RELATIVIZING RELATIVIZED COMPUTATIONS

Neil Immerman and Stephen R. Mahaney

YALEU/DCS/TR-501
November 1986

Relativizing Relativized Computations[*]

Neil Immerman[†‡]
Computer Science Department
Yale University
New Haven, CT 06520

Stephen R. Mahaney[‡]
AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974

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 $\Sigma_2^P = \Pi_2^P$ [KL80].) cannot be strengthened in the presence of certain oracles. This means that current proof techniques cannot strengthen the conclusion to, say, $P = NP$ [BGS75]. 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 [M82].

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 [S82] 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.

---

[*] Results in section 3 of this paper first appeared in [IM83] at the Santa Barbara Conference on Computational Complexity Theory in 1983.
[†] Supported by an NSF Postdoctoral Fellowship and NSF Grant DCR-8603346.
[‡] Parts of this research were carried out while both authors were at the Mathematical Sciences Research Center, in Berkeley, CA.

1. Introduction

Relativization [BGS75] 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 TM's that use the oracle set A.

Typical uses of relativization are in [BGS75] 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. We interpret this to mean that settling the P vs. NP problem will require new proof techniques that do not relativize.

The construction of oracles that satisfy more than one requirement has typically been done by complex arguments that alternate among the requirements. These arguments are difficult because the infinitely many assignments to the oracle that are made to satisfy one requirement may interfere with assignments needed to satisfy a different requirement.

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. 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 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 than these techniques is that we can satisfy one requirement at a time and, indeed, 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. [BGS75], [BS79]) can be applied to these classes of relativized computations. Thus, for any oracle A we can use the construction of [BGS75] to obtain a set S such that $P^{A+S} \neq NP^{A+S}$. ($A+S$ denotes the set of strings $0A \bigcup 1S$.) 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 [KL80], 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 A circuits;' see [W85]), but the results of Karp and Lipton cannot be improved to $\Sigma_2^{P,A} \subseteq \Sigma_1^{P,A}$.

Our method first uses an oracle C such that $P^C = NP^C$; the oracle constructed in [BGS75] 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 [BS79] with minor modification is sufficient. We treat T as a sparse set of strings describing the circuits as in [BH77]. We then show that the sparse oracle $M(T)$ of prefixes of strings in T is sufficient to establish $NP^{C+S} \subseteq P^{C+S+M(T)}$.

Since these results were obtained, Heller [H84] and Wilson [W85] independently constructed oracles that establish a stronger non-improvability of the Karp and Lipton result. Both [H84] and [W85] exhibit oracles X such that $NP^X \subseteq P^{X+S}$ where S is sparse, but $\Sigma_2^{P,X} \neq \Delta_2^{P,X}$.

Our second application complements [S82] in which Sipser presented an oracle in which $NP^A \bigcap coNP^A$ has no complete sets. We present a simple construction of an oracle B in which $NP^B \bigcap coNP^B$ has complete sets and $NP^B \bigcap coNP^B$ 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 [HI85], but the proof here is much simpler.

2. Definitions

We assume familiarity with basic definitions in computational complexity as found in [HU79] including *P*, *NP*, *PSPACE*, polynomial-time many-one and Turing reductions, complete sets, and relativization. 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 size n is at most $p(n)$.

Definition 2.2. We will use $A+B$ to denote $0A \bigcup 1B$ (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 size 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 [BH77].

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 Turing reducibility to a sparse oracle corresponds to solvability with polynomial-size circuits and that polynomial many-one reducibility to a sparse set corresponds to solvability by 'look-up' in a small table [BH77].

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. [KL80] 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. [M82] If NP-complete sets are polynomial-time many-one reducible to a sparse set S, then $P = NP$.

The second result has not only a stronger conclusion, it also gives a precise characterization of a nonstandard computational model. Note that the hypothesis of Theorem 3.1 is equivalent to sets in *NP* being solvable by polynomial-size circuits [BH77]. The hypothesis of Theorem 3.2 is that the *NP*-complete sets are solvable by a single 'look-up' in a polynomial-size table.

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 computational model of polynomial-size circuits.

Another advantage of Theorem 3.2 is that we get 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 $P = NP$ and we might as well seek ordinary algorithms for these problems. Theorem 3.1 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 [BGS75, BS79] that our present methods are unlikely to strengthen the conclusion of Theorem 3.1.

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

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

It follows from Theorems 3.3 and 3.4 that improving the consequence of Theorem 3.1 to, say, $P = NP$ or $NP = \Sigma_2^P$, will require new proof methods that do not relativize. Note that Theorems 3.1 and 3.2 remain true in the presence of 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: $<_T^{P,A}$), then $\Sigma_2^{P,A} = \Pi_2^{P,A}$.

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: $<_m^{P,A}$) to a sparse set, then $P^A = NP^A$.

Theorems 3.4 and 3.5 indicate that Theorem 3.1 seems to be the best possible result with present techniques. For both results it is interesting to note that the combinatorial methods in the proofs remain valid even with oracles present.

3.2. Proofs

In this section we prove Theorems 3.3 and 3.4.

Theorem 3.7. [BGS75] For $C = QBF$ or any other *PSPACE*-complete set, $P^C = NP^C$.

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

Theorem 3.8. For any oracle C, there is a sparse oracle S so that

$P^{C+S} \neq NP^{C+S}$.

