On the Structure of Feasible Computations

J. Hartmanis

March 1982

TR 82-484

Department of Computer Science
Cornell University
Ithaca, New York 14853

On the Structure of Feasible Computations*

J. Hartmanis
Department of Computer Science
Cornell University
Ithaca, New York 14853

Abstract

This paper discusses the study of computational complexity of feasible computations and surveys some recent insights and results about the structure of NP complete languages and the attempts to separate the classic complexity classes DLOGTAPE, NDLOGTAPE, PTIME, NPTIME, ..., Σ_k^P, ..., PTAPE.

---

* Presented at the 7th IBM Symposium on Mathematical Foundations of Computer Science, May 24, 1982, Japan.

Introduction

Looking from a historical point of view at the major developments in theory of computation and computational complexity, we can distinguish three principle phases of this development which have lead to our current research field.

1. During the 1930's the mathematical community, motivated by questions about the foundations of mathematics, investigated what is and is not effectively computable and laid the foundations for the subsequent developments in logic and recursive function theory.

2. During the 1960's, motivated by the emergence of large scale electronic computers, mathematicians and computer scientists investigated quantitative problems in computing and initiated a systematic development of the theory of computational complexity.

3. During the 1970's, following the pioneering work of S.A. Cook, the research in computational complexity theory concentrated on the central problem of what is and is not feasible (or practically) computable.

The study of feasible computations during the 1970's revealed the existence of natural complexity classes such as DLOGTAPE, NDLOGTAPE, PTIME, NPTIME and PTAPE, which could be used to classify the complexity of many practical problems. Of these families, PTIME is currently considered a good mathematical model for the feasible computations, though it is understood that not all computations in PTIME are practically computable.

The main technical tool to explore the feasible complexity classes and to classify computations is the concept of "feasible" reductions between problems. The major insight from this work is the realization that each of these families contains complete problems to which all other problems from the family can be reduced to by "feasible" reductions and that each class contains complete problems which appear naturally in computer science. The great surprise was the abundance of natural complete problems in these families, particularly NPTIME contains hundreds of complete problems which appear naturally all over computer science and many other mathematical fields. Similarly, the other natural complexity classes contain many natural complete problems. The existence of so many natural complete problem has enhanced the importance of these families to computer science and has amply justified their introduction and study. The richness and variety of the known NPTIME complete problems and their importance to computer science are documented in "Computers and Intractability. A guide to the Theory of NP-Completeness" by M.R. Garey and D.S. Johnson [GJ].

The second great surprise about the study of feasible computations is how hard some of the problems appear to be. Today, in spite of ten or more years of considerable research effort, we still cannot answer most major questions about the classes DLOGTAPE, NDLOGTAPE, PTIME, NDTIME and PTAPE. As a matter of fact, we do not yet know the proper containment relations between adjacent pairs of these families. Clearly, the outstanding problem in this area is the separation of PTIME from NPTIME. There is a strong belief that

$$\text{PTIME} \neq \text{NPTIME}$$

and that therefore the NPTIME complete problems are not feasibly computable. This belief is so deeply held that a proof that some problem is NPTIME complete is immediately accepted as a proof that the problem is not feasibly computable.

Even if the major separation problems for DLOGTAPE, NDLOGTAPE, PTIME, NPTIME and PTAPE have not yet been solved, considerable progress has been made in the study of these classes. In the following we will review some of the recent results about the structure of the problems in NPTIME and discuss the major open problems.

The Isomorphism Conjecture

Let P or PTIME designate all the languages which can be accepted by deterministic Turing machines in polynomial time in the length of the inputs. Similarly, let NP or NPTIME designate the family of languages accepted by nondeterministic Turing machines in polynomial time.

