POLYNOMIAL TIME COMPUTATIONS IN MODELS OF ET*

Deborah Joseph

TR 82-500
June 1982

Computer Science Department
Upson Hall
Cornell University
Ithaca, New York 14853

*This paper is an extension of the abstract, "Fast programs for initial segments and polynomial time computations in weak models of arithmetic," which was presented at SIGACT in 1981. It contains additional results from the author's Ph.D. dissertation, which was written at Purdue University under the direction of Paul Young. Much of this research was done with the support of NSF Grant MCS-7609232 A02.

POLYNOMIAL TIME COMPUTATIONS IN MODELS OF ET[+]

Deborah Joseph

Computer Science Department
Cornell University
Ithaca, NY 14853

ABSTRACT

We investigate formal notions of computations in nonstandard models of the weak arithmetic theory ET - the theory of exponential time. It is shown that ET is a sufficiently weak theory that many of the natural notions are not preserved.

[+] This paper is an extension of the abstract, "Fast programs for initial segments and polynomial time computations in weak models of arithmetic," which was presented at SIGACT in 1981. It contains additional results from the author's Ph.D. dissertation, which was written at Purdue University under the direction of Paul Young. Much of this research was done with the support of NSF Grant MCS-7609232 A02.

POLYNOMIAL TIME COMPUTATIONS IN MODELS OF ET[+]

Deborah Joseph

Computer Science Department
Cornell University
Ithaca, NY 14853

1. Introduction

In this paper we study two alternative approaches for investigating whether NP-complete sets have fast algorithms. One is to ask whether there are long initial segments on which such sets are easily decidable by relatively short programs. The other approach is to ask whether there are weak fragments of arithmetic for which it is consistent to believe that P = NP. We show that the two questions are equivalent: It is consistent to believe that P = NP in certain models of weak arithmetic theories iff it is true (in the standard model of computation) that there are infinitely many initial segments on which satisfiability is polynomially decidable by programs that are much shorter than the length of the initial segment.[1]

In a paper presented at SIGACT in 1980 DeMillo and Lipton, [D&L-80],

---

[+]This paper is an extension of the abstract, "Fast programs for initial segments and polynomial time computations in weak models of arithmetic," which was presented at SIGACT in 1981. It contains additional results from the author's Ph.D. dissertation, which was written at Purdue University under the direction of Paul Young. Much of this research was done with the support of NSF Grant MCS-7609232 A02.

1. The relationship between independence results and fast algorithms for NP-complete sets is also discussed, but in a different light, by DeMillo and Lipton in [D&L-79].

brought model theoretic techniques to bear on the P =? NP question. They presented a weak theory of arithmetic, which they called ET (Theory of Exponential Time), and showed that there is a model, $M_{D\&L}$, of ET in which every NP-complete A predicate is equivalent to a predicate in a class which they call P (we will refer to it here as $P_{D\&L}$). One believes intuitively that the predicates in $P_{D\&L}$ should be polynomially computable. Nevertheless ET is a weak theory and models like $M_{D\&L}$ have some fairly unusual properties. For example, we will show that the fact that every predicate in $P_{D\&L}$ has a program whose runtime is bounded by a polynomial in models of Peano arithmetic, does not imply that the predicates in $P_{D\&L}$ are actually computable in models of ET.

In Sections 2 and 3 we discuss polynomial time computations in models of ET, and we give examples of simple functions that are not total in $M_{D\&L}$ as well as predicates in $P_{D\&L}$ that are not computable in $M_{D\&L}$. We then, in Section 4, investigate a slightly richer theory, ET(Elem), and characterize the sets that are polynomially computable in some model of ET(Elem). In doing so, we show that satisfiability (SAT), or for that matter <u>any</u> elementary set, is polynomially decidable in certain models similar to $M_{D\&L}$ if and only if in the standard model of computation there are short, fast programs (polynomial or linear time) for arbitrarily long initial segments. In this way we show that the question of polynomial time computation in models of ET(Elem) is strongly tied to a fairly natural and important question concerning the feasible computation of hard sets: What types of hard sets have arbitrarily long initial segments that can be efficiently decided by short programs? This question is important in its own right. From a certain finitistic point of view such sets might be considered easily decidable even if there is no single program that uniformly decides the entire set in polynomial time. Section 5

is devoted to a discussion of this question. We show that it is easily resolved for P and EXP-time complete sets by observing that these sets never have arbitrarily long initial segments that are efficiently decided by short programs. Unfortunately, the question for NP-complete sets is left open, leaving it unclear whether these sets are actually polynomially computable in models similar to $M_{D\&L}$.

2. The Theory ET and a Theorem of DeMillo and Lipton

The theory ET is based on a language that is an extension of the language for Peano arithmetic. In addition to function symbols for + and * the language for ET includes function symbols: $\dot{-}$ for monus (subtraction on the natural numbers), min(x,y), max(x,y) and 0^x, 1^x, 2^x, ... for constant exponentiation. It also contains predicate symbols, $P_0(\bar{x})$, $P_1(\bar{x})$, $P_2(\bar{x})$, ... for each of the standard polynomially time testable relations. DeMillo and Lipton defined ET to be the theory that has as axioms all of the <u>true</u> universal sentences over this extended language. That is, all of the sentences in prenex form involving only universal quantifiers that are expressible in the language of ET and are <u>true</u> in the standard model -- the natural numbers.

In [D&L-80] DeMillo and Lipton investigate whether or not ET is a rich enough theory to prove that P $\neq$ NP. They begin by defining a class of predicates which they call P; we will refer to this class as $P_{D\&L}$. Intuitively, one believes that all of the predicates in $P_{D\&L}$ should be testable in polynomial time and with this assumption DeMillo and Lipton show that is consistent with ET to believe that P = NP. Thus showing that ET is not strong enough to prove P $\neq$ NP even if this is the case. The main theorem of DeMillo and Lipton's paper is the following:

2.1. Theorem (DeMillo and Lipton): Let A be a predicate so that $\{x: (\exists \bar{y})\ A(x,\bar{y})\ \}$ is NP-complete. Then for some predicate $\phi_A(x) \in P_{D\&L}$,

$$ET + (\forall x)[(\exists \bar{y})A(x,\bar{y}) <=> \phi_A(x)]$$

is consistent. (The class $P_{D\&L}$ is defined formally below.)

As we have mentioned, $P_{D\&L}$ is intended to be a class of polynomial time testable predicates. Formally, the class $P_{D\&L}$ is defined syntactically using a fixed constant symbol. (DeMillo and Lipton use the symbol α, however we will use the symbol a so as not to confuse this symbol with its interpretation in the nonstandard model, which will be α.) DeMillo and Lipton defined $P_{D\&L}$ as follows:

2.2. Definition:

i) every ET predicate symbol is in $P_{D\&L}$,

ii) $P_{D\&L}$ is closed under Boolean operations,

iii) if $\phi(x) \in P_{D\&L}$, then $(\exists x<a)[\phi(x)] \in P_{D\&L}$.

For the proof of Theorem 2.1 it appears that the class $P_{D\&L}$ also needs to be closed under substitution of constants, in particular a. That is,

iv) if $\phi(\bar{x},y) \in P_{D\&L}$ then $\phi(\bar{x},a) \in P_{D\&L}$.

Notice that for every choice of $a \in N$ the predicates in $P_{D\&L}$ are polynomially testable in any of the standard models of computation, e.g. Loop programs[2] over the natural numbers.

---

2. Using Peano arithmetic, or even weaker systems such as Basic Number Theory, [J&Y-81a], one can prove that all of the natural notions of computation: Loop programs, Turing machines, RAMs, Markov Algorithms, Partial Recursive Functions, ... , are equivalent and in fact time and space requirements for these different schemes are polynomially related. However, as we will discuss later, ET is a sufficiently weak theory that this may not be the

DeMillo and Lipton's proof begins by constructing a nonstandard model, $M_{D\&L}$, of ET that contains a nonstandard element α. In this model the constant a in the definition of $P_{D\&L}$ is interpreted as the element α. With this interpretation they show that for every NP-complete set described by a quantifier free predicate in the language of ET, there is a predicate in $P_{D\&L}$ that is equivalent in $M_{D\&L}$.

DeMillo and Lipton's proof can in fact be carried through for every existential predicate $(\exists \bar{y})A(x,\bar{y})$ in the language of ET and α. That is, the proof holds for nonstandard predicates as well as standard predicates and does not depend on the predicate $(\exists \bar{y})A(x,\bar{y})$ characterizing an NP-complete set[3]. Hence with only minor modifications to their proof one obtains the following:

2.3. Theorem: Let $(\exists \bar{y})A(x,\bar{y})$ be any existential predicate definable in $L(ET,\alpha)$. Then for some predicate $\phi(x)_A \in P_{D\&L}$,

$$ET + (\forall x)[(\exists \bar{y})A(x,\bar{y}) <=> \phi_A(x)]$$

is consistent.

At least on the surface Theorem 2.3 is very strong and therefore it seems appropriate to look carefully at what is actually being said. In an attempt to do this, we will spend the remainder of this section and the next section discussing the following questions:

i) What types of sets can be characterized by existential predicates in $L(ET,\alpha)$?

ii) What does it mean to say that a predicate is computable in polynomial

---

case.

3. This observation has been made independently by L. Kirby, [unpublished].

time in a model of ET?

iii) Are the predicates in $P_{D\&L}$ in fact polynomially computable in models of ET?

