Independence Results about Context-Free Languages and Lower Bounds

Juris Hartmanis*

TR 84-606
May 1984

Department of Computer Science
Cornell University
Ithaca, New York 14853

*Supported in part by the National Science Foundation Grant MCS 8301766.

Independence Results about Context-Free Languages and Lower Bounds

Juris Hartmanis*
Department of Computer Science
Cornell University
Ithaca, NY 14853

Abstract

We show that for any axiomatizable, sound formal system **F** there exist instances of natural problems about context-free languages, lower bounds of computations and P versus NP that are not provable in **F** for any ***recursive representation of these problems.*** Most previous independence results in computer science have been proven for specific representations of the problems, by exploiting the "opaqueness" of Turing machine names. Our results rely on the complexity of the logical structure of the problem and yield independence results which do not depend on the representations of problems. We show, for example, that there exists a non-regular context-free language L_0 such that for no cf-grammar G, $L(G) = L_0$, is it provable in **F** that "$L(G)$ is not regular." We also show, among other results about P and NP, that there exists a recursive oracle A such that $NP^A \neq P^A$, and that this fact is not provable in **F** for any recursive representation of A.

The difference of what is and is not provable in **F** is well illustrated by questions about polynomial time isomorphisms (p-isomorphisms) of NP complete sets. We show that for every NP complete set, L, there is a representation of L by a nondeterministic polynomial time machine for which we can prove that L is NP complete. Furthermore if L is p-isomorphic to SAT then this is also provable in **F** for some representation of L. On the other hand, if there exist NP complete sets not p-isomorphic to SAT then there exists an NP complete set L for which, independent of its representation, there is no proof in **F** that L *is* or *is not* p-isomorphic to SAT.

* Supported in part by the National Science Foundation Grant MCS 8301766.

1. *Introduction*

In this paper we study instances of natural problems in computer science that can neither be proved nor disproved in formal mathematical systems. In the following let **F** be any axiomatizable, consistent and sound mathematical system in which we can prove at least the elementary facts about computation, for example Peano Arithmetic. If a statement w is provable in **F** we will write $\mathbf{F}\vdash 'w'$ or simply $\vdash 'w'$.

It is well known from Goedel's incompleteness theorem that for all such systems **F** there exists true statements that are independent of **F** , i.e., neither these statements nor their negations are provable in **F** [Ro, AHU, Ha2]. Furthermore, many natural problems in computer science dealing with formal languages and computational complexity have instances that are independent of **F** [AHU, Haj, Ha2, HH, Ko]. Many of these known independence results are derived with respect to a specific problem formulation or stated in terms of a specific Turing machine name. In a sense, these proofs of independence exploit the "opaqueness" of Turing machine names and do not necessarily remain valid for equivalent machine descriptions [AHU, Haj, HH]. For example, it is easily seen that there exists a Turing machine M_i which accepts a finite set, i.e. $L(M_i)$ is finite, but for which there is no proof in **F** of "$L(M_i)$ is finite" or "$L(M_i)$ is infinite." At the same time, there is some other TM, M_j, such that $L(M_i) = L(M_j)$ and

$$\mathbf{F}\vdash 'L(M_j) \text{ is finite}'.$$

Therefore, it is seen, that this independence result is only exploiting "bad" Turing machine names for finite sets, since for every finite set there is a representation of the set for which it is provable that it is finite.

On the other hand, we show that the situation is quite different for the question whether a recursive set is infinite, where we exploit the logical complexity of the problem and show that the independence result remains valid for all equivalent (provably total) Turing machines. More explicitly we show:

> There exists an infinite language L_o in $SPACE[n]$ such that for no provably total Turing machine M_i with $L(M_i) = L_o$, is there a proof in **F** that "$L(M_i)$ is infinite".

From this result follows, as already observed in [Haj, Ko], that there are lower bound problems for computations which are not provable in **F** for any provably recursive representation. For example there exists an L_o,

$$L_o \in SPACE[n^2] - SPACE[n].$$

such that for no n^2-space-bounded Turing machine, M_i, with $L(M_i) = L_o$, can we prove in **F** that "$L(M_i) \in SPACE[n^2] - SPACE[n]$."

