Theoretical Computer Science 68 (1989) 267-276
North-Holland

RELATIVIZING RELATIVIZED COMPUTATIONS*

Neil IMMERMAN
Computer and Information Science, University of Massachusetts, Amherst, MA 01003, U.S.A.

Stephen R. MAHANEY
Computer Science Dept., University of Arizona, Tucson, AZ 8572, U.S.A.

Communicated by Paul Young
Received June 1986
Revised June 1988

Abstract. This paper introduces a technique of relativizing already relativized computations and gives two interesting applications. The techniques developed here are simpler than the usual methods for constructing oracles that satisfy several requirements simultaneously. The first application shows that a result of Karp and Lipton (if sets in NP are decidable with polynomial-size circuits, then $\Sigma_2^P = \Pi_2^P$) cannot be strengthened in the presence of certain oracles. This means that relativizable proof techniques cannot strengthen the conclusion to, say, P = NP. Such a stronger conclusion would be desirable as it would establish the equivalence of polynomial-time programs and polynomial-size circuits for solving NP-complete problems and would extend the known equivalence of polynomial-time programs and programs that are allowed a single query to a polynomial-size table. The second application gives an oracle C for which $P^C \neq (NP^C \cap coNP^C) \neq NP^C$ and $NP^C \cap coNP^C$ has complete sets under polynomial-time many-one reductions. This complements a result of Sipser in which an oracle B is constructed for which $NP^B \cap coNP^B$ has no complete sets. These results suggest that current proof methods will not settle whether NP $\cap$ coNP has complete sets.

1. Introduction

Relativization [1] is a method of introducing additional information into models of computation. In its simplest form a Turing machine (TM) computation may write an *oracle query string* onto a designated tape; it may enter a *query state* and then will enter one of two states indicating whether or not the oracle query string is in the *oracle set A*. Analogous to the classes P and NP, the relativized computational complexity classes P^A and NP^A denote sets accepted in deterministic (resp. nondeterministic) polynomial-time by TMs that use the oracle set A.

* Results in Section 3 of this paper first appeared in [8] at the Santa Barbara Conference on Computational Complexity Theory in 1983.

Typical uses of relativization are in [1] where oracles A and B are constructed that satisfy requirements $P^A = NP^A$ and $P^B \neq NP^B$. The interpretation of these results follows from noting that conventional proof methods in complexity (diagonalization, simulation, etc.) apply equally well to relativized computations. Thus, a conventional proof of $P = NP$ implies that for all oracles B, $P^B = NP^B$. This contradicts the result cited above. A conventional proof of $P \neq NP$ has similar consequences. The conclusion of this is that settling the P versus NP problem will require new proof techniques that do not relativize.

The construction of oracles that satisfy more than one type of requirement has typically been done by complex arguments that alternate among the requirements. These arguments are difficult when requirements are present that are satisfied by coding; i.e. making assignments to the oracle for all inputs. The oracle assignments to satisfy such a requirement may interfere with other assignments needed to satisfy different requirements.

This paper introduces a method to apply relativization to already relativized complexity classes. It has generally been overlooked that relativization can be applied to a class of computations, even if that class is already relativized. Our model of relativizing NP^{A_1} is $NP^{A_1+A_2}$ where A_1+A_2 denotes the set of strings $0A_1 \cup 1A_2$. With this method we can begin with an oracle A_1 that establishes one requirement, then add another oracle A_2 to establish a second requirement, and so on. (The resulting $NP^{A_1+A_2}$ may not satisfy the requirements established in NP^{A_1}. The results here actually use this property.) The successive oracles can be constructed without concern for *how* the previous requirements were satisfied. This method permits us to obtain simple proofs of some new and some known oracle results. The reason our methods are simpler is that we can satisfy one requirement at a time and thus recycle existing constructions. Examples can best illustrate the ease of this method.

For example, we consider relativized classes such as P^A and NP^A. We show that the familiar oracle construction methods (e.g. [1, 2]) can be applied to these classes of relativized computations. Thus, for any oracle A we can use the construction of [1] to obtain a set S such that $P^{A+S} \neq NP^{A+S}$. When these methods are applicable, the oracle constructions are much simpler than single constructions that meet multiple requirements.