One commonly used characterization of the r.e. sets is that they are exactly the sets that are Σ_1 definable in L(PA). That is, they have characteristic predicates of the form $(\exists x)\phi(x)$ where $\phi(x)$ may contain bounded quantifiers. Matijesevic has shown, [MAT-70], that every r.e. set is the solution of a diophantine equation and hence every r.e. set is definable by an existential predicate. However, it is unlikely that Matijesevics' theorem is provable in ET. Wilkie and Manders [WIL-81], [MAN-81], have shown that Matijesevics' theorem is provable in $PA^- + I\Sigma_0$ if and only if NP = coNP. ($PA^- + I\Sigma_0$ is Peano arithmetic with induction restricted to formulas whose only quantifiers are bounded.) Thus there may be nonstandard models of ET in which an r.e. set's Σ_1 predicate does not agree with its diophantine description. From a computational viewpoint perhaps the most natural description of the r.e. sets is based on Turing machines and Kleene's $T(i,\bar{x},y,z)$ predicate[4]. However the normal formulation of the T-predicate in the language of Peano arithmetic uses bounded quantifiers and hence is Σ_1 rather than existential. What's more it seems unlikely that there is a natural formulation of the T-predicate in $L(ET,\alpha) - \{P_0(\bar{x}), P_1(\bar{x}), \cdots\}$ that does not use bounded quantifiers and it also seems unlikely that there is a natural "T-like" predicate that works for NP-complete sets, but perhaps <u>not</u> all r.e. sets, that is formulated in $L(ET,\alpha) - \{P_0(\bar{x}), P_1(\bar{x}), \cdots\}$ without using bounded quantifiers. Nevertheless, the T-predicate is polynomially testable (or at least there are

---

4. $T(i,\bar{x},y,z) \equiv$ y is a halting computation of program i on input x with output z.

computationally natural variations of it that are), so there is a predicate symbol in L(ET) that represents $T(i,\bar{x},y,z)$. We will refer to this symbol as $P_T(i,\bar{x},y,z)$. However, it should be noted that the fact that P_T was included in L(ET) to represent T does not mean that

$$ET \vdash (\forall i,\bar{x},y,z)[P_T(i,\bar{x},y,z) \iff T(i,\bar{x},y,z)].$$

In fact it is only the universal sentences and their consequences involving P_T and various instantiations of T that are guaranteed to hold in all models of ET. It seems quite likely that by carefully examining the T-predicate one could construct models of ET that contain elements i, $\bar{x}$, y and z for which $P_T(i,\bar{x},y,z)$ but not $T(i,\bar{x},y,z)$ and vice versa.

It appears that DeMillo and Lipton's claim that ET + P = NP is consistent requires that the sets in NP be characterized by existential predicates. Since it is not known whether there are computationally natural existential predicates in L(PA) that do this, one of the ways to characterize them in an existential manner is to use the predicate symbols in L(ET). The predicate symbol P_T can be used for this purpose, and in addition it can be used to provide a notion of computation in models of ET. However, the reader should be warned now that P_T may present a very inaccurate notion of computation. Throughout the remainder of Sections 2 and 3 we will investigate the use of the predicates T and P_T as formal notions of computation in models of ET.

Since P_T provides a characterization of each r.e. set in existential form, Theorem 2.3 could be interpreted as saying that ET + "every r.e set is decidable in polynomial time" is consistent, which leads one to seriously question whether ET is rich enough to accurately capture our intuitive notions of computation. In order to understand what sets are really polynomially computable in models of ET one needs to carefully analyze models similar to those

constructed by DeMillo and Lipton. Therefore we will begin by sketching the construction of such models and the proof of Theorems 2.1 and 2.3.

DeMillo and Lipton's proof begins by expanding the theory ET to form a new theory, which we will call ET^*. This is done by adding a new constant symbol, α, to the language of ET and axioms that force $\alpha \geq i$ for each $i \in N$. Next, DeMillo and Lipton take an r.e. list $\{s_i\}$ of all the quantifier free sentences over $L(ET,\alpha)$ and in stages add to ET either s_i or $\neg s_i$ depending on which is consistent. The resulting theory, ET^*, is consistent and hence it has a model, M. However for the proof one needs to look at a submodel of M. Define $M_{D\&L} \subseteq M$ as follows,

$$M_{D\&L} = \{t_i(\alpha): t_i \text{ is a term in } L(ET)\}.$$

First, DeMillo and Lipton show that $M_{D\&L} \models ET^*$. This follows from the fact that ET^* has as axioms only universal sentences over the language $L(ET,\alpha)$ thus the only elements of a model that the axioms can assert anything about are the terms of the language applied to α, 0 or 1. Second, they show that the set,

$$\{s: s \text{ is a quantifier free sentence over } L(ET,\alpha) \text{ and } M_{D\&L} \models s\}$$

is arithmetically definable. This follows from the fact that the quantifier free sentences of ET are arithmetically definable and when the consistent quantifier free sentences over $L(ET,\alpha)$ were added to ET^* this was done in an arithmetically definable manner.

The proof of the theorem now rests on the following lemma:

2.4. Lemma (DeMillo & Lipton):

i) The standard elements of $M_{D\&L}$ are defined by a formula in $P_{D\&L}$.

ii) There is a formula $B(x) \in P_{D\&L}$ such that,

$$M_{D\&L} \models (s \iff B(\ulcorner s \urcorner))$$

for every quantifier free sentence s in $L(ET,\alpha)$.

Sketch of the Proofs:

i) Claim: $x \in N$ iff $x \leq \log^*(\alpha)$.

Since the relation "$x \leq \log^*(y)$" is polynomially testable in the standard model, there is a predicate symbol $R(x,y)$ in $L(ET)$, and hence in $P_{D\&L}$, that represents the relation. DeMillo and Lipton show that the predicate $R(x,\alpha)$ correctly characterizes the standard elements of $M_{D\&L}$. (Here we need the fourth part of the definition of $P_{D\&L}$.)

To show that $R(x,\alpha)$ iff $x \in N$ first suppose that $x = n \in N$. Then for some $m \in N$, $M_{D\&L} \models R(n,m)$. Also $M_{D\&L} \models (\forall y>m)R(n,y)$ and $M_{D\&L} \models \alpha>m$. Therefore, $M_{D\&L} \models R(n,\alpha)$. On the other hand, suppose x is nonstandard. Then $x = t_i(\alpha)$ for some term t_i in $L(ET)$. It can be shown that every unbounded term of $L(ET)$ grows faster than $\log^{(k)}$ for some standard integer k. (See DeMillo and Lipton's paper for details.) In addition this fact is expressible as a true universal sentence over $L(ET)$:

Suppose that $t_i(z) \geq \log^{(k)}(z)$ a.e. Then,

$$\forall z>m,\ 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\Big\}m > z \text{ for some } m \in N.$$

Therefore if $j = 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\Big\}m$ then,

$$M_{D\&L} \models (\forall z>m)[t_i(z)^j > z].$$

So $M_{D\&L} \models t_i(\alpha)^j = x) > \alpha$. Note also that $\exists n \in N$ such that,

$$M_{D\&L} \models (\forall z>m)(\forall y)[R(x,y) \ \&\ t_i(z)^j > y \Rightarrow y < n],$$

since $N \models \log^{(k)}(y) > \log^*(y)$ a.e. So if $y = \alpha$ and $z = x$ then,

$$M_{D\&L} \models [R(x,\alpha) \ \&\ x^j > \alpha \Rightarrow \alpha < n].$$

Therefore $M_{D\&L} \models \neg R(x,\alpha)$.

ii) We need to show that the set of all true, quantifier free sentences in $M_{D\&L}$ is defined by a predicate in $P_{D\&L}$. Since the set of all true quantifier free sentences in $M_{D\&L}$ is arithmetically definable there is a predicate,

$$B'(x) \equiv Q_1 y_1\ Q_2 y_2 \ldots Q_n y_n\ b(x,\bar{y})$$

with b quantifier free, such that

$$M_{D\&L} \models s \quad <=> \quad N \models B'(\ulcorner s \urcorner).$$

Now suppose that $a \in N$ and $\bar{c} \in N^n$. Then either $N \models b(a,\bar{c})$ and $b(a,\bar{c})$ is an axiom of ET or $N \models \neg b(a,\bar{c})$ and $\neg b(a,\bar{c})$ is an axiom of ET. Therefore if we relativize the predicate B' to N in the following manner:

replace each occurrence of

$(\exists y_i)c(x,y_i)$ by $(\exists y_i<\alpha)[y_i \in N \ \&\ c(x,y_i)]$

$(\forall y_i)c(x,y_i)$ by $(\forall y_i<\alpha)[y_i \in N \Rightarrow c(x,y_i)]$,

then, we will have formed a new predicate B(x) such that

$$N \models B'(\ulcorner s \urcorner) \quad <=> \quad M_{D\&L} \models B(\ulcorner s \urcorner) \quad <=> \quad M_{D\&L} \models s.$$

Clearly, $B(x) \in P_{D\&L}$. This proves the lemma. □

It remains to show that if A is a quantifier free predicate for a set $S = \{x: (\exists \bar{y})A(x,\bar{y})\}$, then there is a predicate $\phi(x) \in P_{D\&L}$ such that,

$$M_{D\&L} \models (\forall x)[(\exists \bar{y})A(x,\bar{y}) <=> \phi(x)].$$

Roughly, the idea is to define $\phi(x)$ to be:

$$(\exists i<\alpha)(\ \exists j<\alpha)\ [x=t_i(\alpha)\ \&\ B(\ulcorner A(t_i(\alpha),t_j(\alpha))\urcorner)\ \&\ i,j \in N].$$

This definition would work provided we could decide "$x = t_i(x)$" using a predicate in $P_{D\&L}$. DeMillo and Lipton show that this is in fact possible. Their proof involves showing that $M_{D\&L}$ can be constructed in such a way that

$$M_{D\&L} \models (x = t_i(\alpha) \iff (\forall m,k \in N)[m = x \pmod k \iff m = t_i(\alpha) \pmod k\ ]).$$

Since x = y (mod z) is a polynomially testable predicate there is a predicate symbol $P_{mod}(x,y,z) \in L(ET)$ that represents it which one can use to test whether or not $m = x \pmod k$[5]. To test whether or not $m = t_i(\alpha) \pmod k$ one can use the predicate B of Lemma 2.4. That is,

$$M_{D\&L} \models m = t_i(\alpha) \pmod k \quad \iff \quad M_{D\&L} \models B(\ulcorner m = t_i(\alpha) \pmod k \urcorner).$$

Working out the details of this argument is technically fairly tricky and since we will not use this technique again the details have been omitted. (They can be found in DeMillo and Lipton's paper.) This completes the proof of Theorems 2.1. and 2.3. □

Now that we have outlined the construction of $M_{D\&L}$ we are in a position to discuss the behavior of polynomial time predicates in models of ET.