Note that we do not demand that the fact $L(M_i) = L_o$ is provable in **F**.

It is very interesting to note that the same proof techniques cannot be used to show the independence of the corresponding positive result if $L_o \in SPACE[n]$. In this case there exists an n-space bounded Turing machine, M_j, such that

$$L_o = L(M_i) \text{ and } \mathbf{F} \vdash 'L(M_i) \in SPACE[n]'.$$

The difference will be seen to stem from the logical structure of these problems: the *lower bound problems* are of the logical type $\forall\exists$ and they are Π_2-complete in Kleene hierarchy, whereas the *containment* or *fast algorithm* problems are of the type $\exists\forall$ and they are Σ_2-complete [Ro, Hal]. The results of this paper show that the Π_2-type problems have representation-free independence results and that this is not the case for the Σ_2-type independence results which generally hold only for a specific problem representation.

To illustrate the application of these methods, we show that there exists a non-regular context-free language L_o such that there is no proof in **F** that

"$L(G)$ is not regular"

for any context-free grammar G with $L(G) = L_o$. Analogous results hold for context-free languages L which are not deterministic, for which $\bar{L}$ is not a context-free language and for which $\bar{L}$ is infinite.

Furthermore, these methods yield new results about P versus NP. A number of independence results about P and NP are listed in [AHU, Haj, HH, Yo]. We strengthen these results by showing that there exists a provably recursive oracle A such that

$$P^A \neq NP^A,$$

but that this fact or its negation cannot be proven in **F** for any provably total representation of A.

It has been shown in [Ko] that if $P \neq NP$ then there exists an L_0 in $NP-P$ such that for no nondeterministic polynomial time Turing machine, N_i, with $L(N_i) = L_0$ is there a proof in **F** that "$L(N_i)$ is not in P".

We observe, using results from [Haj, LLR], that if

$$\mathbf{F} \vdash \text{'}P \neq NP\text{'},$$

there for any NP-complete language L there is a representation of L by a nondeterministic polynomial time machine N such that

$$\mathbf{F} \vdash \text{'}L(N) \text{ is NP-complete'}$$

and therefore

$$\mathbf{F} \vdash \text{'}L(N) \text{ is not in P'} .$$

On the other hand, we show that if $P \neq NP$ and there exist NP-complete sets that are not polynomial time isomorphic [BH], then there will exist an NP-complete set L_0, not isomorphic to SAT, for which this fact is not provable in **F** (though we can prove in **F** that L_0 is NP-complete). At the same time, if L_1 is an NP-complete set isomorphic to SAT, then there is a representation of L_1 by a nondeterministic polynomial time machine N_1 such that

$$\mathbf{F} \vdash \text{'}L(N) \cong_p SAT\text{'}.$$

We believe that the results of this paper clarify the nature of independence results in computer science by deliniating which independence results hold for all total representations of the problem and which do not. The fact that lower bound problems are already logically so complex (Π_2-complete in the Kleene Hierarch) that, independent of their representation, there will be instances not provable in any formal system **F**, may be partially responsible for the difficulty of obtaining non-trivial results in this area. Clearly, the P and NP problem is the classic open problem of this type.

2. *Independence Results*

Let $M_1, M_2, \cdots$ be a standard enumeration of Turing machines and let $L(M_i)$ denote the language accepted by M_i. Let $|L(M_i)|$ denote the cardinality of $L(M_i)$. For a string x, $|x|$ denotes the number of symbols in x.

For any family of languages, C, we say that C has a ***recursive representation*** if there exists a recursively enumerably set of total Turing machines

$$\{M_{i_1}, M_{i_2}, \cdots\}$$

such that

$$C = \{L(M_{i_j}) \mid j \geq 1\}.$$

The next result links recursive representations and independence results.

Naming Lemma: A family of recursive sets C has a recursive representation if and only if there exists a formal system $\mathbf{F}$ in which for each L in C there is a recursive *TM*, M, such that

$$\mathbf{F} \vdash 'M \text{ is total and } L(M) \epsilon C\ '.$$

Proof: If C has a recursive representation $\{M_{i_1}, M_{i_2}, \cdots\}$ then we can take Peano Arithmetic and add the set of theorems $\{'L(M_{i_j}) \epsilon C$ and M_{i_j} is total' $\mid j \geq 1\}$ by axiomatizing the Turing machine which enumerates the recursive representation of C.

