\mathrm{P}^{L_{\text{search}}}$-print $S$. Thus, $S$ is $\mathrm{P}^S$-printable.

$\Leftarrow$ Suppose $S$ is $\mathrm{P}^S$-printable and $\mathrm{P} = \mathrm{NP}$. Let $L \in \mathrm{NP}^S$, say $L = L(N^S)$ running in $\mathrm{NTIME}[p(n)]$. Consider the set $U = \{x\#T \mid T$ is a set of strings and $N^T(x)$ accepts$\}$. $U$ is in NP. Thus by assumption $U$ is in P. Since $S$ is $\mathrm{P}^S$-printable, we can build a $\mathrm{P}^S$ machine that, on input $x$, first finds $(\Sigma + \epsilon)^{p(|x|)} \cap S$, and then asks if $x\#((\Sigma + \epsilon)^{p(|x|)} \cap S)$ is in $U$. It accepts if so, otherwise it rejects. Thus $L \in \mathrm{P}^S$. Since $L$ was an arbitrary $\mathrm{NP}^S$ language, $\mathrm{P}^S = \mathrm{NP}^S$. ♠

Interestingly, the characterization of Theorem 2.2 is not one of classical (uniform) complexity. Indeed, many uncomputable sets $S$ are $\mathrm{P}^S$-printable. The theorem states that the P = NP question crucially depends on the complexity of the organization of $S$, a non-uniform property.

We proved our theorem in a simple form. In the following section we dissect and strengthen our theorem, and explore the behavior of sparse self-printable sets in worlds where P = NP.

## 3 Self-encodability

The previous section focused on self-printability. Self-printability is a valuable tool in discussions of sparse sets, but only sparse sets can be self-printable. Nonetheless, we'd like to be able to prove theorems about sets that have "sparse information," even if they are dense.

**Definition 3.1** *A set $S$ is* self-encodable *iff there are polynomial time machines $H$ and $J$ so that* $S = \{x \mid H(x, J^S(1^{|x|}))$ accepts$\}$.

**Definition 3.2**

1. $C/\mathrm{poly} = \{S \mid \exists f$ *so* $\lambda y.|f(1^{|y|})|$ *is in* P *and* $D_{S,f} \cap C \neq \emptyset\}$, *where* $D_{S,f} = \{T \mid (\forall x)[x\#f(1^{|x|}) \in T \iff x \in S]$.
2. $C//G = \{S \mid S$ *is in* $C/\mathrm{poly}$ *with $f$ computable in* $G^S\}$.
   *Note: $S$ is self-encodable* $\iff S \in \mathrm{P}//\mathrm{poly}$.

Though we give a machine-based definition of self-encodability, the motivation for self-encodability lies in the non-uniform complexity classes of Karp and Lipton [10]. Definition 3.2, part 2, notes this connection.

The Karp-Lipton classes of Definition 3.2, part 1, contain sets that are easy to recognize, ***given a (perhaps uncomputable) advice string dependent only on the length of the input string.*** For example, P/poly is the class of all sets $S$ so that there is an advice function $f$ (whose output *size* is polynomial in the size of its input) for which "$x\#f(1^{|x|}) \in S$?" is answerable in P. Self-encodability reflects the added property that a set's advice function can be easily computed, given the set.

Loosely put, a self-encodable set is one for which, given the set on loan, we can quickly distill a small 'crib sheet' with which we can quickly answer membership questions even when the set is taken away. Self-encodability has less to do with the classical complexity of the set than with the fact that the set's organization is internally systematic. Indeed, though there are uncomputable self-encodable sets, there are also non-self-encodable sets in $\mathrm{TIME}[n^{\log n}]$.

Using self-encodability, the flavor of Theorem 2.2 applies even to non-sparse sets. Theorem 3.3 is a broad generalization of a theorem of [2] that applies only to tally sets.