3. The Behavior of Polynomial Time Predicates in Models of ET

If we analyze the complexity of the predicates in $P_{D\&L}$ we see that the natural Loop programs[6] for computing them consist of some standard number, n, of possibly nested loops each with index running from 0 to α, enclosing tests

---

5. Throughout DeMillo and Lipton's paper it is implicitly assumed that the predicate symbols behave properly. That is, if R(x) is a standard polynomial time testable relation and $P_R(x)$ is the symbol in L(ET) which represents it, then

$$ET \vdash (\forall x)R(x) \iff P_R(x).$$

As we have already remarked and will discuss in detail later, this is not always the case. However the predicate symbol used in the proof of Lemma 2.4 to represent $\log^*$ and the predicate symbol used here to represent mod can be shown to behave properly.

for a standard number, m, of P_i's. Since the P_i's are polynomially testable in the standard model there are standard subroutines for testing these predicates. For example a natural Loop program for the predicate $\phi_A(x)$ constructed in the proofs of Theorems 2.2 and 2.3 would look like:

```
PROGRAM φ_A(x,y):

     y ← 0;
     LOOP α;
          i ← 0;
          SubroutineLog*(xi,i);
          IF xi = 1
               THEN;
               LOOP α;
                    j ← 0;
                    SubroutineLog*(xj,j);
                    IF xj = 1
                         THEN;
                         Code for x0 ← 1 <=> x = t_i(α);
                         Code to compute gn = ⌜A(t_j(α),t_i(α))⌝;
                         SubroutineB(x1,gn);
                         IF x0 = 1 & x1 = 1;
                              THEN;
                                   y ← 1;
                              END;
                              ELSE;
                                   j ← j+1;
                              END;
                         END;
               END;
               i ← i+1;
               END;
     END;
```

SubroutineLog*(xi,i) tests whether $i \leq \log^*(\alpha)$ and if so assigns xi the value 1.

SubroutineB(x1,gn) tests whether the predicate B of Lemma 2.4 (ii) holds for gn and if so assigns x1 the value 1.

A Goedel number for the program $\phi_A(x,y)$ can be computed using some

---

6. Brainerd and Landweber define a syntax and semantics for Loop programs in [B&L-74] and show that all of the partial recursive functions are computable by Loop programs. It can also be shown that the natural time measure for Loop programs is polynomially related to the time measure for Turing machines.

standard encoding function for finite sequences. Notice that we can rewrite the program so that the element α appears only once since we can begin by assigning α to a program variable a and then use a throughout the rest ot the program. Thus,

$$\phi'_A(x,y) \equiv \begin{array}{l} a \leftarrow \alpha; \\ P_A(x,y) \end{array}$$

where $P_A(x,y)$ is the appropriately modified version of $\phi_A(x,y)$. There are reasonable encoding functions that allow one to compute the number ot the program $\phi'_A(x,y)$ from the numbers for the programs $a \leftarrow \alpha$; and $P_A(x,y)$ in such a way that the number for the program $\phi'_A(x,y)$ will equal some standard polynomial applied to α and the number for $P_A(x,y)$. If this is done, one obtains a number i_ϕ for Program $\phi'_A(x,y)$. Notice that, provided a reasonable encoding is used, the number i_ϕ is in $M_{D\&L}$. In fact for any predicate $Q(\bar{x})$ in $P_{D\&L}$ there is a number i_Q that is in $M_{D\&L}$ for a program for $Q(\bar{x})$. However this does not necessarily mean that,

$$M_{D\&L} \models (\forall \bar{x})[(\exists y)T(i_Q,\bar{x},y,1) \iff Q(\bar{x})],$$

or even that,

$$M_{D\&L} \models (\forall \bar{x})[(\exists y)P_T(i_Q,\bar{x},y,1) \iff Q(\bar{x})].$$

In fact, unless M is a model of a theory about as strong as Basic Number Theory[7] it need not even be the case that,

$$M \models (\forall \bar{x})[(\exists y)T(i_Q,\bar{x},y,1) \iff Q(\bar{x})].$$

This is because both M and $M_{D\&L}$ may only be models ot very weak theories.

---

7. Basic number theory is a sufficiently strong theory that w.r.t. polynomially time computation it proves the same sets computable as does Peano arithmetic.

Recall our earlier discussion concerning the extension of ET to form ET^* and the models M and $M_{D\&L}$: ET had as axioms all of the true universal sentences over L(ET). To form ET^* we added to ET axioms that said $\alpha \geq i$ for each $i \in N$ and more importantly for each quantifier free sentence s in $L(ET,\alpha)$ we added either s or $\neg s$ depending on which was consistent. In particular this means that for each predicate symbol $P_i(x_0,x_1,\ldots,x_n)$ in L(ET) we added

$$\text{either} \quad P_i(t_{j_0}(\alpha),\ldots,t_{j_n}(\alpha))$$

$$\text{or} \quad \neg P_i(t_{j_0}(\alpha),\ldots,t_{j_n}(\alpha))$$

for each n-tuple of terms $t_{j_0}, \ldots, t_{j_n}$. If $t_{j_0}(\alpha), \ldots, t_{j_n}(\alpha)$ are all equal to standard natural numbers then there was no choice - we added which ever one is true in N. However if some of $t_{j_0}(\alpha), \ldots, t_{j_n}(\alpha)$ are nonstandard then it is less clear - we could add either provided that it is consistent with ET and the sentences that we have already added. If we begin with an arbitrary r.e. list of quantifier free sentences it is not clear which sentences are added to ET. However it seems conceivable that the resulting theory ET^* may not be consistent with Peano arithmetic. That is, there may not be a model of Peano arithmetic, in which the predicate symbols of L(ET) are interpreted as they are intended to be, that is also a model of ET^*. If this is possible then the analysis of models of ET^* becomes all that much more difficult. To forestall these difficulties we will assume that ET^* and $M_{D\&L}$ are constructed in the following manner:

As before we begin with the theory ET and add axioms that say $\alpha \geq i$ for each $i \in N$. Next let M be a nonstandard model of the theory of Peano arithmetic + the true Π_2 sentences, which contains a nonstandard constant α. Such models can be arithmetically defined.[8] In M the function and predicate

8. The Henkin proof of the completeness theorem provides an arith-

symbols of L(ET) can be naturally interpreted. Therefore M |= ET. Now let ET^* be the set of all true universal sentences in the model M. At this point we can define $M_{D\&L}$ as before:

$$M_{D\&L} = \{x: M \models x = t_i(\alpha) \text{ for some } i \in N\}.$$

The function and predicate symbols in L(ET) are interpreted in $M_{D\&L}$ as their restriction from M. As before $M_{D\&L}$ is a model of ET^* and the true quantifier free sentences of $M_{D\&L}$, over $L(ET,\alpha)$, are arithmetically definable. In addition, with this construction, for each standard polynomial time testable relation $R(\bar{x})$, the following is true:

$$(\forall \bar{x} \in M_{D\&L})\ [M \models R(\bar{x}) \quad <=> \quad M_{D\&L} \models P_R(\bar{x})]$$

where P_R is the predicate symbol in L(ET) which represents R. Also since M is a model of T_{Π_2} (the true Π_2 sentences of arithmetic) it behaves as a model of full arithmetic with respect to polynomial time decidability.[9] Thus,

$$M \models (\forall \bar{x})[(\exists y)T(i_R,\bar{x},y,1) <=> R(\bar{x})],$$

for a standard Loop program i_R. In fact for every predicate $Q \in P_{D\&L}$ there is a Loop program similar to the one described earlier, that runs in polynomial

---

metically definable model provided the consistent set of sentences that one starts out with is arithmetically definable. The theorems of Peano arithmetic and the true Π_2 sentences of arithmetic are arithmetically definable.

9. For each standard polynomially (linearly, exponentially, etc.) decidable set there is a description of the set for which the formal statement that asserts that the set is polynomially (linearly, exponentially, etc.) decidable is a true Π_2 sentence and hence is preserved in every model of T_{Π_2}. (One can of course come up with other descriptions of the set for which one can not even prove that the set is decidable.) In addition, Peano arithmetic or T_{Π_2} is rich enough to prove that all of the predicates in $P_{D\&L}$ are polynomially computable (w.r.t. T).

time in M:

$$M \models (\forall \bar{x})(\exists y,z)[T(i_Q,\bar{x},y,z) \quad \& \text{ NumInstdescrip}(y) < p_{i_Q}(|x|)] \tag{3.a}$$

for some polynomial p_{i_Q}. (By carefully examining the programs for predicates in $P_{D\&L}$ we see that i_Q is in $M_{D\&L}$ and that p_{i_Q} is a polynomial with coefficients from $M_{D\&L}$ and exponents from N.)

