SIAM J. COMPUT.
Vol. 13, No. 4, November 1984

EQUIVALENCE RELATIONS, INVARIANTS, AND NORMAL FORMS*

ANDREAS BLASS† AND YURI GUREVICH‡

Abstract. For an equivalence relation E on the words in some finite alphabet, we consider the recognition problem (decide whether two words are equivalent), the invariant problem (calculate a function constant on precisely the equivalence classes), the normal form problem (calculate a particular member of an equivalence class, given an arbitrary member) and the first member problem (calculate the first member of an equivalence class, given an arbitrary member). A solution for any of these problems yields solutions for all earlier ones in the list. We show that, for polynomial time recognizable E, the first member problem is always in the class Δ_2^{P} (solvable in polynomial time with an oracle for an NP set) and can be complete for this class even when the normal form problem is solvable in polynomial time. To distinguish between the other problems in the list, we construct an E whose invariant problem is not solvable in polynomial time with an oracle for E (although the first member problem is in $\mathrm{NP}^E \cap \text{co-NP}^E$), and we construct an E whose normal form problem is not solvable in polynomial time with an oracle for a certain solution of its invariant problem.

Key words. polynomial time, Turing reducible, NP-complete, equivalence relation, certificate, normal form

Introduction. This paper was stimulated by [3], where finite structures (e.g. graphs) were considered as inputs for algorithms. Since one is usually interested only in the isomorphism types of structures, it is natural to ask whether these types can be represented in a form suitable for use as inputs in place of the structures themselves. To avoid trivial "solutions", we insist that the isomorphism type of a structure be (rapidly) computable when the structure itself is given. For this to be possible, it is necessary that the isomorphism problem, for the structures considered, be (rapidly) solvable; indeed, to tell whether two structures are isomorphic, one just computes their isomorphism types and checks whether they are equal. Is this necessary condition sufficient as well? That is, does a solution of the isomorphism problem yield a presentation of the isomorphism classes as well? In many cases it does, because the solution of the isomorphism problem proceeds by calculating an invariant (or a system of invariants) such that two structures of the sort considered are isomorphic exactly when their invariants agree. In such a case, the invariants themselves can be used as a presentation of the isomorphism classes. It is not at all obvious, however, that every solution of an isomorphism problem must yield such invariants. For example, the solution in [2] for trivalent graphs does not appear to yield invariants. We shall show, in the more general setting of arbitrary equivalence relations rather than isomorphism, that one cannot, in general, calculate invariants even if one can tell when two objects are equivalent.

We shall also consider two problems more difficult than the invariant problem. One is obtained by insisting that the invariant be a member of the equivalence class it represents. In the isomorphism-type situation discussed above, this amounts to insisting that one can (rapidly) compute from an isomorphism type some structure in it. An even more difficult problem arises if one insists that the invariant be the first (in some suitable ordering) member of the equivalence class. We shall prove that each of these problems is strictly more difficult than its predecessor.

* Received by the editors December 22, 1982, and in final form July, 1983.

† Department of Mathematics, University of Michigan, Ann Arbor, Michigan 48109.

‡ Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, Michigan 48109.

To make these ideas precise, we consider an equivalence relation E over Σ^*, where Σ is a finite alphabet, and we consider the following four problems:

Recognition problem. Given x and y in Σ^*, determine whether xEy.

Invariant problem. Calculate, for x in Σ^*, an invariant $F(x)$ (in Σ_1^* for some finite alphabet Σ_1) such that, for all x and y in Σ^*, xEy if and only if $F(x) = F(y)$.

Normal form problem. Calculate, for x in Σ^*, a "normal form" $F(x)$ (in Σ^*) such that, for all x and y in Σ^*, $xEF(x)$ and if xEy then $F(x) = F(y)$.

First member problem. Calculate, for x in Σ^*, the first y such that xEy.