**Theorem 3.3** *If* $\mathrm{P} = \mathrm{NP}$ *and $S$ is self-encodable, then* $\mathrm{P}^S = \mathrm{NP}^S$.

**Proof of Theorem 3.3** Let $L \in \mathrm{NP}^S$, say $L = L(N^S)$ running in $\mathrm{NTIME}[p(n)]$. Let $H$ and $J$ be the machines of Definition 3.1 that certify $S$'s self-encodability. Consider the set

> $U = \{x\#y \mid y = y_0, \ldots, y_{p(|x|)}$ and if we simulate $N(x)$, answering oracle queries of the form "$z \in S$?" by running $H(z, y_{|z|})$, $N$ accepts$\}$.

$U$ is in NP so by assumption $U$ is in P. Thus there is a machine in $\mathrm{P}^S$ that on input $x$ computes $y = J^S(1^0), \ldots, J^S(1^{p(|x|)})$ and then answers the P question: "$x\#y \in U$." Thus $\mathrm{P}^S = \mathrm{NP}^S$. ♠

Finally, Theorem 3.6 shows that in worlds where P = NP many properties become equivalent on sparse sets.

**Definition 3.4** *The time-bounded Kolmogorov class [6][12] $K^S[f(n), g(n)]$ is defined by: $x \in K^S[f(n), g(n)] \iff$ there is a string $y$ of length at most $f(|x|)$ for which $M^S_{\mathrm{universal}}(y)$ computes output $x$ in at most $g(|x|)$ steps.*

**Lemma 3.5** *$S$ is $\mathrm{P}^S$-printable $\iff (\exists c)(S \subseteq K^S[c \log n, n^c + c])$. This says that strings in self-printable sets have short names: their position in the set; strings in the Kolmogorov class can be self-printed by brute force.*

**Theorem 3.6** *If P = NP and $S$ is sparse, the following are equivalent:*

1. *$S$ is $\mathrm{P}^S$-printable (self-printable).*
2. *$S$ is self-encodable.*
3. *$\mathrm{P}^S = \mathrm{NP}^S$.*
4. *$\mathrm{E}^S = \mathrm{NE}^S$.*
5. *$S \in \mathrm{PH//poly}$.*
6. *$(\exists c)(S \subseteq K^S[c \log n, n^c + c])$.*

**Proof of Theorem 3.6** 1≡6 by Lemma 3.5. Clearly $3 \Rightarrow 4$ and $1 \Rightarrow 2$. $4 \Rightarrow 3$, in a fashion similar to the left to right direction of Theorem 2.2, as shown in [8]. Self-encodability, viewed as P//poly, is easily seen to be PH//poly when P = NP, so $5 \equiv 2$. Finally, $2 \Rightarrow 3$ by Theorem 3.3 and $3 \Rightarrow 1$ by Theorem 2.2. ♠

Figure 1 summarizes our classification of the effects of nonuniform oracles. We use, in addition to the results already stated, the simple fact that: If $S$ is in P//NPF but $S$ is not self-encodable, then $\mathrm{P}^S \neq \mathrm{NP}^S$.

# 4 Generalizations

The thrust of this paper is to encourage the characterization of oracles achieving specified relativizations. We've characterized sparse sets separating NP from P in worlds where P = NP. The same techniques apply to a wide variety of classes. For sparse sets, we characterize sets separating coNP and NP in worlds where NP = coNP, and sets separating PSPACE from P in worlds where P = PSPACE.

**Theorem 4.1**

1. *If* $\mathrm{coNP}^A = \mathrm{NP}^A$ *and* $S$ *is a sparse set, then* $[\mathrm{coNP}^{A,S} = \mathrm{NP}^{A,S} \iff S$ *is* $\mathrm{NP}^{A,S}$*-printable*$]$.
2. *If* $\mathrm{P}^A = \mathrm{PSPACE}^A$ *and* $S$ *is a sparse set, then* $[\mathrm{P}^{A,S} = \mathrm{PSPACE}^{A,S} \iff S$ *is* $\mathrm{P}^{A,S}$*-printable*$]$.

Underlying these theorems are lemmas on the effects of oracle queries. We state these lemmas, and some related ones for exponential time classes. Some of the conditions involved are delicate, requiring certain functions to be total and restricting the allowed query lengths of oracle machines. However, the proofs, included in Appendix A, are in spirit the same as the proof of Theorem 3.3. Figure 2 summarizes our results of this type.