Notice that with this construction of M and $M_{D\&L}$,

$$(\forall \bar{x},y,z \in M_{D\&L}) \quad M \models T(i_Q,\bar{x},y,z) \text{ iff } M_{D\&L} \models P_T(i_Q,\bar{x},y,z). \tag{3.b}$$

However, the fact that a program's runtime is bounded by a polynomial in M is still not sufficient to insure that it is computable in $M_{D\&L}$ even w.r.t. P_T. It may be the case that (3.a) and (3.b) are true but nevertheless,

$$(\exists \bar{x} \in M_{D\&L}) \quad \text{such that} \quad M_{D\&L} \models (\forall y,z)\ \neg P_T(i_Q,\bar{x},y,z).$$

This malady in fact plagues programs for some of the predicates ϕ_A of Theorems 2.1 and 2.3:

3.1. Theorem:

i) For every predicate $Q(\bar{x})$ in $P_{D\&L}$ there is a polynomial p such that $p(|\bar{x}|)$ bounds the run time of a program for $Q(\bar{x})$, w.r.t. T and P_T, in M.

ii) Nevertheless, there is a predicate $Q(\bar{x})$ in $P_{D\&L}$ that is not computable (w.r.t. P_T) in $M_{D\&L}$. When we say "not computable" we mean there does <u>not</u> exist $i_Q \in M_{D\&L}$ such that,

$$M_{D\&L} \models (\forall \bar{x})[(\exists y)P_T(i_Q,\bar{x},y,1) \iff Q(\bar{x})].$$

<u>Proofs</u>:

i) Follows from the discussion surrounding statements (3.a) and (3.b) and a detailed examination of the axioms of ET and the structure of the predicates in $P_{D\&L}$ necessary to prove these statements.

ii) We need to construct a set S that in $M_{D\&L}$ is characterized by a predicate ϕ_s which is in $P_{D\&L}$ but is not computable (w.r.t. P_T) in $M_{D\&L}$.

Suppose $S = \{x: \pi_x(x) \neq 1\}$ where π_x is the xth polynomially computable program and let S(x,y) be a standard polynomially testable relation such that

$$S(x,y) \equiv y \text{ is a computation of } \pi_x(x) \text{ with output } \neq 1.$$

Thus S(x,y) implies y witnesses the fact that $x \in S$. We will use the predicate S(x,y) to provide a formal characterization of S. Since S(x,y) is a standard polynomial time testable relation there is a predicate symbol $P_S(x,y) \in L(ET)$ that represents it. As we have already observed, for each $n,m \in N$

$$\text{either } N \models S(n,m) \text{ in which case } P_S(n,m) \text{ is an axiom of ET,}$$

$$\text{or } N \models \neg S(n,m) \text{ in which case } \neg P_S(n,m) \text{ is an axiom of ET.}$$

Also recall that for each $i,j \in N$,

$$\text{either } P_S(t_i(\alpha),t_j(\alpha)) \text{ is an axiom of } ET^*,$$

$$\text{or } \neg P_S(t_i(\alpha),t_j(\alpha)) \text{ is an axiom of } ET^*,$$

depending on which was added in our extension of ET to ET^*. If the extension was carried out as described above, then $P_S(t_i(\alpha),t_j(\alpha))$ is an axiom of ET^* iff $M \models S(t_i(\alpha),t_j(\alpha))$. Now consider the predicate,

$$\phi_s(x) \equiv (\exists i,j < \alpha)[t_i(\alpha) = x \ \& \ B(\ulcorner P_S(t_i(\alpha),t_j(\alpha))\urcorner) \ \& \ i,j \in N]$$

of Theorem 2.3. We will show that ϕ_s can not be computable w.r.t. P_T. Again, when we say computable we mean that there is a program $i_s \in M_{D\&L}$ such that

$$M_{D\&L} \models (\forall x)(\exists y,z)[P_T(i_s,x,y,z) \ \& \ (z=1 \iff \phi_s(x)].$$

Suppose that ϕ_s is computable in $M_{D\&L}$. Then there is a program $i_s \in M_{D\&L}$ and an $i \in N$ such that $i_s = t_i(\alpha)$. We will consider two cases:
First, suppose that $M_{D\&L} \models \phi_s(i_s)$. Then by definition

$$M_{D\&L} \models (\exists j<\alpha)[B(\ulcorner P_S(t_i(\alpha),t_j(\alpha))\urcorner) \ \& \ N(j)],$$

where N(j) is the predicate in $P_{D\&L}$, described in Lemma 2.4, for testing membership in N. This is true if and only if

$$M_{D\&L} \models (\exists j<\alpha)[P_S(t_i(\alpha),t_j(\alpha)) \ \& \ N(j)];$$

which implies that

$$M \models (\exists j<\alpha)[S(t_i(\alpha),t_j(\alpha)) \ \& \ N(j)].$$

Thus in M, $t_i(\alpha) = i_s \in S$. However by the definition of S, $i_s \in S \Rightarrow \pi_{i_s}(i_s) \neq 1$. That is,

$$M \models (\forall y,z)[T(i_s,i_s,y,z) \Rightarrow z \neq 1].$$

Therefore

$$M_{D\&L} \models (\forall y,z)[P_T(i_s,i_s,y,z) \Rightarrow z \neq 1].$$

But this contradicts the assumption that i_s computes ϕ_s in $M_{D\&L}$.
Therefore if ϕ_s is computable by i_s then it must be the case that

$$M_{D\&L} \models \neg\phi_s(i_s).$$

Again, by definition this means that

$$M_{D\&L} \models \neg\,(\exists j<\alpha)[B(\ulcorner P_S(t_i(\alpha),t_j(\alpha))\urcorner) \;\&\; N(j)].$$

Thus,

$$M_{D\&L} \models (\forall j<\alpha)[\neg P_S(t_i(\alpha),t_j(\alpha)) \;V\; \neg N(j)].$$

So if $j \in M_{D\&L}$ and $N(j)$ then $M_{D\&L} \models \neg P_S(t_i(\alpha),t_j(\alpha))$ which means that $M \models \neg S(t_i(\alpha),t_j(\alpha))$. Thus in M, $t_j(\alpha)$ does not witness the fact that $\pi_{i_s}(i_s) \neq 1$. This means that either,

$t_j(\alpha)$ is not a computation of $\pi_{i_s}(i_s)$

or $t_j(\alpha)$ is a computation of $\pi_{i_s}(i_s)$ but the output equals 1.

Both of these contradict the assumptions that ϕ_s is computable and that $M_{D\&L} \models \neg\phi_s(i_s)$. Thus we have shown that w.r.t. P_T, ϕ_s is not computable in $M_{D\&L}$, which completes the proof of the theorem. □

A reasonable question to raise at this point is whether Theorem 3.1 (ii) can be proved using a standard T-predicate, instead of P_T, as the notion of computation. Once again we see that the answer is dependent on whether or not the predicate P_T means what it ought to in $M_{D\&L}$. That is, if we look at the proof of Theorem 3.1(ii) we see that $M_{D\&L} \models \phi_s(i_s)$ implies that $M \models (\forall y,z)[T(i_s,i_s,y,z) \Rightarrow z\neq 1]$ which in turn implies that $M_{D\&L} \models (\forall y,z)[P_T(i_s,i_s,y,z) \Rightarrow z\neq 1]$. Now, the predicate P_T is supposed to represent the T-predicate and if it accurately does then $M_{D\&L} \models (\forall y,z)[T(i_s,i_s,y,z) \Rightarrow z\neq 1]$ which would contradict the assumptions that $M_{D\&L} \models \phi_s(i_s)$ and i_s computes ϕ_s (w.r.t. T). A similar contradiction arises if we assume that $M_{D\&L} \models \neg\phi_s(i_s)$. Thus if P_T accurately represents T

we can show that ϕ_s is not computable, w.r.t. T, in $M_{D\&L}$.

3.2. Corollary: If the predicate symbol P_T accurately represents the T-predicate in $M_{D\&L}$ then there are predicates in $P_{D\&L}$ that are <u>not</u> computable (w.r.t. T) in $M_{D\&L}$.

This result is best illustrated if we consider the set

$$K = \{i: \phi_i(i) \downarrow \}$$

for a standard programming system $\{ \phi_i \}$ and the standard predicate

$$k(i,j) \equiv j \text{ is a halting computation of } \phi_i(i).$$

Specifically, let k(i,j) be the sentence $(\exists z<|j|)T(i,i,j,z)$ for a fixed, standard, polynomially testable T-predicate for $\{ \phi_i \}$, and let P_k be the predicate in $P_{D\&L}$ that represents k. It follows from the proof of Theorem 2.3 that in the model $M_{D\&L}$

$$x \in \bar{K} \iff$$
$$\phi_{\neg k}(x) \equiv (\exists i,j<\alpha)[i,j \in N \;\&\; t_i(\alpha) = x \;\&\; B(\ulcorner \neg P_k(t_i(\alpha),t_j(\alpha)) \urcorner)].$$

Then by the argument present above, either $\phi_{\neg k}$ is not computable in $M_{D\&L}$ (w.r.t. T) or $\neg P_k(i,j)$ does not accurately represent $\neg k(i,j)$. That is, either

$$\text{not } \exists i_k \in M_{D\&L} \text{ such that } (\forall x)(\exists y,z)[T(i_k,x,y,z) \;\&\; (z = 1 \iff x \in \bar{K})]$$

or

$$\text{not } M_{D\&L} \models (\forall x,y)[\neg P_k(x,y) \iff \neg k(x,y)].$$

In essence this says that either ϕ_k is <u>not</u> computable in $M_{D\&L}$ or it is a predicate for the <u>wrong</u> set.

4. Witness Functions and Polynomial Time Decision Procedures

In this section we consider the relationship between witness functions for P ≠ NP and P ≠ EXP-time and the existence of nonstandard models of ET containing polynomial time decision procedures for problems complete in NP and EXP-time.