Our first application of these methods shows that there are relativizations in which a result of Karp and Lipton regarding the consequences of polynomial-size circuits for NP cannot be substantially improved. In [9], they showed that if there is a sparse set S such that $NP \subseteq P^S$ (which is equivalent to assuming sets in NP have polynomial-size circuits), then $\Sigma_2^P = \Pi_2^P$. We apply our methods to show that there is an oracle A and sparse set S such that $NP^A \subseteq P^{A+S}$ (informally, NP^A has polynomial-size circuits with gates that query A; see [14]), but the results of Karp and Lipton cannot be improved from $\Sigma_2^{P,A} = \Pi_2^{P,A}$ to $\Sigma_2^{P,A} = \Sigma_1^{P,A}$.

Our method first uses an oracle C such that $P^C = NP^C$; the oracle constructed in [1] suffices. Then we construct an oracle S with polynomial-size circuits T such that $\Sigma_2^{P,C+S} \neq \Sigma_1^{P,C+S}$; the method of [2] with minor modification is sufficient. We

treat T as a sparse set of strings describing the circuits as in [3]. We then show that the sparse oracle, $M(T)$, of prefixes of strings in T is sufficient to establish $\mathrm{NP}^{C+S} \subseteq \mathrm{P}^{C+S+M(T)}$.

Since these results were obtained, Wilson [14] and Heller [6] independently constructed oracles that establish a stronger nonimprovability of the Karp and Lipton result. Both [14] and [6] exhibit oracles X such that $\mathrm{NP}^X \subseteq \mathrm{P}^{X+S}$ where S is sparse, but $\Sigma_2^{\mathrm{P},X} \neq \Delta_2^{\mathrm{P},X}$.

Our second application complements [12] in which Sipser presented an oracle in which $\mathrm{NP}^A \cap \mathrm{coNP}^A$ has no complete sets. We present a simple construction of an oracle B in which $\mathrm{NP}^B \cap \mathrm{coNP}^B$ has complete sets and $\mathrm{NP}^B \cap \mathrm{coNP}^B$ is neither equal to P^B nor equal to NP^B. (Thus the complete sets are not trivially the complete sets of P^B nor those of NP^B.) This result was previously obtained by Hartmanis and Immerman [5], but the proof here is much simpler.

2. Definitions

We assume familiarity with basic definitions in computational complexity as found in [7] including P, NP, PSPACE, polynomial-time many-one and Turing reductions, complete sets and relativization. The polynomial-time hierarchy (Σ_i^{P}, Π_i^{P} and Δ_i^{P}) is defined in [13]. We use QBF to denote the set of quantified boolean formulas which is PSPACE-complete [7]. Less familiar definitions follow.

Definition 2.1. A set S is *sparse* if there is a polynomial $p(n)$ such that the number of elements in S of length n is at most $p(n)$.

Definition 2.2. We will use $A+B$ to denote $0A \cup 1B$ (corresponding to a disjoint union of two oracle sets.) For definiteness, $A+B$ is left associative. With oracle $A+B$ the query $x \in A$ means $0x \in A+B$.

Definition 2.3. A set S has *polynomial-size circuits* if there is a polynomial $p(n)$, a family $\{C_n\}$ of strings such that $|C_n| \leq p(n)$, and a polynomial-time computable *circuit evaluator* predicate $E(x, C)$ such that for all $x \in \Sigma^*$ of length up to n, $x \in S$ if and only if $E(x, C_n)$ evaluates to *true*. The strings C_n can be thought of as circuits for S. This is known to be equivalent to S being polynomial-time Turing reducible to a sparse set T [3].

We define complete sets under many-one and Turing reductions for the relativized setting as follows.

Definition 2.4. We say that B is *polynomial-time Turing complete* for NP^A if B is in NP^A and $\mathrm{NP}^A \subseteq \mathrm{P}^{A+B}$. We say B is *polynomial-time many-one complete* for NP^A if B is in NP^A and for all C in NP^A, there is a function f that is polynomial-time

computable *relative to the oracle A* such that f reduces C to B. For classes such as $NP^B \cap coNP^B$ we define complete sets similarly.

3. Polynomial size circuits for NP

3.1. Overview and motivation

Sparse sets are used in computational complexity as an alternative to P, deterministic polynomial time, for a model of feasible computability. Note that polynomial-time Turing reducibility to a sparse oracle corresponds to solvability with polynomial-size circuits and that polynomial-time many-one reducibility to a sparse set corresponds to solvability by look-up in a small table [3].

Two recent results show that if computational problems are reducible to such small amounts of information, then there are strong consequences for complexity classes.

Theorem 3.1 [9]. *If sets in* NP *are polynomial-time Turing reducible to a sparse oracle* S (*equivalently* $NP \subseteq P^S$), *then* $\Sigma_2^P = \Pi_2^P$.