In the first member problem, "first" refers to the following standard ordering of Σ^*: y precedes z if either y is shorter than z or they have the same length and y lexicographically precedes z. For many of the equivalence relations we consider, equivalent words have the same length, so "first" can be understood as "lexicographically first".

It is clear that any solution of the first member problem solves the normal form problem and that any solution to the normal form problem solves the invariant problem. Furthermore, given a solution of the invariant problem, we can solve the recognition problem by computing and comparing $F(x)$ and $F(y)$.

In the context of ordinary recursion theory, all four problems are equivalent, for, if we know how to solve the recognition problem, then we can solve the first member problem by testing each y in Σ^* in turn (in standard order) until we find one equivalent to x. Since the search is bounded by x, the problems remain equivalent if we consider only primitive recursive computations. Our objective is to prove that all four problems are inequivalent in the context of polynomial time computability. (They are also inequivalent in higher-type recursion theory, but that does not concern us here.) Unfortunately, this objective cannot be achieved without proving $\mathrm{P} \neq \mathrm{NP}$.

Indeed, the search procedure indicated above implies that, when the recognition problem is in P, the first member problem is in the polynomial hierarchy of Stockmeyer [4] and is therefore in P if $\mathrm{P} = \mathrm{NP}$. More precisely, the first member problem is in Stockmeyer's class Δ_2^{P}, that is, it is computable in polynomial time with an oracle for an NP set, if E is in P. The computation proceeds as follows, using an oracle for the NP set

$$\{(x, y) \mid \text{some member of the equivalence class of } x \text{ precedes } y\}.$$

Given x, we begin by querying the oracle about $(x, 1^l)$, for various values of $l \leqq \text{length}(x)$, to determine by binary search the length m of the first y in the equivalence class of x; this takes approximately $\log(\text{length}(x))$ steps. Then we query the oracle about $(x, 10^{m-1})$ to determine the first digit y_1 of the desired y. Then we query about $(x, y_1 10^{m-2})$ to determine the next digit y_2, etc. In m such steps we determine all of y.

We shall, in § 1, show that the complexity estimate just obtained is optimal, by constructing an E whose first member problem is Δ_2^{P}-complete while the normal form problem has a solution in P (and therefore so do the invariant and recognition problems). This result implies that, if $\mathrm{P} \neq \mathrm{NP}$, the first member problem is strictly harder than the normal form problem.

In § 2, we shall show that no problem in our list is polynomial time Turing reducible to an earlier one. Specifically, we shall show that the invariant problem is strictly harder than the recognition problem by constructing an equivalence relation E whose invariant problem has no solution computable in polynomial time with an oracle for E. We improve this result by showing that we can simultaneously arrange for the first member problem to be in $\mathrm{NP}^E \cap \text{co-NP}^E$. Then we show that the normal form problem is strictly harder than the invariant problem by constructing an equivalence relation

E and a solution F of its invariant problem such that no solution of the normal form problem is computable in polynomial time with an oracle for F. The corresponding result for the normal form and first member problems, namely the existence of an equivalence relation E and a solution F of its normal form problem such that the first member problem is not solvable in polynomial time with an oracle for F, follows immediately from Theorem 1, relativized to an oracle A such that $P^A \neq NP^A$ [1]. Nevertheless, we present a direct and somewhat simpler construction of such E and F using some of the same methods as the other constructions in § 2.

It is natural to ask whether the inequivalence of each pair of consecutive problems in our list can be proved for absolute (without oracles) polynomial time computability. As indicated above, this cannot be done without proving $P \neq NP$. Still, in view of Theorem 1, one could hope to show that these questions are equivalent to natural questions such as $NP \stackrel{?}{=} co\text{-}NP$. Results of this sort will be presented in a subsequent paper (to appear in the proceedings of the Recursive Combinatorics Symposium, Münster, 1983, Springer Lecture Notes).

Finally, in § 3, we formulate some open problems.