Suppose that a set S is decidable but not polynomially decidable, then there is a total recursive witness function w_s such that,

$$w_s(x) = i \Rightarrow \pi_x(i) \neq c_s(i),$$

where π_x is the function computed by the xth polynomial time Loop program[10] and c_s is the characteristic function for S. Typically, w_s is defined to be the minimum i such that $\pi_x(i) \neq c_s(i)$. In addition, if S is provably recursive and provably nonpolynomial (w.r.t. Peano arithmetic) then there is a total recursive witness function w_s such that

$$PA \vdash (\forall x)(\exists y)[w_s(x) = y]. \tag{4.a}$$

$$\& \quad PA \vdash (\forall x,y)[w_s(x) = y \Rightarrow \pi_x(y) \neq c_s(y)] \tag{4.b}$$

Therefore in every nonstandard model of Peano arithmetic S is not polynomially decidable.

Witness functions for distinguishing P and NP have been studied by several previous authors, [KOZ-80], [K&M-80], [ODO-79]. In particular, O'Donnell [ODO-79] observed that if P ≠ NP but Peano arithmetic + T_{Π_1} is not

---

10. Throughout this section and Section 5 we will use an enumeration of polynomial time programs with the following properties: if x is the Goedel number of the xth program then it is the result of encoding the instructions of a Loop program together with a clock which controls its runtime. Thus we can assume that the xth program runs in time bounded by $p(|y|) \leq |y|^x + x$.

adequate to prove P ≠ NP, then

$$w_{SAT}(x) = (\min i)[\pi_x(i) \neq SAT_{opt}(i)]$$

(where SAT_{opt} is the near optimal algorithm for the satisfiability problem described by Levin [LEV-72] and Schnorr [SCH-76]),

must grow faster than every provably recursive function.

Notice that in order to formally express the facts,

$$(\forall x)(\exists y)[w(x) = y] \quad \text{and} \quad (\forall x,y)[w(x) = y \Rightarrow \pi_x(y) \neq c_s(y)],$$

in the language of Peano arithmetic, one must have some formal notion of computation. That is, implicitly these statements involve a predicate which is in essence a T-predicate. Therefore when we said that (4.a) and (4.b) imply that in any model of Peano arithmetic S is not polynomially computable, we meant polynomially computable in the sense of this predicate.

Although we formulated statements (4.a) and (4.b) in terms of Peano arithmetic, they can just as well be formulated for ET or any other theory of arithmetic with the same consequences. That is,

4.1. Observation:

If there is a formula $w_s(\ )$ in L(ET) for which

$$ET \vdash (\forall x)(\exists y)[w_s(x) = y] \tag{4.1.a}$$

$$ET \vdash (\forall x,y)[w_s(x) = y \Rightarrow \pi_x(y) \neq c_s(y)] \tag{4.1.b}$$

then S is not polynomially computable in any model of ET.

In Section 2 and 3 we considered two formal notions of computation: first, the standard Kleene T-predicate and second, the notion provided by the predicate symbol P_T. In this section we will look at the use of each of these

notions in defining witness functions.

Witness functions that use P_T as their notion of computation

In Section 3 we showed that there are recursive sets which are characterized by predicates in $P_{D\&L}$ that are not polynomially computable (w.r.t. P_T) in $M_{D\&L}$. However the set constructed in the proof of Theorem 3.1 was not in NP. If we could show that in $M_{D\&L}$ there is a total witness function for an NP-complete set (assuming $P \neq NP$) then we could significantly improve upon Theorem 3.1.

We doubt that the witness functions for all NP-complete sets are total (again, assuming $P \neq NP$) in $M_{D\&L}$. However, we will show that the witness functions for certain hard sets in NP (assuming $P \neq NP$) and EXP-time are total, w.r.t. P_T, in all models of a slightly richer theory, ET(Elem), provided that they do not grow too rapidly, i.e., they are elementary.

4.2. Theorem: If S is an elementary set then S is polynomially computable (w.r.t. P_T) in some model of ET(Elem) iff

$$w_S = (\min\ y \geq x)[\pi_x(y) \neq S(y)]$$

is not bounded by any elementary function.

While this theorem does not deal solely with sets in NP or EXP-time, it does characterize, in terms of behavior of the witness functions, the circumstances under which it is consistent with ET(Elem) to believe that hard sets in NP and EXP-time are polynomially computable.

The theory ET(Elem) is an extension of ET. Our motivation for looking at it comes from remarks made by DeMillo and Lipton. In Section 5 of [D&L-80]

they discuss other theories for which Theorems 2.1 and 2.3 are provable. While it does not appear that these theorems are provable for ET(Elem), DeMillo and Lipton suggest that they may be provable for very similar theories. We will discuss this in more detail after proving Theorem 4.2.

Proof of Theorem 4.2

We begin by defining the theory ET(Elem).

4.3. Definition: Let

$$L = \{+, \cdot, -, \min(x,y), \max(x,y), c_0, c_1, c_2, \ldots$$
$$f_0, f_1, f_2, \ldots$$
$$P_0, P_1, P_2, \ldots \}$$

where

i) the c_i's are function symbols for each standard constant function; we will assume $c_i(x) = i$ for all x,

ii) the f_i's are function symbols for every polynomially honest[11], elementary function that grows faster than some iterate of the log function almost everywhere, (If f_i is an n-ary function then we will assume that for some integer k,

$$f(x_1,\ldots,x_n) > \log^{(k)}(\max\{x_1, \ldots, x_n\}) \text{ a.e.})$$

iii) and as before, the P_i's are predicate symbols for all of the standard polynomially time testable relations.

ET(Elem) is the theory of all true universal sentences over L.

---

11. A polynomially honest function is a function whose complexity is bounded by a polynomial function of its input and output.

For the proof of Theorem 4.2 we need to show that if S is an elementary set then S is polynomially computable in some model of ET(Elem) iff

$$w_s(x) = (\min\ y \geq x)[\pi_x(y) \neq S(y)]$$

is not bounded by any elementary function. We will assume that c_s is a standard Loop program which decides membership in S and we will use c_s as the formal definition of S. We begin by proving the forward implication:

We will show that if w_s is bounded by an elementary function, i.e., it is elementary, then S is not polynomially computable w.r.t. P_T in any model of ET(Elem). First, notice that statement (4.1.b) can be formulated as a true universal sentence in L if we take P_T as our notion of computation. That is, for each standard recursive set S we can construct a standard program i_w that computes the witness function for S. Therefore if c_s is a standard program for the characteristic function for S then,

$$ET(Elem) \vdash (\forall x,y_1,z_1)[P_T(i_w,x,y_1,z_1) \Rightarrow \neg(\exists y_2,z_2,y_3,z_3)[P_T(x,z_1,y_2,z_2) \ \&\ P_T(c_s,z_1,y_3,z_3) \ \&\ z_2 = z_3]]. \qquad (4.2.a)$$

This means that if i_w is total in a model M of ET(Elem), that is if $M \models (\forall x)(\exists y,z)P_T(i_w,x,y,z)$, then S is <u>not</u> polynomially <u>computable</u> in M, w.r.t. P_T. Thus it remains to show that i_w is total in all models of ET(Elem).

Assume that i_w is the Goedel number of the following Loop program for w_s:

Program $\text{witness}_s(x)$:

$z \leftarrow x$;

REPEAT

$x_1 \leftarrow$ output of $\pi_x(z)$ computed for $x|z|^x + x$ steps;

$x_2 \leftarrow$ output of the char. function for S on input z;

$z \leftarrow z+1$;

UNTIL $x_1 \neq x_2$

$i \leftarrow z-1$;

We will show that if the witness function w_s is elementary, then program witness_s is total, w.r.t. P_T, in every model M of ET(Elem).

Suppose $w_s(x) = i$, then Program witness_s iterates the REPEAT-UNTIL loop at most i times and each iteration requires $\max\{x \cdot |i|^x + x, 2^{2^{\cdot^{\cdot^{\cdot^{2^{p(|i|)}}}}}}\}$ (m twos) steps, for some fixed integer m and polynomial p which depends on S. Thus if the function $w_s(x)$ is elementary then the runtime of Program witness_s is elementary as is the function which given x produces the encoding of the computation sequence of Program witness_s on input x. This latter function is also polynomially honest and grows at least linearly. (The size of the computation sequence y is polynomial related to the time required to compute it.) Therefore there is a function symbol, f_{comp}, in L which represents this function. Also, if one has a computation sequence for witness_s then one can compute the output of witness_s using a polynomially honest function which grows faster than $\log^*$. This is because the REPEAT-UNTIL iterates at most i times and each iteration requires time $2^{2^{\cdot^{\cdot^{\cdot^{2^{p(|i|)}}}}}}$ (m twos) for large i; thus the variables values never exceed about $2^{2^{\cdot^{\cdot^{\cdot^{2^{p(|i|)}}}}}}$ (m twos). Since there are a fixed number of variables,

n, the computations sequence is an encoding of fewer than i•n numbers each less than or equal to $2^{2^{\cdot^{\cdot^{\cdot^{p(|i|)}}}}}\Big]_m$. If a reasonably efficient encoding function is used the computation sequence y will be less than $2^{2^{\cdot^{\cdot^{\cdot^{2^{p(|i|)^{(n\cdot i)}}}}}}}\Big]_m$. Therefore there is a function symbol, f_{out}, in L which represents the output function and we have that

$$M \models (\forall x)P_T(i_w, x, f_{comp}(x), f_{out}(f_{comp}(x)))$$

i.e., i_w is total in M, w.r.t. P_T. This, coupled with statement (4.2.a), shows that if w_s is elementary then S is not polynomially computable (w.r.t. P_T) in any model of ET(Elem).