Conversely, if for a formal system $\mathbf{F}$ for each L in C there is a total *TM*, M, such that $L(M) = L$ and

$$\vdash 'M \text{ is total and } L(M) \epsilon C\ ',$$

then we can recursively enumerate the *TM's* for which the above theorem is in $\mathbf{F}$ and thus obtain a recursive representation of C. □

Note that for any standard enumeration of tape or time bounded machines (with standard tape or time shut-off mechanisms) we can prove in $\mathbf{F}$ that these machines are total. We generalizes this below.

Let $\mathbf{R} = \{R_1, R_2, \cdots\}$ be a recursive subset of total Turing machines. Then the family of languages C has an $\mathbf{R}$-*representation* iff there exists recursively enumberable subset of $\mathbf{R}$, $\{R_{i_1}, R_{i_2}, \cdots\}$, such that

$$C = \{L(R_{i_j}) \mid j \geq 1\}\ .$$

Corollary: A family of language C has an $\mathbf{R}$-representation iff there exists a formal system $\mathbf{F}$ in which for each L in C there exists an R_i in $\mathbf{R}$, such that $L(R_i) = L$ and

$$\mathbf{F} \vdash 'L(R_i) \epsilon C'.$$

Proof: Follows from Naming Lemma. □

The next result exhibits the main proof technique used in this paper.

Theorem: The family of languages $C = \{L \mid L \epsilon SPACE[n]$ and L is infinite$\}$ has no recursive representation by n-space bounded machines.

Proof: It is easily shown that the set

$$INF = \{M_i \mid L(M_i) \text{ is infinite}\}$$

is a Π_2-complete set of the Kleene hierarchy [AHU, Ro, Haj].

Let,

$$VALCOMP[M_i]$$

denote the set of valid, accepting computations of M_i [AHU, Hal] and let $M_{\sigma(i)}$, be recursively constructed machine from i, which runs on n-tape and is such that

$$L(M_{\sigma(i)}) = VALCOMP[M_i].$$

Since

$$|L(M_i)| = \infty \text{ if and only if } |VALCOMP[M_i]| = \infty,$$

we see that INF is many-one reducible to $\{M_{\sigma(i)} \mid L(M_{\sigma(i)}) \text{ is infinite}\}$.

On the other hand, if C has a recursive representation

$$\{M_{i_1}, M_{i_2}, \cdots\}$$

by n-tape bounded TM's, then we can test if $M_{\sigma(i)}$ is infinite by checking if there exists an M_{i_j} such that

$$L(M_{i_j}) = L(M_{\sigma(i)}).$$

But this test hast the form:

$$(\exists M_{i_j})(\forall x)[M_{i_j}(x) = M_{\sigma(i)}(x)],$$

where M_{i_j} and $M_{\sigma(i)}$ are total machines. Thus we see that

$$\{M_{\sigma(i)} | L(M_{\sigma(i)}| = \infty\} \epsilon \Sigma_2 .$$

Since

$$INF = \{M_i | L(M_i)| = \infty\} \leq_m \{M_{\sigma(i)} | L(M_{\sigma(i)})| = \infty\}$$

we see that

$$INF \text{ is in } \Sigma_2,$$

but this is a contradiction, since INF is Π_2-complete. Thus we conclude that C does not have a recursive representation. □

The next result shows, as already observed in [Haj, HH, Ko], that many *lower bound* problems are independent of **F**.

Corollary: There is a language

$$L_o \epsilon SPACE[n^2] - SPACE[n]$$

such that there is no proof in **F** that

$$L(M) \notin SPACE[n],$$

provided

$$L(M) = L_o \text{ and } \mathbf{F} \vdash 'M \text{ is total}'.$$

Therefore, for no standard n^2-tape bounded machine $M^{[n^2]}$ which accepts L_o, is there a proof in **F** that $L(M^{[n^2]}) \notin SPACE[n]$.