Proof. The proof is a straightforward adaptation of results in [BGS75]. Let

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

Recall that the construction in [BGS75] 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}. QED

A sparse set S has short, easily decoded descriptions of its elements. Explicitly, the elements of S of size up to n can be coded into a single string, s_n of size 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 size up to n; the C_n's are encoded as strings whose size 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 sparse also. 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 $M(T)$ from which the circuits can be reconstructed by a deterministic polynomial time machine.

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

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

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

Proof of part (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 size at most $q(n)$. Let $C_{q(n)}$ be the small circuit coded in $M(T)$ which describes this part of S. Then $M^{C+S}(x)$ can be simulated be a nondeterministic polynomial-time machine $M_2^C(x, C_{q(n)})$ 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 for a language in P^C. We can now define a $P^{C+S+M(T)}$ 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 $D_2^C(x, C_{q(n)})$.

The proof of part (b) is similar. QED

We now note that Theorem 3.3 follows as a corollary of Theorem 3.10. To obtain Theorem 3.4 we adapt the arguments in [BS79] where an oracle is constructed that separates $\Sigma_2^{P,A}$ and NP^A. We modify this construction to insure 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 [BS79]. 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

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

Let $A_0 = \varnothing$. At stage i, the construction will satisfy the condition 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 n_i be large enough that $2^{n_i} > p_i(n_i)$ and no strings of size n_i have been queried or put into A_i in previous stages. Let

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

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 condition. 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 condition.

Finally, let $A = \cup S_i$. Then A satisfies all the conditions. Note that we need at most $n+1$ bits of information to describe all the strings of size n in A. Thus A has polynomial-size circuits. QED

4. Complete sets for NP $\cap$ coNP

The following theorem due to Sipser establishes the possibility under relativization that $NP \cap coNP$ has no complete sets.

Theorem 4.1. [S82] 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 conclusion 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. [BGS75] 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 sizes, $S = \{ n_1 < n_2 < \cdots \}$ 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 size 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

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

Then $Z(D)$ has both NP^{A+D} and $coNP^{A+D}$ characterizations:

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

so it is in $NP^{A+D} \cap coNP^{A+D}$. We will construct D so that $Z(D) \notin P^{A+D}$.

Construction.

Let SAT^{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, $\cdots$ be a listing of all clocked deterministic polynomial-time TM's where the runtime of each M_i is bounded by some polynomial $p_i(n)$. The construction satisfies the following conditions:

$C_1(i)$: The i^th^ $coNP^{A+D}$ machine does not recognize SAT^{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 $P^{A+D} \neq NP^{A+D} \bigcap coNP^{A+D}$.

We define n_i to be the smallest natural number such that

(a): Conditions $C_1(1)$, $C_1(2)$, ..., $C_1(i)$ are all witnessed by computations of length less than $\frac{n_i}{i}$, with D_{i-1} substituted for D.

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

Note that such n_i always exist because D_{i-1} is finite and thus the oracle A guarantees that $NP^{A+D_{i-1}} \neq coNP^{A+D_{i-1}}$.

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, 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 1/2 of the strings of size n_i may be queried. Let $D_i = D_{i-1} \bigcup \{w\}$. This ensures that every condition $C_2(i)$ is met.

Thus we have

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

QED

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

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

and such that there is a sparse set S which is complete for $NP^{A+D} \bigcap coNP^{A+D}$ 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),

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

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

Hartmanis and Immerman [HI85] showed that $NP \cap coNP$ has a complete set under polynomial-time, Turing reductions if and only 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

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.

5. Acknowledgement.

We thank Joan Feigenbaum and Al Aho for numerous helpful comments.

6. REFERENCES

[BGS75] T. Baker, J. Gill and R. Solovay, 'Relativizations of the $P =? NP$ question,' SIAM J. Comput., vol 4, (1975) 431-442.

[BS79] T. Baker and A. Selman, 'A second step toward the polynomial hierarchy,' Theor. Comput. Sci. 8, (1979) 177-187.

[B78] 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, Vol. 62, Springer-Verlag, Berlin, (1978) 63-71.

[BH77] L. Berman and J. Hartmanis, 'On isomorphism and density of *NP* and other complete sets,' SIAM J. Comput. 6 (1977) 305-322.

[H84] H. Heller, 'On relativized exponential and probabilistic complexity classes,' manuscript 1984.

[HI85] J. Hartmanis, and N. Immerman, 'On complete problems for $NP \cap coNP$,' 'Twelfth International Colloquium on Automata, Languages and Programming,' Lecture Notes in Computer Science, Vol. 194, Springer-Verlag, Berlin, 250-259.

[HU79] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation,* Addison Wesley, 1979.

[IM83] N. Immerman and S. Mahaney, 'Oracles for which *NP* has Polynomial Size Circuits,' Conference on Computational Complexity Theory, Santa Barbara, March 1983, 89-93.

[KL80] R. Karp and R. Lipton, 'Some connections between nonuniform and uniform complexity classes,' Proc. 12th ACM Sympos. on Theory of Computing,

(1980) 302-309.

[L82] T. Long, 'A note on sparse oracles for *NP*,' J. Comput. Systems Sci. 24, (1982) 224-232.

[M82] S. Mahaney, 'Sparse complete sets for *NP*: solution of a conjecture of Berman and Hartmanis,' J. Comput. Systems Sci. 25, (1982) 130-143.

[S82] M. Sipser, 'On relativization and the existence of complete sets,' 'Ninth International Colloquium on Automata, Languages and Programming,' Lecture Notes in Computer Science, Vol. 140, Springer-Verlag, Berlin, 523-531.

[W85] Wilson, C. 'Relativized circuit complexity,' J. Comput. Systems Sci. 31, (1985) 169-181.