**Theorem 4.2** [5]

1. $\mathrm{NP} = \mathrm{coNP} \iff (\forall T \in \mathrm{PH}//\mathrm{NPF}_{\mathrm{total}})[\mathrm{NP}^T = \mathrm{coNP}^T]$.
2. $\mathrm{P} = \mathrm{PSPACE} \iff (\forall T \in \mathrm{P}//\mathrm{poly})[\mathrm{P}^T = \mathrm{PSPACE}^T]$.
3. $\mathrm{E} = \mathrm{NE} \iff (\forall T \in \mathrm{P}/(\mathrm{poly}, \mathrm{EF}_{\mathrm{P}}))[\mathrm{E}_{\mathrm{P}}^T = \mathrm{NE}_{\mathrm{P}}^T]$.
4. $\mathrm{NE} = \mathrm{coNE} \iff (\forall T \in \mathrm{P}/(\mathrm{poly}, \mathrm{NEF}_{\mathrm{P,total}}))[\mathrm{NE}_{\mathrm{P}}^T = \mathrm{coNE}_{\mathrm{P}}^T]$.

**Corollary 4.3**

1. *If* $\mathrm{coNP} = \mathrm{NP}$ *and* $S$ *is* $\mathrm{NP}^S$*-printable, then* $\mathrm{NP}^S = \mathrm{coNP}^S$.
2. *If* $\mathrm{P} = \mathrm{PSPACE}$ *and* $S$ *is* $\mathrm{P}^S$*-printable, then* $\mathrm{P}^S = \mathrm{PSPACE}^S$.

# 5 The Polynomial Hierarchy and Compressible Sets

Just as Theorem 3.3 broadens results from tally sets to self-encodable sets, so also can we broaden many results from sparse sets to the advice hierarchy [14]. The advice hierarchy, PH/poly in the Karp-Lipton notation of Definition 1, is the class of all sets recognizable within the polynomial hierarchy,

[5]Notations defined in Figure 2.

given a polynomial sized amount of (sorcerously obtained) advice dependent only on the length of the input.

A detailed combination of the techniques of this paper and the methods of [2] shows that an oracle from the advice hierarchy collapses its relativized polynomial hierarchy if and only if the unrelativized polynomial hierarchy collapses. (This was previously known only for the special case of sparse sets, which are all contained in P/poly.) As a consequence, if PSPACE is contained in the advice hierarchy, then the polynomial hierarchy collapses (and, indeed, by employing the techniques of [10], PSPACE=PH).

**Theorem 5.1** *If $T$ is in the advice hierarchy, then* [ PH *collapses* $\Longleftrightarrow$ $\mathrm{PH}^T$ *collapses*].

**Proof of Theorem 5.1** Appendix B.

**Corollary 5.2** *If* PSPACE *is in the advice hierarchy, then* PH *collapses.*

# 6 Conclusions

Self-encodability largely characterizes the oracle set organizations that allow the NP mechanism no more power than the P mechanism. For sparse sets in worlds where P = NP, this characterization is complete. Thus, our suggested research paradigm of classifying oracles that achieve a specified relativization has rewarded us with deeper insight into the linkage between the relativized and unrelativized structure of the polynomial hierarchy.

# A Appendix:

### Proof of Theorem 4.2

$\Longleftarrow$ directions: Simply set $T = \emptyset$.
$\Longrightarrow$ directions:

1. Assume NP = coNP and $T \in \mathrm{PH}//\mathrm{NPF}_{\mathrm{total}}$. We must show $\mathrm{NP}^T = \mathrm{coNP}^T$.

   (a) First we show that $\mathrm{NP}^T \subseteq \mathrm{coNP}^T$. Let $L$ be any $\mathrm{NP}^T$ language. We will show that $L$ is in $\mathrm{coNP}^T$.

   Since $T$ is in $\mathrm{PH}//\mathrm{NPF}_{\mathrm{total}}$, there is a machine $M_T$ in the polynomial hierarchy (and thus in NP) and an NP function $f$, well-defined on all inputs, so $x$ is in $T$ if and only if $M(x\#f^T(1^{|x|}))$ accepts.