We say that a language A is NP [illegible] if and only if A is in NP and for any other language B in NP, $B \subseteq \Sigma^*$, there exists a polynomial time computable function f which reduces B to A, that is

$$(\forall x \in \Sigma^*)[x \in B \Leftrightarrow f(x) \in A].$$

As indicated earlier, there exist virtually hundreds of natural NP complete problems in computer science and other mathematical disciplines, as documented in [GJ]. Furthermore, by definition, all these problems, no matter how different they look, can be reduced to each other by polynomial time reductions. Clearly, this implies that if any one of these problems could be solved in deterministic polynomial time, then all of them could and we would know that P = NP. Currently, this looks like a very unlikely possibility and we expect that it will eventually be shown that $P \neq NP$. At the same time, this still appears not likely to happen in the near future, since we seem to lack techniques to attack this problem. We expect that we will first have to solve the easier problems of separating P from PTAPE and/or DLOGTAPE and NLOGTAPE from P and NP. We will return to these separation problems later.

A careful study of the known NP complete problems reveals that they exhibit very strong similarities, and there is the haunting feeling that it is just one problem appearing in different disguises. These considerations led L. Berman and the author in 1975 to conjecture that all NP complete problems are polynomial time isomorphic [BH].

Definition: Two languages A and B, $A \subseteq \Sigma^*$, $B \subseteq \Gamma^*$ are polynomial time isomorphic (p-isomorphic), $A \cong_p B$, if and only if there exists a bijection $f: \Sigma^* \to \Gamma^*$ such that:

1. f and f^{-1} are polynomial time computable,

2. f reduces A to B and f^{-1} reduces B to A, i.e.

$$(\forall x \in \Sigma^*)[x \in A \Leftrightarrow f(x) \in B].$$

Conjecture: All NP complete problems are p-isomorphic.

This conjecture is supported by extensive circumstantial evidence which we discuss below. As a matter of fact, for many of the classical NP complete problems the p-isomorphisms can be so constructed that the bijection even preserves the number of solutions between problems. For example, for the classic problems of satisfiability of Boolean functions in conjunctive normal form, SAT, and the problem of Hamiltonian circuits in directed graphs, there exists a p-isomorphism which maps Boolean functions with k different satisfying assignments onto graphs with k different Hamiltonian circuits and vice versa.

On the other hand, we cannot expect to solve the p-isomorphism conjecture easily, since one can simply show the following [BH].

<u>Theorem</u>: If all NP complete sets are p-isomorphic then $P \neq NP$.

Nevertheless, we were able to develop very simple conditions for an NP complete problem to be p-isomorphic to the classic NP complete problems, such as SAT [BH].

<u>Padding Lemma</u>: An NP complete problem B is p-isomorphic to SAT if and only if there exist two polynomial time computable functions S and D satisfying

$$(\forall x,y)[S(x,y) \in B \Leftrightarrow x \in B]$$
$$(\forall x,y)[D \circ S(x,y) = y].$$

One can easily convince oneself that all the well-known NP complete problems possess these padding functions and one is lead to the following observation.

<u>Fact</u>: All the "known" NP complete problems are p-isomorphic.

Here, by "known" NP complete problems, we mean the NP complete problems that have been described in the major journals or listed in [GJ]. As a matter of fact, the Padding Lemma is so easy to apply that in most cases the existence of the p-isomorphism can be derived by quick inspection of the new NP complete problem. Only one NP complete problem gave us enough difficulty to feel justified to publish the proof of the existence of p-isomorphism; this case dealt with the problem of positive solutions for a class of second order diophantine equations [HB].

So far, all attempts to construct or find an NP complete problem that is not p-isomorphic to SAT has failed, and the p-isomorphism conjecture looks like a very safe bet. Unfortunately, it could be a bet whose resolution may require considerable effort and time before it is resolved.

One attempt to disprove the p-isomorphism conjecture led to some interesting results about sparse NP complete problems, as described in the next section.

The Sparseness Conjecture

Since polynomial time computable reductions cannot map strings onto much longer strings, any two p-isomorphic languages must have roughly the same density. Thus, the p-isomorphism conjecture could be disproved if there would exist NP complete sets with radically different densities. It is easily seen from inspection of SAT or from the Padding Lemma, that the classic NP complete languages contain (roughly) exponentially many elements up to size n (i.e. $SAT \cap \Sigma^n = O(2^n)$). This observation led L. Berman and the author to conjecture that there do not exist NP-complete sets which are sparse, as stated below [BH].