Proof: We only have to show that

$$C = \{L \mid L \text{ in } SPACE[n^2] - SPACE[n]\}$$

has no recursive representation. By diagonalization we know that $SPACE[n^2] \neq SPACE[n]$. Therefore, we can show that for n^2-space bounded TM's, $M_i^{[n^2]}$, $\Delta = \{M_i^{[n^2]} \mid L(M_i^{[n^2]}) \epsilon SPACE[n^2] - SPACE[n]\}$ is Π_2-complete. If C would have a recursive representation, then again we would show that Δ is in Σ_2, which yields a contradiction. □

From the above result it follows, that there are many lower bound problems which, independent of what provably total algorithm we use to represent them, are not provable in **F**. This fact, as already observed in [Haj, Ko], many account for difficulties of proving some lower bound problems. For other possible reasons see [HB].

3. *Independence Results about CFL's*

Next we derive some independence results about context-free languages.

For technical reasons we will assume that all of our Turing machines make at least three moves before halting. It is easily seen [AHU, Hal] that the valid computations of Turing machines establish a link with context-free languages. For any M_i there exists a context-free grammar $G_{\sigma(i)}$ [AHU, Hal] such that

$$\Sigma^* - VALCOMP[M_i] = L(G_{\sigma(i)}).$$

From this we obtain the following connections.

$L(M_i)$ is finite if and only if:

a) $L(G_{\sigma(i)})$ is regular,

b) $L(G_{\sigma(i)})$ is a deterministic cfl,

c) $\Sigma^* - L(G_{\sigma(i)})$ is a context-free language,

d) $\Sigma^* - L(G_{\sigma(i)})$ is finite.

Corollary: For any formal system **F** there exists a context-free language L_0 such that for no context-free grammar G generating L_0, can it be proved or disproved in **F** that:

a) $L(G)$ is not regular,

b) $L(G)$ is not a deterministic cfl,

c) $\Sigma^* - L(G)$ is not a cfl,

d) $\Sigma^* - L(G)$ is not finite.

Proof: Note that any of the above properties hold for

$$\Sigma^* - VALCOMP[M_i] = L(G_{\sigma(i)})$$

if and only if $L(M_i)$ is infinite. If for every infinite $L(M_i)$ there would be a grammar generating $L(G_{\sigma(i)})$ for which any of the above properties would be provable in **F**, then again we would conclude that the Π_2-complete set INF is in Σ_2. Thus none of the above properties are provable for some $L(G_{\sigma(i)})$. □

4. *Independence Results about P and NP*

Next we derive some independence results about the classes P and NP.

In [HH] it was shown that for any formal system $\mathbf{F}$ there exists an M_{i_0} such that $L(M_{i_0}) = \phi$ and there is no proof in $\mathbf{F}$ of either

$$NP^{L(M_{i_0})} = P^{L(M_{i_0})}$$

or

$$NP^{L(M_{i_0})} \neq P^{L(M_{i_0})}.$$

Again the result depends very directly on the use of the specific Turing machine name M_{i_0}. If the above result would hold for any equivalent TM, M_j, such that $\mathbf{F} \vdash 'L(M_j) = \phi'$ then we would have shown that the classic $P = NP$? problem is independent of $\mathbf{F}$, which would indeed be a startling result.

We cannot show that the classic $P = NP$? problem is independent of Peano Arithmetic, but we will prove the existence of a recursive oracle A such that

$$P^A \neq NP^A$$

and for which, for any provably recursive representation of A, $A = L(M_i)$, there is no proof in $\mathbf{F}$ that

$$"P^{L(M_i)} \neq NP^{L(M_i)}" \text{ or } "P^{L(M_i)} = NP^{L(M_i)}".$$

It should be noted, that in this construction A is not the empty set as in [HH]; see also [Haj]. We first prove a technical result which will yield the desired independence result.

Let $M_{E(1)}, M_{E(2)}, \cdots$ be a standard enumeration (i.e. recursive representation) of Turing machines with a 2^{2^n}-tape bound. Similarly, let $M_{p(1)}, M_{p(2)}, \cdots$ be an enumeration of deterministic polynomial time bounded machines. For related results see [Haj].