   Let $N_L^T$ accept $L$ and w.l.o.g. run in Ntime$[n^i + i]$. Let's define two useful sets.

   $R = \{x\#z \mid M(x\#z) \text{ accepts}\}$
   $S_\star = \{x\#y_{|x|^i+i}\#y_{|x|^i+i-1}\# \cdots \#y_0 \mid N_L^R(x) \text{ accepts }\}$

   $R$ is in NP (=coNP), so $S_\star$ is in $\mathrm{NP}^R \subseteq \Sigma_2 \subseteq \mathrm{coNP}$. $S_\star$ says: using $y$ as a collection of advice strings for $T$, we think that $N_L^T(x)$ accepts.

   Now, consider the coNP machine that (on input $x$) on each computation path tries to guess a function-computing path[6] of $f(1^0)$ and $f(1^1) \ldots$ and $f(1^{|x|^i+i})$. On a coNP path that fails to compute one of the advice strings (i.e., which has guessed some non-function-computing path), accept out of hand. On paths that do compute all the advice functions, $y = y_{|x|^i+i}\#y_{|x|^i+i-1}\# \cdots$-$\#y_0$, accept if and only if $x\#y$ is in $S_\star$. Since $T \in \mathrm{PH}//\mathrm{NPF}_{\mathrm{total}}$, some path will know the advice, so we'll accept if and only if $x$ is in $L$. Thus $\mathrm{NP}^T \subseteq \mathrm{coNP}^T$.

   (b) The proof that $\mathrm{coNP}^T \subseteq \mathrm{NP}^T$ is similar.

2. $\mathrm{P} = \mathrm{PSPACE} \wedge T \in \mathrm{P}//\mathrm{poly} \Longrightarrow \mathrm{P}^T = \mathrm{PSPACE}^T$, just as in Theorem 3.3. Crucially, note that we are using the standard model of

[6]Recall, for our total NP function, at least one computation path computes the function value, and all paths that compute a value ("function-computing paths") compute the same value.

relativized PSPACE [4]; there is a polynomial bound on the lengths of oracle queries.

3. It suffices to show: $\mathrm{E} = \mathrm{NE} \wedge T \in \mathrm{P}/(\mathrm{poly}, \mathrm{EF_P}) \Longrightarrow \mathrm{E}_\mathrm{P}^T = \mathrm{NE}_\mathrm{P}^T$. Since $T \in \mathrm{P}/(\mathrm{poly}, \mathrm{EF_P})$, there is a polynomial time machine $M$ and an exponential time function $f$ with a polynomial bound on its query lengths and output size for which:

$$x \in T \Longleftrightarrow M(x, f^T(1^{|x|})) \text{ accepts.}$$

Now let $A$ be an arbitrary language in $\mathrm{NE}_\mathrm{P}^T$. Thus $A = L(N^T)$ for some NE machine $N$ that on input $x$ queries strings at most polynomially longer that $x$ (say at most length $q(|x|)$). The following scheme shows that $A$ is in fact in $\mathrm{E}_\mathrm{P}^T$.

(a) On input $x$, compute $y = f^T(1)\# \cdots \# f^T(1^{q(|x|)})$.

(b) Ask if $x\#y \in B$, where

$B = \{a\#b \mid b = b_1, \ldots, b_{q(|x|)} \text{ and } N^{C_b}(a) \text{ accepts}\}$.

$C_b = \{z \mid M(z, b_{|z|}) \text{ accepts}\}$.

These two steps can easily be done by a $\mathrm{E}_\mathrm{P}^T$ machine, since $B \in \mathrm{NE}$ so $B \in \mathrm{E}$.

4. Combine the method of parts 1 and 3 above.

# B Appendix:

### Proof of Theorem 5.1

Here we prove Theorem 5.1 of Section 5. For sparse oracles, this theorem couples the collapse of the relativized and unrelativized polynomial hierarchies. Our supporting lemmas go further. They link the extent of these collapses, as described below in Fact B.4. When either the unrelativized or relativized polynomial hierarchy collapses, the other can never extend too much farther.