Definition: A set B, $B \subseteq \Sigma^*$, is sparse if and only if there exists a k such that for all n

$$|B \cap \Sigma^n| \leq n^k + k.$$

<u>Sparseness Conjecture</u>: There do not exist sparse NP complete sets. In particular, no set A over a single letter alphabet $A \subseteq a^*$, is NP complete.

This conjecture aroused considerable interest but remained unsolved for several years until Piotr Berman provided a key idea. Berman, among other results, was able to show that if $P \neq NP$ then no set over a single letter alphabet can be NP complete [Be]. More precisely:

<u>Theorem</u>: If there exists an NP complete set A, $A \subseteq \Sigma^*$, such that $\bar{A} = \Sigma^* - A$ is sparse, then $P = NP$.

From this result it immediately followed the solution of our special conjecture.

<u>Corollary</u>: If A, $A \subseteq a^*$, is NP complete then $P = NP$.

As a matter of fact, a direct proof of this Corollary is not very difficult if one reduces SAT to A, $A \subseteq a^*$, and then uses the sparse set A as labels to prune the potentially exponential tree of partial assignments to a Boolean function. The result is a deterministic polynomial time algorithm for SAT.

The P. Berman's result provided one of the necessary techniques and the impetus for a serious attack on the sparseness conjecture. The final solution of this problem was achieved by the author's Ph.D. student S. Mahaney in 1979 [Ma]. For a review of these problems see [HM].

<u>Theorem</u>: If a sparse set A is NP complete then $P = NP$.

The proof of this result is elegant, quite involved and exploits P.

Berman's technique as well as other ideas, including some provided by R. Karp and R. Lipton [KL], which will be discussed later.

The above result has an interesting implication about NP complete problems. If there existed a sparse NP complete set, then the necessary information to solve NP complete problems up to size n would have been condensed into polynomially many strings. Unfortunately, Mahaney's result shows that if $P \neq NP$, then the necessary information to solve NP complete problems (by reduction), cannot be condensed into a sparse set.

This result shows that, under the assumption that $P \neq NP$, we cannot disprove the p-isomorphism conjectured by showing the existence of sparse NP complete sets (to which we cannot one-one reduce the classic NP complete problems). Still, the original p-isomorphism conjecture remains unsettled. Even under a strong assumption that the NP complete sets have the same density, we have not been able to prove the conjecture. Nevertheless, we strongly believe that all NP complete sets are p-isomorphic and hope that a proof of this conjecture could be obtained in the near future under the assumption that $P \neq NP$.

The above result about sparse NP complete sets is complemented by a related result about sparse oracle sets. We recall that the original work on NP complete problems used the complete NP set as an oracle set for a deterministic Turing machine to solve problems from NP in deterministic polynomial time [Co]. In this approach, the Turing machine has the option of testing the oracle for different strings. Whereas for many-one reductions, as discussed above, only one question is asked, namely, is the string resulting from the reduction in the complete set?

The following result about sparse oracles was obtained by R. Karp, R. Lipton and M. Sipser [KL].

<u>Theorem</u>: If there exists a sparse oracle set A such that $NP \subseteq P^A$, then the polynomial time hierarchy collapses to Σ_2^P, i.e.

$$\Sigma_2^P = \Sigma_{2+k}^P, \text{ for all } k \geq 0.$$

We recall that in the polynomial time hierarchy

$$\Sigma_1^P = NP \text{ and } \Pi_1^P = \text{CO-NP}$$

and the rest of the hierarchy is defined inductively by alternating polynomial size bounded existential and universal quantifiers [St].

The above result is proven under a weaker hypothesis than Mahaney's theorem but it also yields a weaker conclusion. Recall that Mahaney's result asserts that if there exist sparse NP complete sets then

$$\Sigma_k^P = \Sigma_1^P = NP, \text{ for all } k \geq 0.$$

We conjecture that the existence of sparse oracles for NP also implies that $P = NP$ and believe that this also could be proven in the near future.

A more recent result shows that if there existed two NP complete sets that are not p-isomorphic, then there must exist infinitely many different families of NP complete sets which are not p-isomorphic [Mah].