1. The first member problem. We show, in this section, that the upper bound of Δ_2^P, for the complexity of the first member problem associated with a polynomial time computable equivalence relation, is optimal, even if we require a polynomial time solution for the normal form problem rather than just the recognition problem.

THEOREM 1. *There is an equivalence relation E on $\{0,1\}^*$ for which the normal form problem is solvable in polynomial time, but the first member problem is Δ_2^P-complete.*

Remark 1. When we say that the problem of computing a certain function F from Σ^* to Σ_1^* is complete for (or hard for, or in) a certain complexity class, we mean that the following decision problem is complete for (or hard for, or in) that class: Given a word x in Σ^*, a letter a in Σ_1 and a positive integer n (in unary notation), decide whether the nth letter in $F(x)$ is a.

Remark 2. In the statement and proof of Theorem 1, completeness refers to log-space reductions. (The proof of the weaker result, where completeness refers to polynomial time reduction, would be no easier, although the words w in the proof could then be assumed to be empty by coding them into m.)

Remark 3. For the equivalence relation E that we shall construct, the last member problem is solvable in polynomial time.

Proof of Theorem 1. We shall work with the alphabet $\{0,1,2\}$; the reduction to $\{0,1\}$ is routine. Let A be a fixed NP-complete set in $\{0,1\}^*$, accepted by a particular nondeterministic Turing machine N in time bounded by a polynomial. Let p be a monotone increasing polynomial (essentially the square of the time bound) such that every computation of N with an input of length l can be coded (in a standard way) by some c in $\{0,1\}^*$ of length $p(l)$. Henceforth, we shall not distinguish between c and the computation it codes.

Consider the following auxiliary problem. An instance of the problem is $m2w2^k$, where m is a word in $\{0,1\}^*$ coding, in some standard way, a query machine M (a deterministic Turing machine with alphabet $\{0,1\}$ that is to interact with an as yet unspecified oracle), w is a word in $\{0,1\}^*$, and k is a positive integer. The question in the instance $m2w2^k$ of the auxiliary problem is whether machine M with oracle A, henceforth written M^A, with input w, halts in fewer than k steps. It is routine to check that this problem is Δ_2^P-complete.

To prove the theorem, we shall, after some preliminary work, define an equivalence relation E on $\{0,1,2\}^*$, give a polynomial time computable solution for its normal

form problem, and give a log-space computable reduction of the auxiliary problem to the first member problem for E. For each instance of the auxiliary problem, we define its *plausible computations* to be the words in $\{0, 1, 2\}^*$ that consist of

(i) $m2w2^k$, followed by
(ii) a string $a_1, \cdots, a_k$ of length k, followed by
(iii) a single digit h, followed by
(iv) a string of length k^2, which we think of as consisting of k blocks $q_1, \cdots, q_k$ each having length k, followed by
(v) a string of length $k \cdot p(k)$, which we think of as consisting of k blocks $c_1, \cdots, c_k$ each having length $p(k)$,

subject to the following requirements:

(α) When the query machine M coded by m is started with input w and allowed to run for $k-1$ steps (or until it halts if this occurs earlier), its first query (if any) to the oracle concerns the unique string q_1' such that $q_1 = q_1'10^r$ for some r. (Note that, by the time bound, length $(q_1') < k =$ length (q_1), so this makes sense.) If the answer to this first query is a_1, under the convention that 0 means "Yes" and 1 means "No", then the next query, if any, concerns the q_2' such that $q_2 = q_2'10^r$ for some r. If the answer to the second query is a_2, the next query (if any) concerns the q_3' such that $q_3 = q_3'10^r$ for some r, etc. (Note that, by the time bound, there will be fewer than k queries altogether.)

(β) If there are exactly j queries in the computation described in (α), then $a_i = 1$ and $q_i = 0^k$ and $c_i = 0^{p(k)}$ for $j < i \leq k$.