**Theorem B.1** *If $T$ is in the advice hierarchy, then* [PH*collapses if and only if* $\mathrm{PH}^T$ *collapses*].

**Proof of Theorem B.1** The theorem follows from the two lemmas below. ♠

**Lemma B.2** *If* $T \in \mathrm{PH/poly}$ *and* $(\exists k)[\mathrm{PH} \subseteq \Sigma_k^T]$, *then the polynomial hierarchy collapses.*

**Lemma B.3** *If* $T \in \mathrm{PH/poly}$ *and the polynomial hierarchy collapses, then* $\mathrm{PH}^T$ *collapses.*

**Corollary B.4**

1. *Suppose* $\mathrm{PH} = \Sigma_k$. *For all sets* $T$ *in the advice hierarchy,* $\mathrm{PH}^T = \Sigma_{k+2}$.
2. (a) *Let* $T$ *be in the advice hierarchy with* $T \in \Sigma_m/\mathrm{poly}$. *If* $\mathrm{PH}^T = \Sigma_l^T$, *then* $\Sigma_{l+k+2} = PH$.
   (b) *If* $T$ *is in the advice hierarchy and* $PH^T = \Sigma_l^T$, *then* $PH = \Sigma_{2l+2}$.

**Proof of Corollary B.4**: These facts follow, respectively, from the proofs of Lemmas B.2 and B.3. ♠

**Sublemma B.5** $(\forall T \in \Sigma_m/\mathrm{poly})(\forall A)$[*If* $A$ *is self-reducible*[7] *and* $(\exists k \geq 1)[A \in \Sigma_k^T]$, *then* $A$ *is in* $\Sigma_{k+m+1}$.]

**Proof of Lemma B.2** $B_i$, the standard $\Sigma_i$ complete set of [13], is self-reducible. So by our hypothesis, $B_i$ is in $\mathrm{PH} \subseteq \Sigma_k^T$. Thus, by Sublemma B.5, $B_i \in \Sigma_{k+3+c_T}$. Since this membership holds for all $i$, the polynomial hierarchy is contained in $\Sigma_{k+3+c_T}$. ♠

**Proof of Sublemma B.5**

Fix $T \in \Sigma_m/\mathrm{poly}$ and $A$ self-reducible. There are strings $\{w_i\}$, a $\Sigma_m$ machine $M_T$, and a polynomial $r(\cdot)$ with $|w_i| \leq r(i)$ so that

$$(\forall z)[z \in T \iff M_T(z, w_{|z|}) \text{ accepts}].$$

Also, there is a polynomial time machine $M_{\mathrm{self}}$ that on input of size $n$ queries strings of length at most $n-1$ for which $(\forall z)[z \in A \iff M_A^A(z) \text{ accepts}]$. Since $A \in \Sigma_k^T$, there is a $\Sigma_k^T$ machine $M_k$ accepting $A$.

Our plan is as follows.

1. On input $x$ we guess some advice strings (y).
2. Each guess $y$ will, via $M_T$, define a set $T_y$.

[7] A set $A$ is self-reducible [2][9] if $[x \in A \iff M^{A|(|x|-1)}(x) \text{ accepts}]$ for some polynomial time machine $M$. $A|k$ denotes $A$ restricted to strings of length at most $k$.

3. For each $T_y$, we check if it has the agreeable behavior that, if it is used as T, $M_{\text{self}}$ and $M_A$ agree up to length $|x|$. Later we note that this implies that they both compute $A$ correctly.

4. For $T_y$ that survived the above test, we use $T_y$ to test if $x$ is in $A$.

Now let's go through each step in detail. In effect, we are describing the code of a $\Sigma_{k+m+2}$ machine accepting $A$.

**Step 1** On input of size $z$, $f$ queries strings at most polynomially longer than $|z|$, say $q(|z|)$. On input $x$, we immediately guess all advice strings $y = y_0\#\cdots\#y_{q(|x|)}$ for which $(\forall i \leq q(|x|))[|y_i| \leq r(i)]$.