<u>Theorem</u>: If A and B are NP complete and not p-isomorphic, then there exists an NP complete set C which is not p-isomorphic to A or B.

Since we now know that there are no sparse complete sets (provided

$P \neq NP$), it is natural to ask whether there are any sparse sets in NP-P. It is known that if $P \neq NP$, then there exist incomplete sets in NP-P [La].

<u>Theorem</u>: If $P \neq NP$ then there exists a set A in NP such that A is not in P and SAT cannot be reduced to A.

Unfortunately, the delayed-diagonalization proof of this theorem yields sets which are not sparse. In essence, through the diagonalization over all P machines and all p-time reductions of SAT to A, A is constructed by alternating segments of SAT with long segments of no strings. This results in a set which is not sparse, though it can have immense gaps with no elements. So far, it is not known whether sparse sets exist in NP-P.

We conjecture that if $P \neq NP$, then there exist sparse sets in NP-P. Furthermore, we conjecture that sparse sets in NP-P can be obtained by intersecting SAT with sparse sets in P.

The last part of this conjecture is of particular interest, since a positive answer would show that the difficulty of recognizing SAT (assuming $P \neq NP$) is not caused by the variety of different Boolean functions and that even if we restrict their format (by the intersection with a sparse set in P) they are not always solvable in polynomial time.

Finally, we should mention the analogies between the polynomial time hierarchy and the well-known Kleene Hierarchy from recursive function theory [Ro]. In this analogy P corresponds to the recursive sets, NP to the recursively enumerable sets, and so on, Σ_k^P to Σ_k and Π_k^P to Π_k. Clearly, the polynomial time reductions used in the study of NP correspond to recursive reductions in the study of recursively enumerable sets. If this analogy is quite direct, then we also must expect that all NP complete sets are p-isomorphic, since the r.e. complete sets are known to be isomorphic under recursive mappings. We recall that the r.e. complete sets are the same as the creative r.e. sets and that they play a very important role in mathematics, since the sets of provable theorems of such axiomatizable mathematical systems as Peano Arithmetic and Z-F Set Theory are r.e. complete sets.

Relativization of the $P \stackrel{?}{=} NP$ Problem

The use of oracles to relativize Turing machine computations has been a very successful technique in recursive function theory and has been used extensively to classify the difficulty of computation [Ro]. The oracle set A can be queried by the Turing machine by writing a string x on its oracle tape and entering a special query state; the answer is yes if x is in A, no otherwise.

There has also been some very interesting work on relativized versions of P and NP. We write P^A and NP^A, respectively, for the family of languages that can be accepted in deterministic and nondeterministic polynomial time with the oracle set A. It can be easily seen that, if we set A=SAT, then

$$NP \cup CoNP \subseteq P^{SAT}.$$

The initial work in this area was motivated by the hope of gaining insights in the $P \stackrel{?}{=} NP$ problem by the study of the relativized version of this question. At least this was the author's hope when he suggested this approach to his Ph.D. student T. Baker in 1974. Baker surprised many by showing that the classic $P \stackrel{?}{=} NP$ problem could be relativized to make the relativized P and NP sets either equal or different. The same result was obtained independently by J. Gill and R. Solovey and the results were published jointly [BGS].

<u>Theorem</u>: There exist recursive sets A and B such that

$$P^A = NP^A \text{ and } P^B \neq NP^B.$$

The first part of the result is not that surprising because we can choose an oracle A which overwhelms the computational power of P and NP and thus makes the relativized versions equal (let A be a PTAPE complete set).

The second part of the result is more surprising and it relies directly on NP's ability to guess whether there exists a string of length n in the oracle set. Since P proceeds deterministically it can ask the oracle only polynomially many different questions and an elegant diagonalization argument exploits this difference. The resulting oracle set B is so constructed that the set

$$D = \{a^n \mid (\exists x)[\,|x| = n \text{ and } x \in B]\}$$

is in NP^B, as can easily be seen, but is not in P^B because of the diagonalization (for any deterministic polynomial time machine M_i there is an n_i such that $x \in B$, $|x| = n_i$ iff M_i rejected a^{n_i} and vice versa. Because M_i can query only polynomial many strings of length n_i, it always gets the answer no and if it accepts a^{n_i}, no string of length n_i is added to B; if not, a string of length n_i which has not been queried by M_i is added to B.)