(γ) If the computation in (α) reaches a halting state, then $h = 0$; otherwise $h = 1$.

(δ) For each i such that $a_i = 0$, c_i has the form $c_i'10^r$, where c_i' is a computation showing that N accepts q_i'. If $a_i = 1$ then c_i is a string of 0's.

Observe that the property of being a plausible computation is decidable in polynomial time.

For orientation and future reference, we construct, for each $m2w2^k$, a particular plausible computation, called the correct computation. Let the machine M^A with input w run for $k-1$ steps. Let $q_1', \cdots, q_j'$ be its queries to the oracle, and let $a_1, \cdots, a_j$ be the oracle's answers, with 0 representing "Yes". By the time bound, j and each length (q_i') must be smaller than k. For each $i \leqq j$, define q_i to be $q_i'10^r$, where r is chosen (depending on i) so that length $(q_i) = k$. For $j < i \leqq k$, let $a_i = 1$, $q_i = 0^k$, $c_i = 0^{p(k)}$ as required by (β). Note that, with our choices of a's and q's, the computation of M with oracle A is exactly the computation described in (α) above. Let h be 0 if this computation enters a halting state, and 1 otherwise. Finally, define c_i to be $0^{p(k)}$ if $a_i = 1$ and to be $c_i'10^r$ if $a_i = 0$, where c_i' is a computation of N accepting q_i', where r is chosen to make length $(c_i) = p(k)$, and where c_i' is chosen, among all computations accepting q_i', so that c_i is lexicographically as early as possible. It is immediate that $s = m2w2^k a_1 \cdots a_k h q_1 \cdots q_k c_1 \cdots c_k$ is a plausible computation for $m2w2^k$. We call it the *correct computation* for $m2w2^k$.

The important fact about this correct computation is that it lexicographically precedes all other plausible computations for $m2w2^k$. Indeed, suppose $s^* = m2w2^k a_1^* \cdots a_k^* h^* q_1^* \cdots q_k^* c_1^* \cdots c_k^*$ were a lexicographically earlier plausible computation for the same $m2w2^k$. If the first difference between these two plausible computations occurs at a_i, then we would have $a_i^* = 0$ and $a_i = 1$. The computations described in (α) for s and s^* are identical up to, but not including, the oracle's response to the ith query. In particular, that query itself is the same: $q_i = q_i^*$. Since $a_i^* = 0$, condition (δ) for s^* guarantees that c_i^* contains a computation $c_i^{*\prime}$ of N accepting q_i^*. So $q_i \in A$. But this contradicts the fact, expressed by $a_i = 1$, that the A oracle responded

negatively to query q_i. Therefore, the first difference between s and s^* must come later than all the a_i's. The computations described in (α) for s and s^* are therefore the same. Thus $h = h^*$ (by (γ)) and $q_i = q_i^*$ for all i (by (α) and (β)). The first difference between s and s^* must therefore be at some c_i, and, by (δ), we must have $a_i = 0$ for this i. But then we cannot have c_i^* lexicographically preceding c_i, because of the way we chose c_i for the correct computation.

We are, at last, ready to define E. For each $m2w2^k$, we put into one equivalence class all its plausible computations together with the extra word $m2w2^k1^l$, where $l = k(p(k)+k+1)+1$. This value of l was chosen so that the extra word has the same length as the plausible computations for $m2w2^k$. Note that the extra word is always the (lexicographically) last member of its equivalence class. To complete the definition of E, we declare that any word that is neither a plausible computation nor an extra word shall be equivalent only to itself.

The normal form problem for E is solved by the following polynomial time algorithm. Given x, check whether it is a plausible computation for its maximal initial segment ending with a 2. If not, give x as output. If so, give the extra word for that initial segment as output. This algorithm always produces the last member of the equivalence class of x.