**Step 2** Each guess $y$ defines a set $T_y$. $T_y$ is what $T$ is if the advice strings $y$ are assumed to be the advice strings of $T$. Similarly, $y$ defines a guess for $A$.

$$T_y = \{z \mid M_T(z, y_{|z|}) \text{ accepts}\}$$
$$A_y = \{z \mid M_A^{T_y}(z, y_{|z|}) \text{ accepts}\}$$

**Step 3** Given a guess $y$, we are interested in if $T_y$ allows us to correctly recreate $A$. Crucially, note that if $(*)$ holds,

$$(\forall z \ni |z| \leq |x|)[M_{\text{self}}^{A_y}(z) \text{ accepts} \iff z \in A_y] \quad (*)$$

then we can conclude that $M_{\text{self}}^{A_y}$ computes $A$ for strings of length at most $|x|$. This conclusion holds by straightforward induction, using the fact that our machine $M_{\text{self}}$ queries only strings shorter than its input.

Now we simply check condition $(*)$ for our current $y$. If it holds, this path goes on to Step 4. Otherwise, this path rejects.

Let's note two facts. First, at least one path will go on to Step 4- the path that guesses the actual advice strings of $T$. Secondly, many paths with different $T_y$'s may go on to Step 4. The fact that the paths have different guesses of $T$ does not trouble us; each path has a guess of $T$ that correctly aids membership testing in $A$.

**Step 4** On the current path, accept if and only if $M_A^{T_y}(x)$ accepts.

We sketch the cost of the above procedure.

$T_y \in \Sigma_m$.
$A_y \in \Sigma_k^{T_y} \subseteq \Sigma_{k+m}$.
$H_y = \{z \mid M_{\mathrm{self}}^{A_y}(z) \text{ accepts}\} \in \mathrm{P}^{A_y} \subseteq \Delta_{k+m+1}$

So testing condition $(*)$ of Step 3 is of the form $\forall_{\mathrm{polynomiallybounded}}[\Delta_{k+m+1} \Longleftrightarrow \Sigma_{k+m}]$. This can be done in $\Pi_{k+m+1}$. Taking into account the nondeterministic initial guess of Step 1, the whole algorithm runs in $\Sigma_1^{\Pi_{k+m+1}} \subseteq \Sigma_{k+m+2}$. ♠

**Proof of Lemma B.3**

Suppose PH $= \Sigma_r$ and $T \in$ PH/poly. Let $Y \in \mathrm{PH}^T$. We show that $Y \in \Sigma_{r+2}^T$, thus $PH^T = \Sigma_{r+2}^T$.

Since T is in PH/poly, there are strings $\{y_i\}$, a PH machine $M_T$, and a polynomial $q(\cdot)$ with $|y_i| \leq q(i)$ so that

$$(\forall z)[z \in T \Longleftrightarrow M_T(z, y_{|z|}) \text{ accepts}].$$

Since $Y \in \mathrm{PH}^T$, there is a $\mathrm{PH}^T$ machine $M_Y$, with each computation path having at most polynomially many steps, say $p(\cdot)$, so $Y = L(M_Y)$.

Let

$L_1 = \{< x, y > \mid y = y_0\# \cdots \# y_{p(|x|)} \text{ and } M_Y^{A_y}(x) \text{ accepts}\}$
$A_y = \{z \mid M_T(z, y_{|z|}) \text{ accepts}\}$.

That is, check if $x \in Y$ assuming that the advice strings $y$ are correct advice strings for $T$.

Now $L_1 \in$ PH so $L_1 \in \Sigma_r$. So, *if we can find good advice strings*, we can use $L_1$ to check membership in $Y$. The set below defines good advice strings.

$L_2^T =$
$\{< l, w > \mid w \text{ is a good advice string of length } l \text{ for } T\} =$
$\{< l, w > \mid (\forall x)[|x| = l \Longrightarrow [x \in T \Longleftrightarrow M_T(x, w) \text{ accepts}]]\}$.

$L_2^T$ is easily in $\Pi_1^{\Sigma_r \oplus T} \subseteq \Pi_{r+1}^T$.

Our $\Sigma_{r+2}^T$ machine for $Y$ runs as follows on input $x$. It guesses many advice strings, then checks to find good advice strings, and finally feeds the good strings to $L_1$.

{Nondeterministically guess advice strings $y = y_0, \ldots, y_{p(|x|)}$.