This result seemed to indicate that relativization is not going to help much with the classic separation problem and that we must deal directly with the specific combinatorial problems of P and NP. The result also seems to indicate that a straight diagonalization argument cannot solve the separation problem since such an argument must work also with oracle sets A and B, which is not possible. See also [KM].

These early impressions of the meaning of the Baker-Gill-Solovey result may have been too pessimistic. Subsequent investigation of other aspects of relativization of the separation problem have provided further insights and promise a better understanding of nondeterminism.

A very interesting and difficult result was obtained by C. Bennett and J. Gill about the frequency of oracles for which $P^A \neq NP^A$ [BG]. They showed that for randomly chosen oracles A $P^A \neq NP^A$ with probability one. This result clearly indicates that the oracles B for which $P^B = NP^B$ are somewhat unusual. Unfortunately, we have no idea whether the empty set is an unusual oracle and thus cannot draw any direct conclusions about the classic separation problem.

Let us now return to the original relativization result and look at the proof that there exists an oracle B such that

$$P^B \neq NP^B.$$

The construction of B is such that the set

$$D = \{a^n \mid (\exists x)[\,|x| = n \text{ and } x \in B]\}$$

is not in P^B because any of the exponentially many strings of length n could be in B and any deterministic p-time machine can query only polynomially many of these strings. On the other hand, a nondeterministic p-time machine exhibits here its full power of nondeterminism by just guessing a string of length n; if there is a string of length n in B, one of the possible nondeterministic computations will find it and D is seen to be in NP^B.

These and related considerations suggest that one should further explore the relation of the access mechanism to the oracles for deterministic and nondeterministic computations. This exploration can proceed by restricting the nondeterministic access to the oracle or increasing the power of access of the deterministic computations.

The first phase of this work, probably guided by a somewhat different motivations as described above, was initiated by R. Book and his collaborators. They showed [SMB] that if the nondeterministic machines were very severely restricted in their access to the oracle then

$$P = NP \Leftrightarrow (\forall A)[P^A = NP^A].$$

This work was generalized by T. Long and we outline this result [Lo].

Let $NPQB^A$ be the family of languages accepted by nondeterministic Turing machines with oracle set A and which are such that for any x the machine can query the oracle for only polynomially many strings (i.e. number of queries $\leq |x|^k+k$, for a fixed k for each machine and all x).

<u>Theorem</u>:

$$P = NP \text{ iff } (\forall C)[P^C = NPQB^C].$$

This result clearly shows that the power of nondeterminism for relativized computations stems from the ability to ask unrestricted questions of the type: "is there an x of length n in C?". Later we will discuss what happens if we endow P computations with such query power.

The above result also shows that if one could find any possible (even noncomputable) set C such that

$$P^C \neq NPQB^C,$$

then we would know that $P = NP$. Unfortunately, a close inspection of the proof of this result suggests that this is not likely to succeed and that one may as well work on the classic separation problem with $C = \emptyset$.

The following result, obtained by my Ph.D. students J. Kadin and V. Sewelson, shows that the previous theorem is about as strong as it can be made.

Let $NP[f]^B$ denote the languages accepted by a nondeterministic p-time machine which for each x asks no more than $f(|x|)$ oracle questions.

<u>Theorem</u>: Let f be a function such that for all k

$$\lim_{n\to\infty} \frac{n^k}{f(n)} = 0.$$

Then there exists an oracle B for which

$$P^B \neq NP[f]^B.$$

From this result we see that if the nondeterministic computations can ask more than polynomially many questions, then there exist oracles which separate deterministic and nondeterministic polynomial time oracle computations. Looking at the original proof of the Baker-Gill-Solovey separation result, we see that the question which cannot be asked deterministically is "does there exist an x of length n in the oracle set?".

To further explore the nature of nondeterminism the author and his Ph.D. student J. Kadin investigated deterministic p-time computer models with enhanced access to the oracle set. To our surprise we found that permitting the deterministic polynomial machine to ask if there exists a string of length n in the oracle, made it incompatible to the nondeterministic polynomial time computations for different oracles.