For the proof of the reverse implication suppose that the witness function w_s for an elementary set S is not bounded by any elementary function. We will construct a model $M_0 \models$ ET(Elem), which is very similar to $M_{D\&L}$, and show that S is polynomially computable, w.r.t. P_T, in M_0. That is, not only is

$$M_0 \not\models (\forall x)(\exists y)[w_s(x) = y]$$

but more importantly, in M_0 there is a polynomially computable Loop program α_0 such that

$$M_0 \models (\forall x)[(\exists y_1, z)P_T(c_s, x, y_1, z) \Rightarrow (\exists y_2)P_T(\alpha_0, x, y_2, z)].$$

M_0 will be the restriction of a nonstandard model M of Peano arithmetic + ET(Elem) which contains a nonstandard constant α. We will begin by constructing M:

Since w_s is not bounded by an elementary function, for any finite set of terms $t_0, \ldots, t_i \in L$, there is an $m \in N$ such that

$$N \models w_s(m) > t_i(m).$$

This is because from $t_0, \ldots, t_i$ we can form a new term $t(x) = t_0(x) \cdot t_1(x) \cdot \ldots \cdot t_i(x)$ that is greater than any of t_0 through t_i almost everywhere, and since t is a term of L, w_s is greater than t infinitely often. Thus for each term t_i, it is consistent with Peano arithmetic + ET(Elem) to believe that $w_s(\alpha) > t_i(\alpha)$. By compactness, let M be a model of the following theory:

i) $\alpha > i$, for each $i \in N$

ii) $w_s(\alpha) > t_i(\alpha)$, for each $i \in N$

iii) Peano arithmetic

iv) ET(Elem).

We can now define $M_0 = \{x \in M: M \models x = t_i(\alpha)$ for some $i \in N\}$. The predicate and function symbols of L can be interpreted in M_0 as their restriction from M. With this interpretation M_0 is a model of ET(Elem). (Notice that we have constructed M_0 in exactly the same manner as $M_{D\&L}$ was constructed in Section 3.)

Now we need to show that S is polynomially computable in M_0. It is important to recall that

$$(\forall i,x,y,z \in M_0)[M_0 \models P_T(i,x,y,z) \iff M \models T(i,x,y,z)].$$

Since M is a model of Peano arithmetic and we constructed it in such a way that

$$M \models w_s(\alpha) > t_i(\alpha), \qquad \text{for all terms } t_i,$$

we know that in M, the αth polynomial time program computes the set S for some segment that is greater than $(\alpha, t_i(\alpha))$ for every term t_i in L. What's more we can patch program α on the segment $[0,\alpha]$ so that it decides every segment $[0, t_i(\alpha))$ correctly. The patched program, call it α_0, is not a great deal larger than α and its runtime is about the same as α's. That is, $\alpha_0 = f(\alpha)$ for some polynomially honest, elementary function, f, that grows faster than $\log^{(k)}$ for some $k \in N$. Therefore $\alpha_0 \in M_0$. Since for every i,

$$M \models (\forall y,z)[T(i_w,\alpha_0,y,z) \Rightarrow y > t_i(\alpha) \;\&\; z > t_i(\alpha)],$$

$$M_0 \models (\forall y,z)\neg P_T(i_w,\alpha_0,y,z).$$

That is, $M_0 \not\models (\forall x)(\exists z)\, w_s(x) = z$.

Now we need to show that if

$$M_0 \models P_T(c_s,x,y_1,z)$$

then there is a $y_2 \in M_0$ such that

$$M_0 \models P_T(\alpha_0,x,y_2,z).$$

In order to show that this is true we need the following facts about enumerations of polynomial time Loop programs: We can assume that program α was the αth polynomial time Loop program from some syntactically "nice" enumeration of clocked Loop programs. Thus for that enumeration we can assume that there is a polynomially honest, elementary function g(i,x) which given polynomial program i and input x, produces the computation sequence of program i run on input x. What's more we can assume g grows faster than some iterate of the log function.

Therefore, for all $x \in M_0$,

$$M \models T(c_s,x,y,z) \quad \Rightarrow \quad T(\alpha_0,x,g(\alpha_0,x),z)$$

and hence

$$M_0 \models P_T(\alpha_0,x,g(\alpha_0,x),z)$$

which completes the proof. □

At this point it is worth observing that the proof of Theorem 4.2 is not particularly dependent on the fact that we are showing that it is consistent to believe that S is polynomially computable as opposed to linearly computable. Also the construction on M_0 can simultaneously consider all elementary sets whose witness functions are not bounded by an elementary function. Hence with only minor modifications to the proof one can prove the following:

Theorem: There is a fixed model M_0 of ET(Elem) such that if S is an elementary set then S is linearly computable, w.r.t. P_T, in M_0 if

$$w_s = (\min\ y \geq x)[\lambda_x(y) \neq S(y)]$$

is not bounded by any elementary function. (Where λ is an enumeration of the linearly computable programs.)

Theorem 4.2 reformulated using the T-predicate

Suppose that we reformulate Theorem 4.2 using the standard Kleene T-predicate rather than the predicate symbol P_T, and suppose the witness function for a fixed elementary set S is elementary. Then whether or not S is really polynomially computable (that is, w.r.t. T) in a model M_0 of ET(Elem) is dependent on whether f_{comp} and f_{out} behave correctly in M_0, i.e., whether for the witness program i_w,

$$M_0 \models (\forall x)T(i_w, x, f_{comp}(x), f_{out}(f_{comp}(x))) \qquad (4.2.b)$$

and $M_0 \models (\forall x, y_1, y_2, z_1, z_2)$ (4.2.c)

$$[T(x, f_{out}(f_{comp}(x)), y_1, z_1)$$
$$\& \; T(c_s, f_{out}(f_{comp}(x)), y_2, z_2) \Rightarrow z_1 \neq z_2]$$

Essentially, f_{comp} must correctly pair together instantaneous descriptions of the computation of i_w, and f_{out} must correctly project out the output of the computation sequence $f_{comp}(x)$. So if $f_{comp}(x) = y$ and $f_{out}(y) = i$ then y must be the pairing

$$y \equiv \langle\langle ID_0\rangle, \langle ID_1\rangle, \ldots, \langle ID_m\rangle\rangle$$

where each instantaneous description ID_i is the pairing of an instruction number together with the values of the variables

$$ID_i \equiv \langle n, x, v_1, \ldots, v_n, z\rangle$$

for x the input variable, $v_1, \ldots, v_n$ program variables and z the output variable, and

$$f_{out}(y) \equiv \Pi(n+2, n+2, \Pi(m, m, y))$$

for some fixed uniform projection function Π. So essentially, f_{out} is a projection function that projects out the last element of the last instantaneous description of y.

At this point we encounter one of the major deficiencies of ET and ET(Elem) - these theories are not powerful enough to prove the existence of projection functions. Herbrand's Theorem tells us that if

$$ET(Elem) \vdash (\forall z)(\exists x)\Pi_1(z) = x$$

$$\text{and } ET(Elem) \vdash (\forall z)(\exists y)\Pi_2(z) = y$$

then

$$ET(Elem) \vdash (\forall z)[\bigvee_{i=0}^{r} \Pi_1(z) = t_i(z)]$$

$$\text{and } ET(Elem) \vdash (\forall z)[\bigvee_{i=0}^{s} \Pi_2(z) = t_i(z)].$$

This is because if we let $z = \langle x,y \rangle = (((x+1)/2) + y)^2$, then with some easy algebraic manipulation, the formulas $\Pi_1(z) = x$ and $\Pi_2(z) = y$ can be expressed in the language of ET (without predicate symbols) using only existential quantifiers, and since the terms of these theories are either almost everywhere constant or grow faster than $\log^*$, the projection functions can not be provably total. So at least on the surface there is little hope of adding projection functions to ET or ET(Elem) and still being able to prove Theorem 2.1. This is because the proof of Lemma 2.4(ii) hinges on the fact that each term of L(ET) grows faster than $\log^*$, in order to show that the standard elements of $M_{D\&L}$ are definable by a predicate in $P_{D\&L}$. Nevertheless our intuitions tell us that projection functions behave nicely enough that their presence should not interfere with this argument since they do not appear to allow one to define monotone functions which grow more slowly than $\log^*$. Adding a uniform projection function to L^+ intuitively seems much more difficult since if such a function exists then one immediately has the ability to discuss finite sequences of any length without using quantifiers. (The projection functions Π_1 and Π_2 only give one the ability to discuss finite sequences of a fixed length.) However without the ability to discuss sequences of arbitrary length it seems hopeless to try to discuss Loop program computations in any natural

way.

However if we return to the problem of showing that (4.2.b) and (4.2.c) are true in all models of ET(Elem) we see that while the existence of total uniform projection functions would certainly imply these statements, something less would also be sufficient. The programs that are mentioned in statements (4.2.b) and (4.2.c) are a concrete class of programs. We believe that i_w can be modified so that f_{out} applied to the input x rather than $f_{comp}(x)$ is a polynomially honest function which majorizes some iterate of the log function and that polynomially honest functions f_1 and f_2 which also majorize some iterate of the log, can be constructed so that

$$M_0 \models (\forall x)[T(x, f_{out}(f_{comp}(x)), f_1(x), f_2(x)) \\ \& \; T(c_s, f_{out}(f_{comp}(x)), f_1(x), f_2(x)) \Rightarrow \\ f_2(c_s, f_{out}(x)) \neq f_2(x, f_{out}(x))]$$

This would be sufficient to prove that if the witness function for S is elementary then S is not polynomially computable, w.r.t. T, in any model of ET(Elem).

The relationship between ET and ET(Elem)

The proof of Theorem 4.2 relied on the fact that L contains function symbols for all of the polynomially honest monotone functions that grow faster than some iterate of log x, in order to show that f_{comp} and f_{out} are total in all models of ET(Elem). At this point it is worth discussing why it is doubtful that Theorem 4.2 is provable for ET.