/* Now there are many computation paths*/

If $(\exists i)[0 \leq i \leq p(|x|) \wedge \langle i, y_i \rangle \notin L_2^T]$ reject on this path.

/* At this point, we know the advice strings are good*/
/* At least one path will survive to this point
since $T$ is in the advice hierarchy */

Accept on this path if and only if $\langle x, y \rangle \in L_1$.
}
The above program runs in $\mathrm{NP}^{L_2^T \oplus L_1} \subseteq \mathrm{NP}^{\Pi_{r+1}^T \oplus \Sigma_r} \subseteq \Sigma_{r+2}^T$. ♠

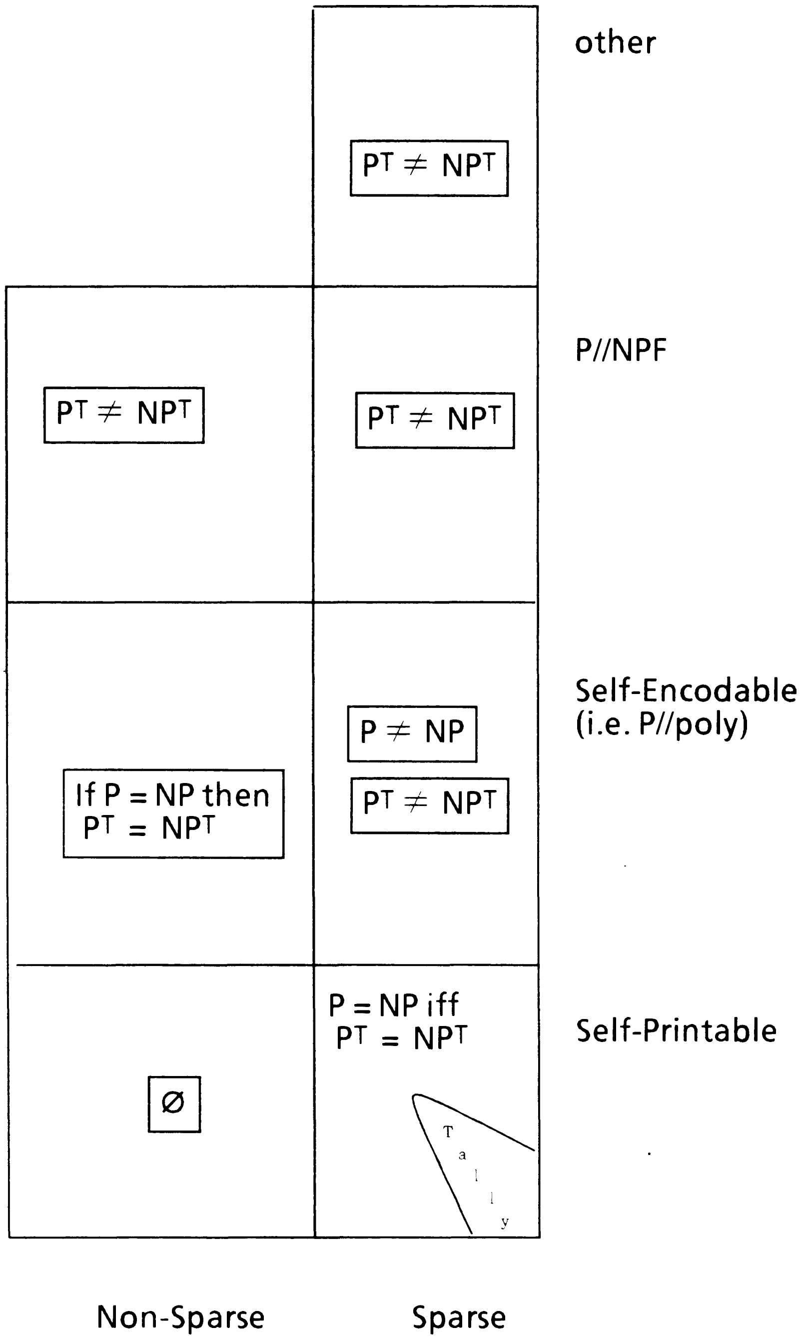


Figure 1

The Effects of Nonuniform Oracles

Each box lists the conclusions that can be drawn if there is a set T in that box. For example, for every sparse set T that is self-encodable but not self-printable, $P^T \neq NP^T$. Indeed, if any such set T exists then $P \neq NP$.