Corollary: $D = \{M_{E(i)} \mid NP^{L(M_{E(i)})} \neq P^{L(M_{E(i)})}\}$ is a Π_2-complete set.

Proof: D is in Π_2 since we can express D in the $\forall\exists$-form over a recursive predicate. Let $L(M^A)$ be the universal-complete language which is NP^A-complete for all oracles A. For complete universal languages see [Ha2]. Then

$$D = \{M_{E(i)} \mid (\forall j)(\exists x)[M^A_{p(j)}(x) \neq M^A(x) \text{ with } L(M_{E(i)}) = A]\}.$$

To see that D is Π_2-complete we will reduce

$$INF = \{M_i \mid L(M_i) \text{ is infinite}\}$$

to D.

In this reduction we will use the oracle T in $SPACE[2^{2^n}]$ such that for all finite sets V

$$\{1^n \mid (\exists x)[|x| = n \text{ and } x \in T-V]\} \text{ is not in } P^{T-V}$$

and therefore

$$NP^{T-V} \neq P^{T-V}.$$

Such an oracle can be constructed using standard methods from [BGS].

For each M_i we will construct an $M_{\sigma(i)}$ such that it runs on 2^{2^n}-tape and $L(M_i)$ is infinite iff $P^{L(M_{\sigma(i)})} \neq NP^{L(M_{\sigma(i)})}$.

Construction of $L(M_{\sigma(i)})$:

$M_{\sigma(i)}$ computes in stages; we let S_k denote the set accepted by $M_{\sigma(i)}$ in the first k stages.

$M_{\sigma(i)}$ starts in stage 1.

For each input x, $|x|=n$, $M_{\sigma(i)}$ uses its available tape to determine how many stages have been completed on strings up to length n. In the k-th stage $M_{\sigma(i)}$ uses its available tape to check if M_i accepts at least k strings. If this cannot be verified on available tape x is accepted iff $x \in QBF$ (valid quantified Boolean formula). If it is verified that M_i accepts at least k strings, then $M_{\sigma(i)}$ accepts x iff x is in T.

The k-th stage is completed iff $M_{\sigma(i)}$ can verify on the available tape that for all $j \leq k$,

$$L(M_{p(j)}) \neq \{1^n \mid (\exists x)[|x|=n \text{ and } x \in S_{k-1}(x)]\},$$

where $S_{k-1}(x)$ is the set accepted by $L(M_{\sigma(i)})$ of strings $y \leq x$, in a standard ordering of Σ^*.

If $L(M_i)$ is finite then $L(M_{\sigma(i)})$ differs from QBF only on a finite set and therefore

$$NP^{L(M_{\sigma(i)})} = P^{L(M_{\sigma(i)})}.$$

If $L(M_i)$ is infinite then for each k, $k \geq 1$, the k-th stage will be completed and we see that

$$NP^{L(M_{\sigma(i)})} \neq P^{L(M_{\sigma(i)})},$$

as was to be shown. □

Theorem: There exists an oracle A in $SPACE[2^{2^n}]$ such that

$$P^A \neq NP^A$$

and for no $M_{E(i)}$, $L(M_{E(i)}) = A$, can it be proven in **F** that

$$P^{L(M_{E(i)})} \neq NP^{L(M_{E(i)})}.$$

Proof: If for each A in $SPACE[2^{2^n}]$ such that $P^A \neq NP^A$ there would be an $M_{E(i)}$ such that $L(M_{E(i)}) = A$ and

$$\mathbf{F}\vdash' NP^{L(M_{E(i)})} \neq P^{L(M_{E(i)})}',$$

then the Π_2-complete set D would have a recursive representation, $M_{i_1}, M_{i_2}, \cdots$. But this would put D in Σ_2, since then

$$D = \{M_{E(i)} \mid (\exists M_{i_j})(\forall x)[x \epsilon L(M_{E(i)}) \Leftrightarrow x \epsilon L(M_{i_j})]\};$$

this is a contradiction. □

Similarly, we can show that if $P \neq NP$ then the set

$\{N_i \mid L(N_i) \epsilon NP-P\}$ is Π_2-complete, where

$N_1, N_2, \cdots$ is a standard enumeration of nondeterministic polynomial time machines. This immediately yields the known results below [Ko]. Recall also that in [LLR, Sch] it was shown that $NP-P$ had no recursive representation, if $P \neq NP$.

