ON γ-REDUCIBILITY VERSUS POLYNOMIAL TIME MANY-ONE REDUCIBILITY*
(Extended Abstract)

Timothy J. Long

Department of Computer Science
New Mexico State University
Las Cruces, New Mexico 88003

ABSTRACT

We prove that a class of functions (denoted by NPC_t^P), whose graphs can be accepted in nondeterministic polynomial time, can be evaluated in deterministic polynomial time if and only if γ-reducibility is equivalent to polynomial time many-one reducibility. We also modify the proof technique used to obtain part of this result to obtain the stronger result that if every γ-reduction can be replaced by a polynomial time Turing reduction then every function in NPC_t^P can be evaluated in deterministic polynomial time.

INTRODUCTION

In this paper we prove the equivalence of two open questions in computational complexity theory. The first question was raised by Adleman and Manders [1] and asks whether a particular nondeterministic version of polynomial time many-one reducibility, which they call γ-reducibility, is equivalent to polynomial time many-one reducibility. The second question was raised by Valiant [6] and asks whether a class of functions, which he calls NPC_t^P, whose graphs can be accepted in nondeterministic polynomial time is contained in a class of functions, which he calls PE^P, which can be evaluated in deterministic polynomial time; that is, whether $NPC_t^P \subseteq PE^P$.

The motivation for examining this equivalence comes from the rather obvious observation that any γ-reduction is realized by an element of NPC_t^P while any polynomial time many-one reduction is realized by an element of PE^P. In fact, the result that $NPC_t^P \subseteq PE^P$ implies that γ-reducibility is equivalent to polynomial time many-one reducibility follows immediately from this observation. It is the other half of the equivalence which we feel is interesting and in Theorem 2 we prove that if γ-reducibility is equivalent to polynomial time many-one reducibility then $NPC_t^P \subseteq PE^P$.

This last result may be thought of as saying that if any γ-reduction can be replaced by a polynomial time many-one reduction then $NPC_t^P \subseteq PE^P$. One way to strengthen this statement is to allow the γ-reductions to be replaced by more general types of polynomial time reduction procedures. This is exactly the type of strengthening which we achieve in Theorem 4 where we prove that if every γ-reduction can be replaced by a polynomial time Turing reduction then $NPC_t^P \subseteq PE^P$. Since polynomial time Turing reducibility seems to be

*This work represents a portion of the author's doctoral dissertation [5] completed at Purdue University and was partially supported by NSF Grant No. MCS-76-09212.

the most general form of polynomial time reducibility (Ladner, Lynch, and Selman [4]), Theorem 4 seems to be the strongest obtainable form where the γ-reductions are to be replaced by polynomial time reduction procedures. Obviously Theorem 2 is an easy corollary to Theorem 4. However because the proof of Theorem 4 uses a modification of the ideas used to prove Theorem 2, we include an independent proof of Theorem 2 in order to motivate the more involved construction used to prove Theorem 4.

BASICS

The three types of reducibilities used in this paper are polynomial time many-one reducibility which was introduced by Karp [3], polynomial time Turing reducibility which was introduced by Cook [2], and γ-reducibility which was introduced by Adleman and Manders [1]. A set A is polynomial time many-one reducible to a set B ($A \leq_m^P B$) if there is a function g which is computable in polynomial time such that for all x, $x \varepsilon A$ if and only if $g(x) \varepsilon B$. A set A is polynomial time Turing reducible to a set B ($A \leq_T^P B$) if there is an oracle Turing machine M which runs in polynomial time such that M, with oracle set B, recognizes A.

For any nondeterministic transducer M which runs in polynomial time, let

$G(M) = \{<x,y> \mid$ some computation sequence of M on input x halts with y on its output tape$\}$.

A set A is γ-reducible to a set B ($A \leq \gamma B$) if there is a nondeterministic transducer M which runs in polynomial time such that:

i: $\forall x \exists y \ (<x,y> \varepsilon G(M))$

ii: $\forall x \forall y \ (<x,y> \varepsilon G(M) \Rightarrow (x \varepsilon A \Leftrightarrow y \varepsilon B))$.