Theorem 3.2 [11]. *If* NP*-complete sets are polynomial-time many-one reducible to a sparse set* S, *then* $P = NP$.

Note that the hypothesis of Theorem 3.1 is equivalent to sets in NP being solvable by polynomial-size circuits [3]. The hypothesis of Theorem 3.2 is that the NP-complete sets are solvable by a single look-up in a polynomial-size table.

The second result has not only a stronger conclusion than the first, it also gives a precise characterization of a nonstandard computational model. If we view Theorem 3.2 as addressing the question of whether the model of computing with a single look-up in a polynomial-size table is more powerful than deterministic polynomial time, we see that, measured by their power to recognize NP-complete sets these models are equivalent; the table look-up gives no advantage. Theorem 3.1 has no such interpretation and it is natural, therefore, to attempt to strengthen the conclusion for the sake of a similarly precise characterization of the power of the polynomial-size circuit model.

Another advantage of Theorem 3.2 is that we obtain a precise answer to the question of whether there is hope of solving NP-complete problems with table look-up methods. If those more general methods can work, then $P = NP$, so we might as well seek ordinary algorithms for these problems. Theorem 3.1 and a general belief that the polynomial-time hierarchy does not collapse indicates that polynomial-size circuits for NP-complete problems are unlikely to exist; however, lacking a stronger conclusion such as $P = NP$, we cannot rule out the advantage of their existence as we can in the case of Theorem 3.2.

We show here by methods of relativization [1, 2] that our present methods are unlikely to strengthen the conclusion of Theorem 3.1. Proofs are in the next section.

Theorem 3.3. *There is an oracle A and a sparse oracle S such that* $\mathrm{NP}^A \subseteq \mathrm{P}^{A+S}$ *(equivalently,* $\mathrm{NP}^A <_T^{\mathrm{P},A} S$*), but* $\mathrm{P}^A \neq \mathrm{NP}^A$.

Theorem 3.4. *There is an oracle A and a sparse oracle S such that* $\mathrm{NP}^A \subseteq \mathrm{P}^{A+S}$ *(equivalently,* $\mathrm{NP}^A <_T^{\mathrm{P},A} S$*), but* $\Sigma_2^{\mathrm{P},A} \neq \mathrm{NP}^A$ $(=\Sigma_1^{\mathrm{P},A}$ *by definition*).

It follows from Theorems 3.3 and 3.4 that improving the consequence of Theorem 3.1 to, say, $\mathrm{P} = \mathrm{NP}$ or $\mathrm{NP} = \Sigma_2^{\mathrm{P}}$, will require new proof methods that do not relativize. The following two results show that Theorems 3.1 and 3.2 remain true in the presence of oracles. The proofs in [9] and [11] relativize directly, so we will not repeat the details. For both results it is interesting to note that the combinatorial methods in the proofs remain valid with oracles.

Theorem 3.5. *For any oracle A, if sets in* NP^A *are polynomial-time Turing reducible to a sparse oracle (even with reductions that use the oracle A:* $<_T^{\mathrm{P},A}$*), then* $\Sigma_2^{\mathrm{P},A} = \Pi_2^{\mathrm{P},A}$.

Theorem 3.6. *For any oracle A, if sets in* NP^A *are polynomial-time many-one reducible (even with reductions that use the oracle A:* $<_m^{\mathrm{P},A}$*) to a sparse set, then* $\mathrm{P}^A = \mathrm{NP}^A$.

Theorems 3.4 and 3.5 and the stronger versions of Wilson and Heller indicate that Theorem 3.1 seems to be the best possible result with present techniques.

3.2. Proofs

In this section we prove Theorems 3.3 and 3.4.

Theorem 3.7 [1]. *For $C = \mathrm{QBF}$ or any other* PSPACE*-complete set,* $\mathrm{P}^C = \mathrm{NP}^C$.

The following illustrates our observation that many relativization methods carry over without modification to relativized classes.

Theorem 3.8. *For any oracle C, there exists a sparse oracle S so that* $\mathrm{P}^{C+S} \neq \mathrm{NP}^{C+S}$.

Proof. The proof is a straightforward adaptation of results in [1]. Let

$$L^1(S) = \{1^n : \exists x |x| = n \text{ and } x \in S\}.$$

Recall that the construction in [1] separates P^S and NP^S by "hiding" strings $x \in S$ from the deterministic computations of sets in P^S. We observe that this construction can be applied directly to hiding strings from the deterministic computations of P^{C+S}. Thus, $L^1(S)$ will be in NP^{C+S} but not in P^{C+S}. □

A sparse set S has short, easily decoded descriptions of its elements. Explicitly, the elements of S of length up to n can be coded into a single string, s_n of length polynomially bounded in n. (The polynomial depends on the polynomial bounding the number of strings in S.)