| $Q \Leftrightarrow (\forall S \in C)[R]$ | | |
|---|---|---|
| Q | C | R |
| P = NP | PH//poly | $P^S = NP^S$ |
| NP = coNP | $PH//NPF_{total}$ | $NP^S = coNP^S$ |
| P = PSPACE | PSPACE//poly | $P^S = PSPACE^S$ |
| E = NE | $P/(poly,EF_P)$ | $E_P{}^S = NE_P{}^S$ |
| NE = coNE | $P/(poly,NEF_{P,\,total})$ | $NE_P{}^S = coNE_P{}^S$ |
| PH collapses | PH/poly | $PH^S$ collapses |

Figure 2

An NP function maps from $\Sigma^*$ to $\Sigma^* \cup$ **undef**. It takes value **undef** if each computation path declares itself unable to compute the function. It computes value y if every path that computes some value computes y, and at least one path computes y. For a valid NP function, we require that one of these two possibilities occurs for each input.

$NPF_{total}$ is the class of NP functions (as defined in this section) that never take on the value **undef**. That is, they are the total functions computable in NP. E(NE), exponential time, is $\cup_{c>0}$Time$[2^{cn}]$ ($\cup_{c>0}$Ntime$[2^{cn}]$). EF is the class of functions computable in deterministic exponential time. $NEF_{total}$ is the class of NE functions that never take on the value **undef**. $E_P$ ($NE_P$) is the class of languages recognized by a deterministic (nondeterministic) exponential time machine that only queries its oracle about strings of length at most polynomial in the input size.

We say $S \in C/(F,G)$ if S is in C/F, and S has an advice function computable in $G^S$.

## References


[1] T. Baker, J. Gill, and R. Solovay, "Relativizations of the *P=?NP* Question," *SIAM Journal on Computing*, 1975, pp. 431-442.

[2] J. Balcázar, R. Book, T. Long, U. Schöning, and A. Selman, "Sparse Oracles and Uniform Complexity Classes," *FOCS 1984*, pp. 308-313.

[3] A. Chandra, D. Kozen, and L. Stockmeyer, "Alternation," *JACM*, V. 26, #1, 1981.

[4] M. Furst, J. Saxe, and M. Sipser, "Parity, Circuits, and the Polynomial-Time Hierarchy," *FOCS 1981*, pp. 260-270.

[5] W. Gasrarch, "Recursion Theoretic Techniques in Complexity Theory and Combinatorics," Center for Research in Computing and Technology Report TR-09-85, Harvard University, May 1985.

[6] J. Hartmanis, "Generalized Kolmogorov Complexity and the Structure of Feasible Computations," Cornell Department of Computer Scince Technical Report TR 83-573, September 1983.

[7] J. Hartmanis, to appear.

[8] J. Hartmanis and Y. Yesha, "Computation Times of NP Sets of Different Densities," *Theoretical Computer Science*, V. 34, 1984, pp. 17-32.

[9] K-I. Ko, "On Self-reducibility and Weak P-Selectivity," *Journal of Computer and System Sciences*, V. 26, 1983, pp. 209-221.

[10] R. Karp and R. Lipton, "Some Connections Between Nonuniform and Uniform Complexity Classes," *STOC 1980*, pp. 302-309.

[11] S. Mahaney "Sparse Complete Sets for NP: Solution of a Conjecture of Berman and Hartmanis," *FOCS 1980*, pp. 54-60.

[12] M. Sipser, "A Complexity Theoretic Approach to Randomness," *STOC 1983*, pp. 330-335.

[13] C. Wrathall, "Complete Sets and the Polynomial-time Hierarchy," *Theoretical Computer Science*, V. 3, 1977, pp. 23-33.

[14] C. Yap, "Some Consequences of Non-uniform Conditions on Uniform Classes," *Theoretical Computer Science*, V. 26, 1983, pp.287-300.