Part i of this definition requires M to produce at least one output value for every input value. Part ii requires all of the output values produced by M to be in B if the input value is in A and all of the output values to be in $\overline{B}$ otherwise. Thus, one can view γ-reducibility as a nondeterministic version of $\leq_m^P$-reducibility.

It is clear from the definitions that for any sets A and B, if $A \leq_m^P B$ then $A \leq \gamma B$. The converse of this statement is one of the open questions with which we are concerned here; that is, does $A \leq \gamma B$ imply that $A \leq_m^P B$ for all sets A and B.

We now discuss, following Valiant [6], the second open question with which we are concerned. An evaluator for a function f (which is possibly multi-valued and not necessarily total) is a transducer which takes as input a string x and outputs on its output tape a value of f(x) if f(x) is defined and outputs the special symbol Ω otherwise. PE is the class of functions which can be evaluated by transducers whose running time is bounded by a polynomial function of the sum of the lengths of the input and output. The restriction of the class PE to functions whose output length is polynomially bounded by the input length is denoted by PE^P. It is easy to verify that PE^P is precisely the class of functions that can be computed by transducers which run in polynomial time.

A checker for a function f is a Turing machine (possibly nondeterministic) which takes as input the string <x,y> and accepts <x,y> if and only if $y \varepsilon f(x)$; that is, if and only if y is a value of f(x). NPC denotes the class of functions which can be checked by nondeterministic

Turing machines whose running time is bounded by a polynomial function of the length of the input <x,y>. The superscript p is again used to denote the restriction of this class to functions whose output length is polynomially bounded by the input length and the subscript t is also introduced to denote the additional restriction of NPC to total functions. Thus, the class NPC_t^P is the class NPC with the two extra restrictions denoted by the subscript t and the superscript p. We take the notation $NPC_t^P \subseteq PE^P$ to mean that if $f \varepsilon NPC_t^P$ then there is a function $g \varepsilon PE^P$ such that $g(x) \varepsilon f(x)$ for all x. In this case we say that g is a restriction of f which is computable in deterministic polynomial time. The second open question with which we are concerned is whether $NPC_t^P \subseteq PE^P$.

MAIN THEOREMS:

PROPOSITION 1: If $NPC_t^P \subseteq PE^P$ then for all sets A and B, $A \leq \gamma B$ implies that $A \leq_m^P B$.

Proof: The proof is actually a trivial consequence of the observation that if $A \leq \gamma B$ via transducer M, then the function which M computes, say f, is an element of NPC_t^P. Then if $NPC_t^P \subseteq PE^P$, a restriction of f computable in deterministic polynomial time witnesses $A \leq_m^P B$.

Theorem 2 establishes the converse of Proposition 1 and the equivalence of these two questions.

THEOREM 2: If $A \leq \gamma B$ implies that $A \leq_m^P B$ for all sets A and B, then $NPC_t^P \subseteq PE^P$.

Sketch of Proof: Let $f \varepsilon NPC_t^P$ and let nondeterministic transducer M compute f in polynomial time. The proof is done by constructing recursive sets A and B such that

(I) $A \leq \gamma B$

(II) If $A \leq_m^P B$ then there is a $g \varepsilon PE^P$ such that $g(x) \varepsilon f(x)$ for all x.

Thus, under the assumption that $A \leq \gamma B$ imples that $A \leq_m^P B$, we can conclude the $f \varepsilon PE^P$.

Recalling the definition of G(M), let

$$G(M,x) = \{u \mid \pi_1(u) = x \text{ and } u \varepsilon G(M)\}.*$$

In order to meet condition (I), whenever the construction assigns a string x to A it assigns all of G(M,x) to B and whenever it assigns x to $\overline{A}$ it assigns all of G(M,x) to $\overline{B}$. Based on this assignment of strings to A, $\overline{A}$, B and $\overline{B}$, the following is an informal description of a nondeterministic transducer M′ which witnesses $A \leq \gamma B$:

> On input x, begin simulating M on input x. On any simulated path of M which produces output, say y, output <x,y>.