A more general property than sparseness is for a set S to have *polynomial-size circuits.* Suppose S has circuits C_n for elements of length up to n; the C_ns are encoded as strings whose length is $p(n)$, a polynomial. (Shorter strings can be padded as needed.) The set $T = \{C_n : n \geq 0\}$ consisting of the circuits for S is sparse.

Definition 3.9. For T a sparse set in Σ^* we define $M(T)$, the *map* of T, to be padded prefixes of elements of T:

$$M(T) = \{x = p\#^k : \exists w \text{ with } pw \in T \text{ and } |pw| = |x|\},$$

where $\#$ is a new symbol.

Since T is sparse, $M(T)$ is also sparse. It is clear from the coding that if a set S has polynomial-size circuits, then these circuits can be encoded in a sparse set T, and the circuits can be reconstructed by a deterministic polynomial-time TM using $M(T)$ as an oracle.

Theorem 3.10. *Suppose S has polynomial-size circuits, $T = \{C_n : n \geq 0\}$.*

(a) *If* $\mathrm{NP}^C = \mathrm{P}^C$, *then* $\mathrm{NP}^{C+S} \subseteq \mathrm{P}^{C+S+M(T)}$.

(b) *If* $\mathrm{NP}^C \cap \mathrm{coNP}^C = \mathrm{P}^C$, *then* $\mathrm{NP}^{C+S} \cap \mathrm{coNP}^{C+S} \subseteq \mathrm{P}^{C+S+M(T)}$.

Proof. (a) Let M^{C+S} be a nondeterministic Turing machine running in polynomial time $q(n)$. We will show that M's computations with oracle $C+S$ can be simulated by a deterministic machine D with oracle $C+S+M(T)$.

For an input x to M^{C+S} with $|x| = n$, M can query strings of S of length at most $q(n)$. Let $C_{q(n)}$ be the small circuit coded in $M(T)$ which describes this part of S. Then $\mathrm{M}^{C+S}(x)$ can be simulated by a nondeterministic polynomial-time machine $\mathrm{M}_2^C(x, C_{q(n)})$ in which queries to S are answered by decoding the circuit in the input and simulating the circuit on the query.

Since M_2^C defines an NP^C language, there is an equivalent deterministic polynomial-time machine D_2^C witnessing this language is in P^C. We can now define $\mathrm{P}^{C+S+M(T)}$ machine D to decide acceptance of x by M^{C+S}. The machine D first uses $M(T)$ to compute $C_{q(n)}$ and then runs $\mathrm{D}_2^C(x, C_{q(n)})$.

The proof of part (b) is similar. □

We now note that Theorem 3.3 follows as a corollary of Theorems 3.7, 3.8 and 3.10. To obtain Theorem 3.4 we adapt the arguments in [2] where an oracle is constructed that separates $\Sigma_2^{\mathrm{P},A}$ and NP^A. The following result modifies this construction to the relativized setting and insures that the oracle has polynomial-size circuits. Theorem 3.4 is then immediate.

Theorem 3.11. *There is an oracle A such that $\Sigma_2^{P,A} \neq NP^A$ and A has polynomial-size circuits.*

Proof. We adapt a construction from [2]. Let

$$L^2(A) = \{x: (\exists u |u| = |x|)(\forall v |v| = |x|) uv \in A\}.$$

We will build an A which has polynomial-size circuits and satisfies

$$L^2(A) \in \Sigma_2^{P,A} - NP^A.$$

We will build A in stages A_i which successively establish the requirement that $L^2(A)$ is not the language accepted by M_i^A where M_i is the ith nondeterministic oracle machine running in polynomial time $p_i(n)$. Let $A_0 = \emptyset$.

Let n_i be large enough that $2^{n_i} > p_i(n_i)$ and no strings of length $2n_i$ have been queried or put into A_i in previous stages. Let

$$S_i = A_{i-1} \cup \{0^{n_i}x: |x| = n_i\}.$$

Consider a run of $M_i^{S_i}$ on the input 0^{n_i}. If M_i rejects, then, letting $A_i = S_i$, we have met the ith requirement. Otherwise, $M_i^{S_i}$ accepts 0^{n_i}. Choose a string w such that $0^{n_i}w$ is not queried in the computation. Then, letting $A_i = S_i - \{0^{n_i}w\}$, we have again satisfied the ith requirement. Note that since $2^{2n_i} > p_i(n_i)$, the computations that satisfy the previous requirements are not affected.