The auxiliary problem is reduced to the first member problem for E by the following algorithm. Given an instance $m2w2^k$ of the auxiliary problem, calculate $l = k(p(k)+k+1)+1$, and then write $m2w2^k1^l$. The answer to this instance is given by the digit h in position length$(m2w)+2k+1$ of the first member of the equivalence class of $m2w2^k1^l$. This algorithm works because the equivalence class of the extra word $m2w2^k1^l$ contains all the plausible computations for $m2w2^k$ and its first member is, as we saw above, the correct computation for $m2w2^k$; the h digit in this correct computation encodes the answer to the auxiliary problem for $m2w2^k$. □

2. Nonreducibility. The main result of this section is that, in the context of polynomial time computability, the recognition problem, the invariant problem, the normal form problem and the first member problem are of strictly increasing complexity. None of these problems can be reduced in general to the previous problem on the list.

THEOREM 2. (a) *The invariant problem is not polynomial time Turing reducible to the recognition problem. Indeed, there is an equivalence relation E on $\{0,1\}^*$ such that the invariant problem for E cannot be solved by a polynomial time algorithm with an oracle for E.*

(b) *The normal form problem is not polynomial time Turing reducible to the invariant problem. Indeed, there are an equivalence relation E on $\{0,1\}^*$ and a polynomially bounded solution $F:\{0,1\}^*\to\{0,1\}^*$ of its invariant problem, such that the normal form problem for E cannot be solved by a polynomial time algorithm with an oracle for F.*

(c) *The first member problem is not polynomial time Turing reducible to the normal form problem. Indeed, there are an equivalence relation E on $\{0,1\}^*$ and a polynomially bounded solution F of its normal form problem, such that the first member problem for E cannot be solved by a polynomial time algorithm with an oracle for F.*

Remark 4. To say that a function F is polynomially bounded means that the length of $F(x)$ is bounded by a polynomial in the length of x. We think of computation with an oracle for a function F (rather than a predicate) as follows. The machine can write a word x on its query tape and receive from the oracle, in a single computation step, the value of $F(x)$, printed on a special tape. A query of this sort can be simulated by a sequence of queries concerning the predicate "The nth letter in $F(x)$ is a".

Remark 5. Theorem 1 is relativizable to any oracle. By taking an oracle relative to which $P \neq NP$, as in [1], we immediately obtain part (c) of Theorem 2. Nevertheless we shall give another, more direct proof of (c).

Proof of Theorem 2. Fix an enumeration $M_0, M_1, \cdots$ of all polynomial time bounded query machines with alphabet $\{0, 1\}$, and let p_n be the polynomial clock bound of M_n.

We construct the desired equivalence relation E in stages. Initially E is the equality relation on $\{0, 1\}^*$. At each stage n we may update E by putting into it two pairs (x, y) and (y, x) for some binary strings x, y of the same length d_n. Here $d_0 < d_1 < \cdots$. Thus the resulting equivalence relation has the special properties that each equivalence class has at most two members, the two members of any nontrivial equivalence class are words of the same length, and there is at most one nontrivial equivalence class for each length.

In part (b) we construct the desired invariant function F simultaneously with E. Initially $F(x) = x1$ for every binary word x. At a stage n we may update F by stipulating that $F(x) = 0^{d_n+1}$ for one or two words x of length d_n. In part (c) we construct the desired normal form function F simultaneously with E. Initially F is the identity function $F(x) = x$. At each stage n we may update F by stipulating that $F(x) = 1^{d_n}$ for one word of length d_n.

The sequence $d_0 < d_1 < d_2 < \cdots$ is chosen in such a way that for each n, $p_n(d_n) < d_{n+1}$ and $p_n(d_n) < 2^{d_n - 1}$. Thus, computing on an input of length d_n, M_n asks fewer than 2^{d_n-1} queries and each query is shorter than d_{n+1}.

In the rest of the proof we consider a stage n of the construction. Let x and y range over the binary words of length $d = d_n$. Let $M = M_n$ and $p = p_n$.