The construction meets condition (II) by building A and B so that if $A \leq_m^P B$ then there must be a function $g \varepsilon PE^P$ which not only witnesses this reduction but also has the property that $g(x) \varepsilon G(M,x)$ for all but finitely many x. Notice that such a function g can be used to compute a restriction of f (with at most finitely many exceptions) in deterministic polynomial time by first computing g(x) and then computing $\pi_2(g(x))$. The complete proof of Theorem 2 is given in the appendix.

COROLLARY 3: $A \leq \gamma B$ implies that $A \leq_m^P B$ for all sets A and B if and only if $NPC_t^P \subseteq PE^P$.

---

*Projection functions are denoted by π_i.

Beginning with an $f \in NPC_t^P$, the construction of Theorem 2 produces sets A and B such that $A \le \gamma B$ and if $A \le_m^P B$ then there is a $g \in PE^P$ such that $A \le_m^P B$ via g and $g(x) \in G(M,x)$ for all but finitely many x. In other words, A and B are constructed so that if $A \le_m^P B$ then there is a special $\le_m^P$-reduction of A to B which, on input x (with at most finitely many exceptions), queries a string from which an element of f(x) can be obtained in deterministic polynomial time. In Theorem 4 we use this same idea again to prove the stronger result that if $A \le \gamma B$ implies that $A \le_T^P B$ for any sets A and B, then $NPC_t^P \subseteq PE^P$. This time, beginning with $f \in NPC_t^P$, A and B are constructed so that $A \le \gamma B$ and if $A \le_T^P B$ then there is a particular $\le_T^P$-reduction of A to B which, on input x (for all but finitely many x), eventually queries a string from which an element of f(x) can be obtained in deterministic polynomial time. Thus, if oracle Turing machine M´ witnesses this special $\le_T^P$-reduction of A to B, we can then use M´ to compute a restriction of f (with at most finitely many exceptions) in deterministic polynomial time by simulating M´ with oracle set B on input x until M´ queries a string from which an element of f(x) can be obtained in deterministic polynomial time. When this happens, the simulation of M´ stops and a value of f(x) is produced.

There are two particular difficulties to deal with in carrying through this construction using $\le_T^P$-reductions which did not have to be dealt with in Theorem 2 where we used $\le_m^P$-reductions. For all but finitely many x, the special $\le_m^P$- reduction of A to B in Theorem 2 queries, on input x, only one element which we know to be in G(M,x). This means that deciding membership in G(M,x) uniformly in x, is not necessary in order to obtain an element of f(x). However, the special $\le_T^P$-reduction of A to B which M´ witnesses may, on input x, query strings not in G(M,x). This implies the need to decide membership in G(M,x) uniformly in x. Since we do not know if G(M) is in P for arbitrary nondeterministic transducers M which run in polynomial time, we do not know if membership in G(M,x) can be decided in deterministic polynomial time uniformly in x. Therefore, we do not know if a simulation of M´ on input x can decide in deterministic polynomial time, uniformly in x, when M´ is querying a string in G(M,x) in order to then obtain an element of f(x) from such a string.

To overcome this difficulty we introduce new sets. For any nondeterministic transducer M which runs in polynomial time, let

$T(M) = \{\langle x,y,z\rangle \mid$ y is the sequence of instantaneous descriptions on a computation path of M which halts with z on its ouput tape when started with input x$\}$

and let $T(M,x) = \{u \mid u \in T(M) \text{ and } \pi_1(u) = x\}$. It is easy to verify that T(M) is in P and that if $u \in T(M,x)$ then $\pi_3(u) \in f(x)$ when M computes f. Thus, membership in T(M,x) can be decided in deterministic polynomial time uniformly in x and, given $u \in T(M,x)$, an element of f(x) can be produced in deterministic polynomial time when M computes f. The construction of Theorem 4 builds A and B so that $A \le \gamma B$ and, if $A \le_T^P B$, then there is a special $\le_T^P$-reduction of A to B which, on input x for all but finitely many x, queries an element of T(M,x) at some point in its computation.