Finally, let $A = \cup A_i$. Then A satisfies all the requirements. To see that A has polynomial-size circuits, observe that we need only describe the values n_i for which the oracle has strings of length $2n_i$ in it; the presence of a value w for the case of the omitted string and, in that case, the value w. Clearly, there is a sparse oracle to represent this. Thus A has polynomial-size circuits. □

4. Complete sets for NP ∩ coNP

The following theorem due to Sipser establishes the possibility under relativization that NP ∩ coNP has no complete sets. A simpler proof is due to Li.

Theorem 4.1 [12, 10]. *There is an oracle A such that $NP^A \cap coNP^A$ does not have a complete set.*

In this section we establish the opposite possibility using our simple methods; there is an oracle C such that $NP^C \cap coNP^C$ has complete sets. Of course we want our oracle to satisfy $P^C \neq NP^C \cap coNP^C \neq NP^C$, lest the complete sets arise for trivial reasons. As before, the oracle is built in steps.

Theorem 4.2 [1]. *There exists an oracle A such that*

$$P^A = NP^A \cap coNP^A \neq NP^A.$$

Theorem 4.3. *Let A be any oracle such that*

$$P^A = NP^A \cap coNP^A \neq NP^A.$$

Then there is a sparse oracle D such that

$$P^{A+D} \neq NP^{A+D} \cap coNP^{A+D} \neq NP^{A+D}.$$

Proof. We will construct a set of lengths, $S = \{n_1, n_2, \ldots\}$ such that S in unary, $\{0^n : n \in S\}$, is recognizable in $DTIME^{A+D}[n]$. We will simultaneously construct D to contain a unique string of length n for each $n \in S$. For a string w let last(w) denote the last bit of w. It will follow that the language defined by

$$Z(D) = \{0^n : n \in S \ \& \ \text{last}(w) = 0 \text{ for the unique } w \in D \text{ of length } n\}$$

has both NP^{A+D} and $coNP^{A+D}$ characterizations:

$$Z(D) = \{0^n : n \in S \ \& \ (\exists w)(|w| = n \ \& \ w \in D \ \& \ \text{last}(w) = 0)\}$$

$$Z(D) = \{0^n : n \in S \ \& \ (\forall w)(|w| = n \ \& \ w \in D \rightarrow \text{last}(w) = 0)\},$$

thus it is in $NP^{A+D} \cap coNP^{A+D}$. We will construct D so that $Z(D) \notin P^{A+D}$. We will also assure that S is sufficiently sparse to maintain $NP^{A+D} \neq coNP^{A+D}$.

Construction: Let U^{A+D} be a complete set for NP^{A+D}. Let D_k be the set consisting of the first k strings of D. Let $M_1, M_2, \ldots$ be a listing of all clocked deterministic polynomial-time TMs where the runtime of each M_i is bounded by some polynomial $p_i(n)$. Let $N_1, N_2, \ldots$ be a similar listing of clocked $p_i(n)$ time-bounded nondeterministic TMs. The construction will satisfy the following requirements:

$C_1(i)$ The ith NP^{A+D} machine N_i does not recognize $\overline{U^{A+D}}$,
$C_2(i)$ M_i^{A+D} does not recognize $Z(D)$.

The conditions $C_1(i)$ will assure the separation of NP^{A+D} and $coNP^{A+D}$. The conditions $C_2(i)$ will assure that $Z(D)$ is not in P^{A+D} and thus $P^{A+D} \neq NP^{A+D} \cap coNP^{A+D}$.

Define D_0 and let $n_0 = 1$. We define n_i to be the smallest natural number greater than n_{i-1} such that

(a) n_i steps a fixed deterministic TM. M^{A+D} can check that requirements $C_1(1), C_2(1), \ldots, C_i(i)$ are satisfied by computations of length less than $\log(n_i)$, with D_{i-1} substituted

(b) $2^{n_i} > 2p_i(n_i)$.

Note that such an n_i always exists because D_{i-1} is finite and thus the oracle A guarantees that $NP^{A+D_{i-1}} \neq coNP^{A+D_{i-1}}$. Also, the above definition assures that S is recognizable in $DTIME^{A+D}[n]$. Finally, n_i is so much larger than n_{i-1} that the addition to D of a string of length n_i will not affect the witnesses that meet previous requirements.