Many of the problems arise from the fact that the domain of $M_{D\&L}$, or any similar model of ET, is small in comparison with nonstandard models of full

Peano arithmetic and this severly restricts the functions that are total in $M_{D\&L}$, (regardless of the notion of computation chosen). For instance there are polynomials over the domain $M_{D\&L}$ which are not total in $M_{D\&L}$ merely because they grow too rapidly:

4.4. Example: Suppose $p(x) = x^{\alpha}$. Recall that $M_{D\&L}$ is the restriction of a model M of Peano arithmetic; therefore p(x) is total in M. However, although p(x) is a polynomial over the domain of $M_{D\&L}$ (its coefficient and exponent are in $M_{D\&L}$), it is not total in $M_{D\&L}$. This is because α^{α} is too large to be an element of $M_{D\&L}$ i.e., when we restricted M to form $M_{D\&L}$, α^{α} was not included.

The polynomial given in this example is not total in what we will call the set theoretic sense. That is, there is an $x \in M_{D\&L}$ such that for all $y \in M_{D\&L}$, $\neg p(x) = y$. Therefore p(x) can not be computable regardless of the notion of computation we select. Recall that a program i in $M_{D\&L}$ computes a total function w.r.t P_T, if for each input sequence $\bar{x} \in M_{D\&L}$ there exists a computation sequence y and an output z in $M_{D\&L}$ such that $M_{D\&L} \models P_T(i,\bar{x},y,z)$. (Alternatively, if we use the standard T-predicate to express the notion of computation then there must exist y and z such that $M_{D\&L} \models T(i,\bar{x},y,z)$.) The polynomial in Example 4.4 is not total in $M_{D\&L}$ because for some elements of $M_{D\&L}$ that are in the domain of the function, in particular α, there is no corresponding range element in $M_{D\&L}$.

There are other very simple functions which are not computable in $M_{D\&L}$ because the function which bounds their computation time is not in the model. For instance a characteristic function that has time complexity $t(x) = |x|^{x}$ can not be computable despite the fact that it is total in the set theoretic sense.

In a set theoretic sense Herbrand's theorem [Her-30] characterizes many of the relations that ET proves total. It tells us that if

$$ET \vdash (\forall \bar{x})(\exists \bar{y})\ \phi(\bar{x},\bar{y})$$

for ϕ quantifier free, then there are terms $t_0,t_1,\ldots,t_n$ of L(ET) such that

$$ET \vdash (\forall \bar{x})[\ \bigvee_{i=0}^{n}\ \phi(\bar{x},t_i(\bar{x}))\ ].$$

Essentially, this tells us that a quantifier free relation is provably total using ET if it is a term of L(ET) or is defined by cases from the terms of L(ET). Since it seems highly unlikely that the relation P_T has this property, i.e., we doubt that there are terms $t_0, \ldots, t_n$ and $s_0,\ldots,s_m$ such that

$$ET \vdash (\forall x)[\ \bigvee_{i=0,j=0}^{n,m}\ P_T(i_w,x,t_i(x),s_j(x))],$$

it seems unlikely that ET is rich enough to prove that a witness function i_w is total even w.r.t. P_T.

Theorem 2.1 reformulated for ET(Elem)

DeMillo and Lipton, [D&L-80, Thm 5.1], state that Theorem 2.1 (and hence 2.3) can be extended to any arithmetic theory T that has as axioms the true universal sentences over a first order language L(T) that contains predicate symbols for all of the standard polynomial time testable relations and function symbols for + and · and any additional function symbols provided that,

i) the terms of L(T) are r.e.,

ii) if t(x) is a term of L(T) and it is unbounded then $t(x) > \lambda(x)$ a.e., where

$\lambda(x)$ is a fixed monotone total recursive function, independent of t,

iii) for each term t_i of L(T), the relation $x = t_i(y)$ is testable in time

$$e(i,x,y) = y + 2^{2^{\cdot^{\cdot^{\cdot^{2^{(|x|+|i|)}}}}}} \Big\}^{m} \text{ for some fixed standard integer } m,$$

provided y is large enough.

Essentially, their proof shows that if conditions (i) and (ii) are met then Lemma 2.4 is provable and if condition (iii) is met then there is a predicate, $E_0(i,x,y)$, that tests $x = t_i(y)$ in time $e(i,x,y)$ from which one can inductively construct a predicate $E_n(i,x,\alpha) \in P_{D\&L}$ which, in $M_{D\&L}$, tests whether or not $t_i(\alpha) = x$.

However, as DeMillo and Lipton point out, extending L(ET) in the manner we have done in Definition 4.3 does not appear to satisfy these conditions. This is mainly because when one includes the functions $\dot{-}$ and min(x,y) and allows the f_i's to grow as slowly as the iterates of the log function, condition (iii) seems to be violated. Nevertheless, if we follow a suggestion of DeMillo and Lipton's and exclude $\dot{-}$, min and any of the f_i's that don't majorize[12] some iterate of $\lfloor\sqrt{x}\rfloor$, from the language L of Definition 4.3, then the terms of the resulting language satisfy conditions (i) - (iii):

4.5. Definition: Let $L^+ = L - \{\dot{-}, \min(x,y), f_i : f_i$ doesn't majorize an iterate of $\lfloor\sqrt{x}\rfloor\}$ and let $ET(Elem)^+$ be the theory of true universal sentences over L^+. (We use the name L^+ because all of the nonconstant functions are now increasing.)

4.6. Lemma: Conditions (i) - (iii) above hold for the terms of L^+.

Proof: Clearly, conditions (i) and (ii) hold for the terms of L^+,

12. DeMillo and Lipton merely exclude $\dot{-}$ and min however we can not prove Lemma 4.a1 unless we also restrict the f_i's.

however, it is less obvious that condition (iii) holds. Nevertheless the polynomial honesty of the functions and the fact that the nonconstant functions grow faster than some iterate of $\lfloor\bar{x}$, will allow us to prove the result:

Since the functions included in L^+ are all polynomially honest, for each base function g, either equal to an f_i or a c_i, there is a polynomial p such that $p(\max|g(y)|,|y|) > \text{Time}_g(y)$. This means that in order to decide whether $x = g(y)$ one can merely compute g(y) for $p(\max|x|,|y|)$ steps and test the output against x. Unfortunately this test handles only the functions given by function symbols rather than all of the terms, and the polynomial is not uniform but rather depends on g.

Nevertheless if we consider the structure of the terms of L^+ we see that they are formed from the function symbols by composition. For example, each m-ary function symbol g is a term and given terms $t_0(\bar{x}_0), \ldots, t_m(\bar{x}_m)$,

$$t(\bar{x}) \equiv g(t_0(\bar{x}_0), \ldots, t_m(\bar{x}_m)), \qquad \bar{x} = U\, \bar{x}_i$$

is a term formed from the function symbol g and the terms $t_0, \ldots, t_m$ by composition. What we would like to show is that t is polynomially honest provided that g and $t_0, \ldots, t_m$ are. So we will begin by supposing that there are polynomials $p_g, p_0, \ldots, p_m$ such that

$$p_g(\max |g(t_0(\bar{x}_0), \ldots, t_m(\bar{x}_m))|,|t_0(\bar{x}_0), \ldots, t_m(\bar{x}_m)|) > \text{Time}_g(t_0(\bar{x}_0), \ldots, t_m(\bar{x}_m))$$

and $p_i(\max |t_i(\bar{x}_i)|,|\bar{x}_i|) > \text{Time}_{t_i}(\bar{x}_i)$ for each $i \leq m$[13].

Then for any reasonable programming system,

13. By $|z_1,\ldots,z_n|$ we mean $|z_1| + \cdots + |z_n|$.

$$\text{Time}_t(\bar{x}) \leq \text{Time}_g(t_0(\bar{x}_0), \ldots, t_m(\bar{x}_m)) + \text{Time}_{t_0}(\bar{x}_0) + \cdots + \text{Time}_{t_m}(\bar{x}_m).$$

$$\text{Time}_t(\bar{x}) \leq p_g(\max |g(t_0(\bar{x}_0),\ldots,t_m(\bar{x}_m))|, |t_0(\bar{x}_0),\ldots,t_m(\bar{x}_m)|) \tag{4.6.a}$$

$$+ p_0(\max |t_0(\bar{x}_0)|, |\bar{x}_0|)$$

$$\vdots$$

$$+ p_m(\max |t_m(\bar{x}_m)|, |\bar{x}_m|)$$

What we would like to show is that there is a polynomial p_t such that

$$\text{Time}_t(\bar{x}) \leq p_t(\max |g(t_0(\bar{x}_0), \ldots, t_m(\bar{x}_m))|, |\bar{x}|) \tag{4.6.b}$$

If g is a constant function, that is, one of the c_i's, then (4.6.b) is clearly true. Suppose that g is not one of the c_i's. Then by the definition of L^+, g grows faster than the nth iterate of the square root function for some fixed $n \in N$. Thus,

$$g(\bar{z})^n > \max\{z_i\} \quad \text{a.e.}$$

$$n \cdot |g(\bar{z})| > \max\{|z_i|\} \quad \text{a.e.}$$

Therefore, for each term t_i,

$$n \cdot |g(t_0(\bar{x}_0)\ldots t_m(\bar{x}_m))| \geq \max\{|t_i(\bar{x}_i)|\} \quad \text{a.e.}$$

Now if we substitute into inequality 4.6.a) we obtain:

$$\text{Time}_t(\bar{x}) \;<\; p_g((m+1)\cdot n\cdot|g(t_0(\bar{x}_0)\ldots(t_m(\bar{x}_m))|)$$

$$+ \sum_{i=0}^{m} p_i(\max\{n\cdot|g(t_0(\bar{x}_0)\ldots t_m(\bar{x}_m))|,\ |\bar{x}_i|)$$

Clearly, from this expression we can construct a polynomial p_t which satisfies (4.6.b). Thus all of the terms of L^+ are polynomially honest.