The second difficulty to overcome is that when simulating M' on input x with oracle set B (where M' witnesses the special $\leq_T^P$-reduction of A to B), any queries which M' generates before querying an element of T(M,x) must be answered correctly with B being the oracle set so that the simulation of M' proceeeds as if M' were witnessing $A \leq_T^P B$. Thus, if the simulation of M' is to be used to compute a restriction of f in deterministic polynomial time, B must be constructed so that the simulation of M' can answer queries about B correctly in deterministic polynomial time. This problem is solved by the way in which strings are assigned to B or $\overline{B}$. Specifically, if $A \leq_T^P B$ with M' witnessing the special $\leq_T^P$-reduction of A to B, then it will be the case (for all but finitely many x) that $T(M,x) \subseteq B$ if the first string queried by M' on input x is an element of T(M,x) and $T(M,x) \subseteq \overline{B}$ otherwise. Also, all of $\overline{T(M)}$ is assigned to $\overline{B}$. This implies that the simulation of M' on input x can process all queries about all strings u (with at most finitely many exceptions) correctly with B being the oracle set using the following rules:

1. If $u \varepsilon T(M,x)$ then do not answer the query. Instead, output $\pi_3(u) \varepsilon f(x)$ and halt.
2. If $u \not\varepsilon T(M)$ then answer NO.
3. If $u \varepsilon T(M,y)$ where $y \neq x$ then answer YES if the first string queried by M' on input y is in T(M,y) and answer NO otherwise.

Note that all of these conditions can be checked in deterministic polynomial time.

We now present the strengthened version of Theorem 2.

<u>THEOREM 4</u>: If $A \leq \gamma B$ implies that $A \leq_T^P B$ for all sets A and B, then $NPC_t^P \subseteq PE^P$.

<u>Sketch of Proof</u>: Let $f \varepsilon NPC_t^P$ and let nondeterministic transducer M compute f in polynomial time. The proof is done by constructing recursive sets A and B so that

(I) $A \leq \gamma B$

(II) If $A \leq_T^P B$ then there is a $g \varepsilon PE^P$ such that $g(x) \varepsilon f(x)$ for all x.

Thus, under the assumption that $A \leq \gamma B$ implies that $A \leq_T^P B$, we can conclude that $f \varepsilon PE^P$.

In order to meet condition (I), whenever the construction assigns a string x to A it assigns all of T(M,x) to B and whenever it assigns a string x to $\overline{A}$ it assigns all of T(M,x) to $\overline{B}$. Based on this assignment of strings to A, $\overline{A}$, B, and $\overline{B}$, the following is an informal description of a nondeterministic transducer M' which witnesses $A \leq \gamma B$:

> On input x, M' simulates M on input x. On any simulated path of M which produces output, say z, M' outputs $\langle x,y,z \rangle$ where y is the sequence of instantaneous descriptions on the simulated path producing z.

The construction meets condition (II) by building A and B so that if $A \leq_T^P B$ then there is an oracle Turing machine M'' which not only witnesses $A \leq_T^P B$ but also has the property that it queries an element of T(M,x) at some point in its computation on input x (for all but finitely many x) with oracle set B. We then use M'' to compute a restriction of f (with at most finitely many exceptions) in deterministic polynomial time by simulating M'' on input x with oracle set B until an element of T(M,x) is queried. When this element, say u, is queried, $\pi_3(u)$ yields an element of f(x) in deterministic polynomial time. The complete proof of Theorem 4 is given in the appendix.

CONCLUSION

We have shown that $NPC_t^P \subseteq PE^P$ if and only if $A \leq_\gamma B$ implies that $A \leq_m^P B$ for all sets A and B. By modifying the technique used to prove half of this equivalence we obtained the result that if $A \leq_\gamma B$ implies that $A \leq_T^P B$ for all sets A and B then $NPC_t^P \subseteq PE^P$.

Valiant [6] related the $NPC_t^P \subseteq PE^P$? question to the P = NP? proving that:

(I) If P = NP then $NPC_t^P \subseteq PE^P$.