More precisely, let PL^A denote the family of languages accepted by deterministic p-time Turing machines which by writing a^n on its oracle tape and entering a special state can ask if there exists a string of length n in A.

<u>Theorem</u>: There exist sets A, B and C such that

$$PL^A = NP^A, \; NP^B - PL^B \neq \emptyset \text{ and } PL^C - NP^C \neq \emptyset.$$

This result shows clearly, that though the original separation proof [BGS] for P and NP does not work for the separation of PL from NP, the proof can be modified to derive a B such that $PL^B \neq NP^B$; the proof exploits the fact that PL^A, in essence, cannot distinguish between one or two strings of length n in A. The proof that $PL^B - NP^B \neq \emptyset$ exploits the fact that there are oracles that separate NP from Co-NP and that PL^B is closed under complements.

The question asking power can be increased considerably and still the deterministic and nondeterministic computations can be oracle separated. For example, let PS^A be the family of languages accepted in deterministic polynomial time by Turing machines which can write on their oracle tape strings of the form

$$1*0** \cdot \cdot \cdot *0$$

and ask the oracle if there exists a string in A matching the specified symbols and having an arbitrary symbol for each star. Note that these machines can perform binary search and for each n find in polynomially many steps a string of length n in A, if it exists.

<u>Theorem</u>: There exist sets A, B and C such that

$$PS^A = NP^A, \; NP^B - PS^B \neq \emptyset \text{ and } PS^C - NP^C \neq \emptyset.$$

Finally, under the assumption that $NP^A = Co - NP^A$ we can show that

$$PL^A \subsetneq NP^A \text{ and } PS^A \subsetneq NP^A.$$

We are currently continuing the investigation of different models of deterministic p-time machines with enhanced oracle access and their relation to nondeterministic oracle machines. Again, the overall impression is that the nondeterministic access to oracles is so different from the various enhanced deterministic models that we are dealing with two radically different phenomenon.

So far the work on the relativization of P and NP has not been able to suggest any direct approaches to the solution of this classic separation problem, but it has supplied further, circumstantial evidence which supports the conjecture that $P \neq NP$. It has also raised a variety of new and interesting questions which should be investigated further and whose solution should yield insights about nondeterminism and feasible computations in general.

Conclusion

There is not much doubt that one of the most important problems in theoretical computer science is the problem of understanding feasible computations and, in particular, resolving the containment relations between DLOGTAPE, NLOGTAPE, P, NP and PTAPE. What is not so clear is why the separation problems are so hard. More than a decade of hard and concentrated effort has provided many and deep insights in this research area, but no major progress on the separation problem.

We believe that part of the reason for this is that no set of mathematics techniques is directly applicable to the problems in computational complexity and that, therefore, we will have to first develop the tools for this field of research and only then can we hope to solve the major problems. Furthermore, we believe that the separation problems in general, and the $P \stackrel{?}{=} NP$ problem in particular, are fundamental problems whose importance extends far beyond computer science. To illustrate this, recall that in essence the P and NP problem is a problem about the difference between *creating* and *checking*. More precisely, the quantitative difference, if any, between P and NP is the difference between *finding* a mathematical proof (or *creating* it) and *checking* that it is a correct proof. It is hard to imagine that there would not be a substantial difference between these activities and, therefore, in a sense, the P and NP separation problem is the fundamental problem about creativity.

To make the above considerations mathematically precise, consider a standard formal mathematical system such as Peano Arithmetic or Z-F Set Theory. Let us pick Peano Arithmetic. Then the set T of theorems in

P.A. followed by their formal proofs i.e.

$$T = \{(x, p_x) \mid x \text{ is a theorem with proof } p_x \text{ in PA}\},$$

is in P (actually, T is a complete set of P). On the other hand, we can also easily show that the set

$$T' = \{(x, a^n) \mid x \text{ is a theorem in PA with a proof of length n or less}\}$$

is NP complete. Thus, the difficulty of finding proofs (of a specified length) is characterized by NP completeness and the difficulty of checking a proof is characterized by P completeness.