Define D_i as follows: examine the computation of $M_i^{D_{i-1}}$ on input 0^{n_i}. Choose a w such that $|w| = n_i$, w is not queried by $M_i^{A+D_{i-1}}$, and last(w) = 0 if and only if

$M_i^{A+D_{i-1}}$ rejects 0^{n_i}. Such a w must exist since (b) ensures that fewer than one half of the strings of length n_i may be queried. Let $D_i = D_{i-1} \cup \{w\}$. This ensures that every condition $C_2(i)$ is met.

Thus we have

$$P^{A+D} \neq NP^{A+D} \cap coNP^{A+D} \neq NP^{A+D}. \quad \square$$

Corollary 4.4. *There is an oracle $A+D$ such that*

$$P^{A+D} \neq NP^{A+D} \cap coNP^{A+D} \neq NP^{A+D}$$

and such that there is a sparse set S which is complete for $NP^{A+D} \cap coNP^{A+D}$ under polynomial-time Turing reductions.

Proof. Let A be as in Theorem 4.3 and let D be constructed as above. Then by application of Theorem 3.10(b),

$$NP^{A+D} \cap coNP^{A+D} \subseteq P^{A+D+M(D)}.$$

Also $M(D)$ is in $NP^{A+D} \cap coNP^{A+D}$. Thus, $M(D)$ is a sparse complete set for $NP^{A+D} \cap coNP^{A+D}$ under polynomial-time Turing reductions. $\square$

Hartmanis and Immerman [5] showed that $NP \cap coNP$ has a complete set under polynomial-time Turing reductions if and only if it has a complete set under polynomial-time many-one reductions. It is not hard to see that their proof goes through when relativized to any oracle C.

Theorem 4.5. *For any oracle C there is a complete set for $NP^C \cap coNP^C$ under polynomial-time Turing reductions if and only if there is a complete set under polynomial-time many-one reductions.*

Applying Theorem 4.5 to Corollary 4.4 we have the following.

Corollary 4.6. *There is an oracle C for which*

$$P^C \neq NP^C \cap coNP^C \neq NP^C$$

and $NP^C \cap coNP^C$ has complete sets under polynomial-time many-one reductions.

Acknowledgment

We thank Joan Feigenbaum, Al Aho and two referees for numerous helpful comments. N. Immerman was supported by an NSF Postdoctoral Fellowship. Parts of this research were carried out while both authors were at the Mathematical Sciences Research Center, in Berkeley, CA.

References

[1] T. Baker, J. Gill and R. Solovay, Relativizations of the P = ?NP question, *SIAM J. Comput.* **4** (1975) 431-442.

[2] T. Baker and A. Selman, A second step toward the polynomial hierarchy, *Theoret. Comput. Sci.* **8** (1979) 177-187.

[3] L. Berman and J. Hartmanis, On isomorphism and density of NP and other complete sets, *SIAM J. Comput.* **6** (1977) 305-322.

[4] P. Berman, Relationship between density and deterministic complexity of NP-complete languages, in: *Fifth International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science **62** (Springer, Berlin, 1978) 63-71.

[5] J. Hartmanis and N. Immerman, On complete problems for NP$\cap$coNP, in: *Twelfth International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science **194** (Springer, Berlin) 250-259.

[6] H. Heller, On relativized exponential and probabilistic complexity classes, *Inform. and Control* **71** (1986) 231-243.

[7] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, Reading, MA, 1979).

[8] N. Immerman and S. Mahaney, Oracles for which NP has polynomial size circuits, Conference on Computational Complexity Theory, Santa Barbara, March (1983) 89-93.

[9] R. Karp and R. Lipton, Some connections between nonuniform and uniform complexity classes, Proc. 12th ACM Symposium on Theory of Computing (1980) 302-309.

[10] M. Li, Lower bounds in computational complexity, Ph.D. Thesis, Cornell University, Ithaca, NY (1985).

[11] S. Mahaney, Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis, *J. Comput. Systems Sci.* **25** (1982) 130-143.

[12] M. Sipser, On relativization and the existence of complete sets, *Ninth International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science **140** (Springer, Berlin) 523-531.

[13] L.J. Stockmeyer, The polynomial time hierarchy, *Theoret. Comput. Sci.* 3(1) (1974) 1-22.

[14] C. Wilson, Relativized circuit complexity, *J. Comput. Systems Sci.* **31** (1985) 169-181.