(a) For each x let x' be the result produced by M on the input x, when it uses the current E as an oracle. If there are distinct x, y with $x' = y'$ go to the next stage. M^E fails to solve the invariant problem for E because $M^E(x) = x' = y' = M^E(y)$, whereas $(x, y) \notin E$. (Note that later stages of the construction will not affect the values of $M^E(x)$ and $M^E(y)$, because they alter E only on words longer than any query involved in the computation of these values.)

Suppose, on the other hand, that $x' \neq y'$ for all $x \neq y$. We say that x *affects* y if M queries E about (x, y) or (y, x) in the computation of y'. Each y is affected by at most $p(d) < 2^{d-1}$ elements x.

LEMMA 1. *Let R be a binary relation on a nonempty set S of cardinality $2k$. Suppose that for each $u \in S$ there are fewer than k elements $v \in S$ with $(u, v) \in R$. Then there exist distinct $u, v \in S$ such that neither (u, v) nor (v, u) is in R.*

Proof of Lemma 1. Otherwise, the set of all $k(2k-1)$ two-element subsets of S would be the union of the $2k$ sets

$$R_u = \{\{u, v\} : (u, v) \in R\}$$

each of which has cardinality $\leqq k-1$. This is a contradiction. The lemma is proved. □

By the lemma (with $S = \{0, 1\}^d$ and R being the relation "is affected by") there are distinct x, y such that neither of x and y affects the other. Put (x, y) and (y, x) into E. This does not alter the computations of x' and y'. Go to the next stage. M^E fails to solve the invariant problem for E because $(x, y) \in E$, whereas $M^E(x) = x' \neq y' = M^E(y)$.

(b) For each x let F_x by the current F with the following change: $F_x(x) = 0^{d+1}$. Let x' be the result computed by M with input x and the oracle F_x.

If there is an x with $x' \neq x$, then choose one such x, and update F by making it equal to F_x. This does not affect the computation of x'. Go to the next stage. Note

that the updated F is one-to-one on $\{0,1\}^d$ so that there is no need to update E. Since $(x, x') \notin E$, M^F fails to solve the normal problem for E.

Suppose, on the other hand, that $x' = x$ for all x. Choose distinct x and y such that M does not query F_x about y in the computation of x', and M does not query F_y about x in the computation of y'. This choice is possible, by Lemma 1, because each computation involves at most $p(d) < 2^{d-1}$ queries. Update F by stipulating $F(x) = F(y) = 0^{d+1}$, put pairs (x, y) and (y, x) into E, and go to the next stage. M^F fails to solve the normal form problem for E because $M^F(x) = x \neq y = M^F(y)$, whereas $(x, y) \in E$.

(c) For each x let F_x be the current F with the following change: $F_x(x) = 1^d$. Let x' be the result computed by M with oracle F_x on input x. If there is an x with $x' \neq x$, then choose one such x, update F by making it equal to F_x, put $(x, 1^d)$ and $(1^d, x)$ into E, and go to the next stage. M^F fails to solve the first member problem for E because $M^F(x) = x' \neq x$ whereas x is the first member in its equivalence class.

Suppose, on the other hand, that $x' = x$ for every x. In particular $(1^d)' = 1^d$. Choose $x < 1^d$ such that M does not query the oracle about x in the computation of $(1^d)'$. This choice is possible because the computation involves at most $p(d) < 2^{d-1}$ queries. Update F by making it equal to F_x. This does not change the computations of x' and $(1^d)'$. Put $(x, 1^d)$ and $(1^d, x)$ into E, and go to the next stage. M^F fails to solve the first member problem for E because $M^F(1^d) = 1^d$, whereas 1^d is not the first member in its equivalence class. □

Our last result is an improvement of part (a) in Theorem 2. It asserts that the invariant problem can be intractable even when the recognition problem is tractable and the first member problem is nearly tractable.