Theorem: If $P \neq NP$ then there exists a set A in $NP-P$ such that for no N_i, with $L(N_i) = A$, can it be proven in **F** that

$$L(N_i) \text{ not in } P.$$

Proof: Follows from the above observation and the Naming Lemma. □

It is interesting to note that the complete sets in NP have a recursive representation [LLR, Sch] and therefore we get our next result.

Theorem: If

$$\mathbf{F}\vdash' NP \neq P',$$

then for every complete set A in NP there exists an N_i, $L(N_i) = A$, such that

$$\mathbf{F}\vdash' L(N_i) \epsilon NP-P'.$$

Proof: Follows from the fact that the complete sets in NP have a recursive representation by nondeterministic polynomial time machines and that if $\vdash' NP \neq P'$ then we can prove in **F** that each of the N_{i_j} of the representation accepts a complete set in NP. For the sake of completeness we outline the construction.

Let $M_{p(j)}$, $j \geq 1$, be a standard enumeration of deterministic polynomial time transducers.

For each N_i and $M_{p(j)}$ construct a $N_{g(i,j)}$ which for input x checks in deterministic polynomial time $|x|^i$ for as many strings as possible whether condition C holds: there exists a y_1 in SAT or y_2 not in SAT such that $M_{p(j)}(y_i)$ is not in $L(N_{g(i,j)})$ or $M_{p(j)}(y_2)$ is in $L(N_{g(i,j)})$. If the above condition is satisfied then x is accepted iff x is in SAT. Otherwise, x is accepted iff x is in $L(N_i)$.

From the construction of $N_{g(i,j)}$ we see and can prove in **F**, that if condition C holds then

$$L(N_{g(i,j)}) = SAT \text{ almost everywhere,}$$

and therefore an NP-complete set. If C never holds then $M_{p(j)}$ reduces SAT to $L(N_i) = L(N_{g(i,j)})$ and again it is an NP-complete set. It is also seen that $N_{g(i,j)}$ is a recursive representation of the set of NP-complete sets.

Therefore, if

$$\mathbf{F} \vdash \text{'} NP \neq P \text{'}$$

then we have a proof in **F** that NP-complete sets are not in P and therefore, science the completeness of the $L(N_{g(i,j)})$ is provable in **F**, we conclude that for every NP-complete set $L(N_i)$ there is some representation $L(N_i) = L(N_{g(i,j)})$ for which there is a proof in

$$\mathbf{F} \vdash \text{'} L(N_{g(i,j)}) \notin P \text{'}. \qquad \Box$$

At the present it is not known whether all NP-complete sets are polynomial-time isomorphic (p-isomorphic) [BH, Ma, Yo]. In the following we prove that if there exist NP-complete sets that are not p-isomorphic, then certain properties for these sets are provable in **F** and others are independent of **F**.

As observed before if $\mathbf{F} \vdash \text{'} NP \neq P \text{'}$, then for all NP-complete sets, A, there exists a representation N_i, $L(N_i) = A$, such that

$$\mathbf{F} \vdash \text{'} L(N_i) \text{ is NP-complete'}.$$

Next we show that p-isomorphism is provable in **F**, but that non p-isomorphism is not always provable in **F**.

Theorem: If A is polynomial-time isomorphic to SAT, $A \cong_p SAT$, then there exists a representation of A, $L(N_i) = A$, such that

$$\mathbf{F} \vdash ' L(N_i) \cong_p SAT\ '.$$

Proof: A slight modification of the previous proof methods show that there exists a recursive representation by non-deterministic polynomial time machines of the set

$$\{A \mid A \cong_p SAT\}$$

and that for each N_{i_j} of this representation

$$\mathbf{F} \vdash ' L(N_{i_j}) \cong_p SAT\ '. \qquad \square$$

Theorem: If $\mathbf{F} \vdash ' P \neq NP\ '$ and there exists an NP-complete set A which is not p-isomorphic to SAT, then there exists an NP-complete set B, not p-isomorphic to SAT, such that for no representation N_i, $L(N_i) = B$, is it provable in $\mathbf{F}$ that "$L(N_i)$ is not p-isomorphic to SAT."