These considerations not only lend importance to the classic separation problems but further emphasize the need to thoroughly understand the nature and structure of NP complete problems: what makes these problems hard? with what frequency are they hard? is there only one type of hard problem? etc.

It is our sincere hope that during this decade some of these problems will be solved. In particular, we believe that some of the extreme separation problems, such as

$$\text{DLOGTAPE from P, NP, or } \Sigma_k^P$$

should be intensively attacked and could possibly be solved in the near future.

A reasonable miniseparation problem which deserves consideration is the problem of showing that there <u>is no</u> sparse oracle S such that

$$\text{PTAPE} \subseteq \text{DLOGTAPE}^S.$$

One can show that if such a sparse oracle exists, then there exists an equivalent sparse oracle S' which is PTAPE computable and for which also,

$$\text{PTAPE} \subseteq \text{DLOGTAPE}^{S'} .$$

Clearly, this is very unlikely to be true, particularly if we recall that there do not exist sparse PTAPE complete sets under LOGTAPE reductions [Ha]. For more background on these problems see [HS].

References

[Be] Berman, P. "Relationships between density and deterministic complexity of NP complete languages," Fifth International Symposium on Automata, Languages and Programming, Udine, Italy. Lecture Notes in Computer Science 62, 1978, pp. 63-71.

[BG] Bennett, C.H., and J. Gill. "Relative to random oracle A, $P^A \neq NP^A \neq \text{Co-}NP^A$ with probability 1," SIAM Journal on Computing 10:1, 1981, pp. 96-113.

[BGS] Baker, T., J. Gill, and R. Solovay. "Relativizations of the $P \stackrel{?}{=} NP$ question," SIAM J. Computing 4:4, 1975, pp. 431-442.

[BH] Berman, L., and J. Hartmanis. "On isomorphisms and density of NP and other complete sets," SIAM J. Computing 6:2, 1977, 305-322.

[Co] Cook, S.A. "The complexity of theorem proving procedures," Proceedings Third Annual ACM Symposium on the Theory of Computing, 1971, pp. 151-158.

[GJ] Garey, M.R., and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness, H. Freeman, San Francisco, 1978.

[Ha] Hartmanis, J. Feasible Computations and Provable Complexity Properties, Society of Industrial and Applied Mathematics, Philadelphia, 1978.

[HB] Hartmanis, J., and L. Berman. "On polynomial time isomorphisms of some new complete sets," Journal of [illegible] and System Sciences 16, 1978, pp. 418-422.

[HM] Hartmanis, J., and S. Mahaney. "An essay about research on sparse NP complete sets," Proceedings 9th Symposium on Mathematical Foundations of Computer Science, 1980. Lecture Notes in Computer Science 88, 1980, pp. 40-57.

[HS] Hartmanis, J., and J. Simon, "On quasi-homomorphisms and the problem of separating LOGSPACE from other complexity classes." In preparation.

[KL] Karp, R., and R. Lipton. "Some connections between nonuniform and uniform complexity classes," Proceedings 12th ACM Symposium on Theory of Computation, 1980, pp. 85-103.

[KM] Kozen, D., and M. Machtey. "On relative diagonals," Journal of Computer and System Sciences, to appear.

[La] Ladner, R.E. "On the structure of polynomial time reducibility," J. ACM 22:1, 1975, pp. 155-171.

[Lo] Long, T. "Relativized nondeterministic time (preliminary draft)," Department of Computer Science, Ohio State University, Columbus, Ohio, December 1981.

[Ma] Mahaney, S. "Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis," Proceedings 21st IEEE Annual Symposium on Foundations of Computer Science, 1980, pp. 54-60.

[Mah] Mahaney, S. "On the number of p-isomorphism classes of NP-complete problems," Proceedings 22nd IEEE Symposium on Foundations of [illegible] Science, 1981, pp. 271-278.

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

[SMB] Selman, A., X. Mei-rui, and R. Book. "Controlled relativization of complexity classes," draft, 1981.

[St] Stockmeyer, L.J., "The polynomial time hierarchy," Theoretical Computer Science 3:1, 1976, pp. 1-22.