THEOREM 3. *There is an equivalence relation E on $\{0,1\}^*$ such that*

(i) *the invariant problem for E is not solvable by a polynomial time algorithm with an oracle for E; and*

(ii) *the first member problem for E is in* NP *and* co-NP *relative to E.*

Proof. In the proof of part (a) of Theorem 2 we constructed an equivalence relation E such that every equivalence class has at most two elements, the two members of any nontrivial equivalence class are words of the same length, and there is at most one nontrivial equivalence class for each length. We shall modify the construction so that for each length $l > 0$ there is exactly one nontrivial equivalence class.

First make sure that $4p_n(d_n) < 2^{d_n} - 3$. Then turn to a stage n of the construction. Here we first update E by adding to it all pairs $(0^l, 1^l)$ and $(1^l, 0^l)$ such that $0 < l < d_0$ if $n = 0$, and $d_{n-1} < l < d_n$ otherwise. We use the notation of the proof of Theorem 2. Let x' be the result produced by M on input $x \in \{0,1\}^d$ with an oracle for the current E. If there is a pair $x \neq y$ with $x' = y'$ pick one such pair x, y. Computing x' and y', M queried E about $\leq 2p(d)$ pairs (u, v) altogether. Hence at most $4p(d) < 2^d - 3$ words were involved in all those queries. Choose $u, v \in \{0,1\}^d$ such that u, v, x, y are different and u, v were not involved in the queries of the computations of x', y'. Update E by adding (u, v) and (v, u) to E. This does not alter the computations of x', y'. Go to the next stage. If, on the other hand, $x' \neq y'$ for all $x \neq y$ proceed as in the proof of part (a) of Theorem 2.

To show that the function sending each x to the first member of its equivalence class is in $\mathrm{NP}^E \cap \text{co-NP}^E$, we first observe that its graph, i.e. the binary relation "y is the first member of the E-class of x," is NP^E by the following procedure. If $y < x$ and $(x, y) \in E$, then accept; if $y = x$, then guess two distinct but E-related words of the same length as x, and accept if x is not the later of these two words. To finish the proof, we make the general remark that any polynomially bounded function F whose

graph is in NP^E is itself in $NP^E \cap$ co-NP^E. The NP^E procedure for determining that the nth letter in $F(x)$ is (resp. is not) a is to guess the value y of $F(x)$ and a computation showing that (x, y) belongs to the graph of F and then to accept if the nth letter of y is (resp. is not) a.

3. Problems. The results in this paper suggest several natural problems. Two that strike us as particularly interesting are the following.

1. Can results similar to ours be obtained if, instead of considering arbitrary equivalence relations, one restricts attention to the relation of isomorphism between (standard representations of) structures?

2. Are there analogues of Theorem 1 for the invariant and normal form problems? That is, is there a polynomial time computable E such that every polynomially bounded solution of its invariant problem is Δ_2^P-hard? And is there an E such that the invariant problem is solvable in polynomial time but every polynomially bounded solution of the normal form problem is Δ_2^P-hard?

REFERENCES

[1] T. BAKER, J. GILL AND R. SOLOVAY, *Relativizations of the* P = ?NP *question*, this Journal, 4 (1975), pp. 431–442.

[2] Z. GALIL, C. M. HOFFMANN, E. M. LUKS, C. P. SCHNORR AND A. WEBER, *An* $O(n^3 \log n)$ *deterministic and an* $O(n^3)$ *probabilistic isomorphism test for trivalent graphs*, Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science, 1982, pp. 118–125.

[3] Y. GUREVICH, *Toward logic tailored for computational complexity*, Proc. European Logic Colloquium, Aachen, 1983, Springer Lecture Notes, to appear.

[4] L. J. STOCKMEYER, *The polynomial-time hierarchy*, Theoret. Comp. Sci., 3 (1977), pp. 1–22.