Now observe that for every polynomial p the function 2^x is greater then p(x) a.e. Thus $2^{\max(|t(y)|,|y|)} > \text{Time}_t(y)$ a.e. (In particular this will be true when $y = \alpha$.) This means that there is a standard predicate, $E_0(i,x,y)$, which is computable in time $e_0(x,y) \leq 2^{(\max|x|,|y|)} \leq 2^{|x|} + y$ and provided y is large enough E_0 correctly tests whether or not $x = t_i(y)$. This predicate satisfies condition (iii). □

It follows from this lemma and DeMillo and Lipton's Theorem 5.1 [D&L-80] that Theorems 2.1 and 2.3 are provable for $ET(Elem)^+$. However when one examines DeMillo and Lipton's proof difficulties arise. There is some notational confusion throughout their paper as to when function symbols are being referred to and when terms of the language are being referred to. In the proof of their Theorem 5.1 this causes difficulties. The <u>proof</u> is clearly incorrect if only the function symbols of L(T) satisfy conditions (i) - (iii). However DeMillo and Lipton begin their proof by defining a function r(x,y):

$$r(x,y) = \begin{cases} |x|+|y|+1 & \text{if } |x|\neq|y| \text{ or } x=y \\ |x|+1 & \text{if } |x|=|y| \text{ and } x\neq y \end{cases} \qquad (4.6.c)$$

where 1 is the first bit where x and y differ. They then claim that since the function r(x,y) satisfies condition (ii) one can assume that L(T) contains a function symbol for r. While this may or may not be true for some interesting languages, if we add r to L^+ condition (iii) seems to be violated, even though

r satisfies both conditions (ii) and (iii). This is because in order to test the terms of L^+ in time $2^{|x|} + y$ we strongly rely on the fact that the decreasing functions of L^+ can not shrink an argument, x, by more than some iterate of $\sqrt{x}$. If we add r(x,y) to L^+ then arguments can be reduced by some iterate of log x and it is no longer clear that the terms of L^+ satisfy condition (iii)[14].

Although at this time we do not know how to modify DeMillo and Lipton's proof to accommodate a function symbol for r we conjecture that their theorem is true for L^+.

5. Short, Fast Programs for Initial Segments of Hard Sets

In this final section we make a few observations about the consequences if the witness function w_s for an NP-complete set S is not bounded by an elementary function.

Recall that $w_s(x)$ equals the first input greater than x on which the xth polynomial time Loop program does not agree with the set S. If w_s is not bounded by an elementary function then there are very long segments on which S is decided by a polynomial time Loop program. That is, the xth polynomial time program correctly decides S on the segment $(x, w_{s(x)})$. Since w_s grows so rapidly, there are infinitely many x for which, the segment [0, x] is quite short, $x << \log^2 w(x)$. Therefore by patching a polynomial time program that runs correctly on $(x, w_{s(x)})$ we can obtain a polynomial time program that correctly decides S on the very long initial segment $[0, w(x))$.

---

14. This also seems to be the case for $L(ET) - \{-, \min\}$. The terms of this language clearly satisfy conditions (i) - (iii), however if we add r it is no longer obvious that this is true. As in the case of L^+ this is because we can produce very complicated terms which do not grow much faster than the identity function.

More generally, we see that in order to decide which sets are polynomially computable in some models of ET(Elem), it is necessary to know which sets have infinitely many very long initial segments that are polynomially decidable by short programs. Obviously any initial segment can be quickly decided by table look-up, so we will restrict ourselves to programs whose size and time complexity is at most the log of some fixed function f applied to the length of the initial segment that it decides. If the set is polynomially decidable we will require that f be linear and if the set is NP-complete or requires exponential time we will require that the function f be polynomial. If a set has infinitely many initial segments that are quickly decidable by small programs, we will say it has "efficiently decidable initial segments:"

The question of whether there are NP-complete sets that have efficiently decidable initial segments was raised by Berman and Hartmanis, [B&H-77]. They observed that if a set is sufficiently sparse, then such programs must exist.

One can easily construct sets that require time $|x|^2$ or $2^{|x|}$ that are sparse, hence there are hard sets in P and EXP-time that have infinitely many initial segments that are decidable by short, fast programs and for which it is consistent with ET(Elem) to believe that they are linearly or polynomially decidable. Also, if we assume that $P \neq NP$, then an extension of Ladner's construction [LAD-75] allows us to obtain a noncomplete set S in NP - P that has efficiently decidable initial segments. (Essentially, one just forces the set resulting from Ladner's construction to have sufficiently long gaps.) Hence if we assume $P \neq NP$, then it is consistent with ET(Elem) to believe that certain sets, which in any model of Peano arithmetic are in NP - P, are polynomially decidable. Thus we obtain the following theorem:

5.1. Theorem:

i) There is a set $S \in P$ such that,

PA |- "S requires time $|x|^2$ to decide,"

but,

ET(Elem) + "S is linearly decidable," is a consistent theory.

ii) Assuming $P \neq NP$, there is a set $S \in NP - P$ that is not Cook complete but for which ET(Elem) + "S is polynomially decidable" is a consistent theory.

iii) There is a set $S \in$ EXP-time such that,

PA |- "S requires time $2^{|x|}$ to decide,"

but,

ET(Elem) + "S is polynomially decidable," is a consistent theory.

However this is not the case for the class of P complete (w.r.t. log-space reducibility) or EXP-time complete sets:

5.2. Theorem: It is <u>not</u> consistent with ET(Elem) to believe that polynomially complete sets are linearly decidable or that EXP-time complete sets are polynomially decidable.

<u>Proof</u>: By diagonalization one can easily construct sets requiring time $|x|^2$ or $2^{|x|}$ that do not have efficiently decidable initial segments[15] and since log-space and polynomial time reductions must map initial segments to initial segments, P complete and EXP-time complete sets can not have efficiently decidable initial segments. What's more, the diagonalization can be carried out in such a way that $w_s(x) \leq 2^{2^x}$. □

---

15. Many general results of this flavor can be found in [DEH-79] and [L,L&R-81].

Theorem 5.2 leaves open the question of whether NP-complete sets can have efficiently decidable initial segments. Not only do we believe the common conjecture that $P \neq NP$ but we also suspect that NP-complete sets do <u>not</u> have efficiently decidable initial segments, and hence that it is <u>not</u> consistent with ET(Elem) to believe that the NP-complete sets are polynomially decidable. Nevertheless a proof of this seems difficult. As might be expected, the simple diagonalization techniques used in the proofs for polynomially and EXP-time complete sets do not shed light on the question for NP complete sets.

References

[B&H-77] Berman, L., and Hartmanis, J., "On the isomorphism and density of NP and other complete sets," SIAM J. Comput. **6**, pp. 305-322 (Dec. 1977).

[B&L-74] Brainerd, W., and Landweber, L., Theory of Computation, John Wiley & Sons, New York (1974).

[D&L-79] DeMillo, R., and Lipton, R., "Some connections between mathematical logic and complexity theory," Proc. of the 11th Symp. on the Theory of Comput., pp. 153-159 (1979).

[D&L-80] DeMillo, R., and Lipton, R., "The consistency of 'P = NP' and related problems with fragments of number theory," Proc. of the 12th Symp. on the Theory of Comput., pp. 45-57 (May 1980).

[J&Y-81a] Joseph, D., and Young, P., "Independence results in computer science?," J. Comput. Systems Sci. (1981).

[J&Y-81b] Joseph, D., and Young, P., "Fast programs for initial segments and polynomial time computation in weak models of arithmetic," Proc. of the 13th Symp. on the Theory of Comput., pp. 52-61 (May 1981).

[KIR-80] Kirby, L., Private communication (1980).

[KOZ-80] Kozen, D., "Indexing of subrecursive classes," Theor. Comput. Sci. **11**, pp. 277-301 (1980).

[K&M-80] Kozen, D., and Machtey, M., "On relative diagonals," IBM Technical Report RC 8184 (April 1980).

[LAD-75] Ladner, R., "On the structure of polynomial time reducibility," J. Assoc. Comput. Machinery **22**(1), pp. 155-171 (1975).

[L,L&R-81] Landweber, L., Lipton, R.J., and Robertson, E., "On the structure of sets in NP and other complexity classes," Theoretical Comput. Sci. **15** (1981).

[LEV-72] Levin, L., "Universal enumeration problems," Problemi Peredaci Informacii **9**, pp. 115-116 (1971).

[MAH-80] Mahaney, S., "Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis," Proc. of the 21st Symp. on the Found. of Comput. Sci., pp. 54-60 (1980).

[MAN-81] Manders, K., "Computational complexity of decision problems in elementary number theory," in Proc. of the Set Theory and Hierarchy Theory Conf., Springer-Verlag, Karpacz (1981). (Lecture Notes in Mathematics Series.)

[MAT-70] Matijesevic, Y., "Enumerable sets are diophantine," Dokl Akad. Nauk SSSR **191**, pp. 279-282 (1979).

[ODO-79] O'Donnell, M., "A programming language theory which is independent of Peano Arithmetic," Proc. of the 11th Symp. on the Theory of Comput., pp. 176-188 (1979).

[SCH-76] Schnorr, C., "Optimal algorithms for self-reducible problems," pp. 322-337, in Automata, Languages and Programming, Univ. Edinburgh Press (1976).

[WIL-81] Wilkie, A., "Applications of complexity theory to sigma-zero definability problems in arithmetic," in Proc. of the Set Theory and Hierarchy Theory Conf., Karpacz (1979), Springer-Verlag (1981). (Lecture Notes in Mathematics Series.)