Proof: We observe that if all NP-complete sets are not p-isomorphic, then the set

$$\{N_i \mid L(N_i) \not\cong_p SAT\}$$

is a Π_2-complete set. To see this associate with every TM, M_i, a modified Mahaney construction [Ma] of an NP-complete set B_i, such that

$$B_i \not\cong_p SAT \Leftrightarrow L(M_i) \text{ is infinite.}$$

Mahaney's construction yields for any two non-isomorphic NP-complete sets, say SAT and A, an NP-complete set B such that B is not p-isomorphic to SAT or A. The only modification we make to this construction is to associate a TM, M_i, with the construction so that the k-th stage of the construction is initiated only after it has been verified that $|L(M_i)| \geq k$. Thus, B_i will be p-isomorphic to SAT iff $L(M_i)$ is finite.

Again, a proof in $\mathbf{F}$ for some representation N_i of each B not p-isomorphic to SAT would imply that $\{N_i \mid L(N_i) \not\cong_p SAT\}$ is in Σ_2. Since this is a contradiction, we know that there must exist NP-complete sets for which we cannot prove that they are or are not p-isomorphic to SAT, independent of their representations. $\square$

The previous result leaves the possibility open that there exist NP-complete sets that are not p-isomorphic to SAT and that this fact is not provable in Peano Arithmetic.

References

[AHU] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms,* 1974, Addison-Wesley, Reading, Massachusetts.

[BGS] T. Barker, J. Gill, and R. Solovay, "Relativizations of the $P=?NP$ Question", *SIAM Journal of Computing 4*, 1975, 431-442.

[BH] L. Berman and J. Hartmanis, "On Isomorphisms and Density of NP and Other Complete Sets", *SIAM Journal of Computing 6,* 1977, 305-327.

[GJ] M.R. Garey and D.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness,* 1979, W.H. Freeman and Co., San Francisco, California.

[Ha1] J. Hartmanis, *Context-Free Languages and Turing Machine Computations,* Proc. Symposium in Applied Math. Vol. 19, American Mathematical Society, Providence, Rhode Island, 1967, 42-51.

[Haj] P. Hajek, "Arithmetical Hierarchy and Complexity of Computation", *Theoretical Computer Science 8,* 1979, 227-237.

[Ha2] J. Hartmanis, Feasible Computations and Provable Complexity Properties, *CBMS-NSF Regional Conference Series in Applied Mathematics 30,* SIAM Monograph, 1978.

[HB] J. Hartmanis and T. Baker, "Relative Succinctness of Representation of Language and Separation of Complexity Classes", *Proc. 9th Symp. on Mathematics Foundations of Computer Science, Lecture Notes in Computer Science 74,* Springer-Verlag, 1979, 70-88.

[HH] J. Hartmanis and J.E. Hopcroft, "Independence Results in Computer Science", *ACM SIGACT News*, 8, 1976, no. 4.

[Ko] W. Kowalczyk, "Some Connections Between Representability of Complexity Classes and the Power of Formal Systems of Reasoning", to appear in *FMCS.*

[LA] R.E. Ladner, "On the Structure of Polynomial Time Reducibility," *Journal of the ACM 22,* 1975, 155-171.

[LLR] L. Landweber, R. Lipton and E. Robertson, "On the Structure of Sets in NP and Other Complexity Classes", *Theoretical Computer Science 15*, 1981, 181-200.

[Ma] S. Mahaney, "Sparse Complete Sets for NP: Solution of a Conjecture of Berman and Hartmanis," *Proceedings 21st IEEE Foundations of Computer Science Symposium*, 1980, 42-49.

[Ro] H. Rogers, *Theory of Recursive Functions and Effective Computability,* McGraw-Hill, New York, 1967.

[Sch] U. Schoning, "A Uniform Approach to Obtain Diagonal Sets in Complexity Classes", *Theoretical Computer Science 18*, 1982, 95-103.

[Yo] P. Young, "Some Structural Properties of Polynomial Reducibilities and Sets in NP", *Proc. 15 ACM Symp. Theory of Computation*, 1983, 392-401.