(II) If $NPC_t^P \subseteq PE^P$ then P = NP $\cap$ co-NP.

It is easy to prove directly that these same relations hold between the P = NP? question and the question of γ-reducibility being equivalent to $\leq_m^P$-reducibility. Alternatively, these relations follow from Corollary 3 and Valiant's results. An interesting open question remaining from Valiant's work is whether the converse of either (I) or (II) holds; in fact, this question was reformulated by Adleman and Manders [1] who asked if P $\neq$ NP implies that γ-reducibility is not equivalent to $\leq_m^P$-reducibility.

ACKNOWLEDGEMENT

I would like to thank my thesis advisor, Paul Young, for his encouragement and guidance.

REFERENCES

[1] Adleman, L. and K. Manders, "Reducibility, Randomness, and Intractibility," Proc. 9th ACM STOC, 1977, pp. 151-163.

[2] Cook, S., "The Complexity of Theorem Proving Procedures," Proc. 3rd ACM STOC, 1971, pp. 151-158.

[3] Karp, R., "Reducibility Among Combinatorial Problems," Complexity of Computer Computations," Miller and Thatcher, eds., Plenum Press, NY, 1972, pp. 85-103.

[4] Ladner, R., N. Lynch, A. Selman, "A Comparison of Polynomial Time Reducibilities," Theoretical Computer Science, vol. 1(1975), pp. 103-123.

[5] Long, T., "On Some Polynomial Time Reducibilities," Ph.D. Dissertation, Purdue University, 1978.

[6] Valiant, L., "Relative Complexity of Checking and Evaluating," Information Processing Letters, vol. 5, no. 2(1976), pp. 20-23.

APPENDIX

THEOREM 2: If $A \leq_\gamma B$ implies that $A \leq_m^P B$ for all sets A and B, then $NPC_t^P \subseteq PE^P$.

Proof: The construction assigns strings to A, $\overline{A}$, B, and $\overline{B}$ in stages with string x being assigned to A or $\overline{A}$ at or before stage x and string $y \in G(M,x)$ being assigned to B or $\overline{B}$ at or before stage x. To start the construction assign λ to A, let $\overline{A} = \emptyset$, assign $G(M,\lambda)$ to B, let $\overline{B} = \overline{G(M)}$, and GOTO stage 0.

STAGE X :

1. If x was assigned to A or $\overline{A}$ during an earlier stage, then GOTO stage $x \oplus 1$.
2. Otherwise, find the smallest index not cancelled at an earlier stage, say j. (We let $g_0, g_1, g_2, \ldots$ be an effective enumeration of the class of functions which are computable in polynomial time.)
3. Compute $g_j(x)$.

   CASE 1: $g_j(x) \notin G(M,x)$

   There are three mutually exclusive possibilities under case I.

   I. 1: $g_j(x)$ was assigned to $\overline{B}$ during an earlier stage. In this case do the following:

   Assign x to A; Assign G(M,x) to B;

Cancel index j; GOTO stage $x\oplus 1$.

I.2: $g_j(x)$ was assigned to B during an earlier stage.

In this case do the following:

Assign x to $\overline{A}$; Assign G(M,x) to $\overline{B}$;

Cancel index j; GOTO stage $x\oplus 1$.

I.3: $g_j(x)$ has not yet been assigned to B or B.

In this case since $\overline{G(M)} \subseteq \overline{B}$, $g_j(x) = \langle u,v\rangle \epsilon G(M)$ and since the construction is in case I, $u \neq x$. In this case do the following:

Assign x to A: Assign G(M,x) to B;

Assign u to $\overline{A}$; Assign G(M,u) to $\overline{B}$;

(Notice that his assigns $g_j(x)$ to $\overline{B}$)

Cancel index j; GOTO stage $x\oplus 1$.

CASE II: $g_j(x) \epsilon G(M,x)$.

In this case do the following:

Assign x to A; Assign G(M,x) to B;

GOTO stage $X\oplus 1$.

END OF STAGE X

From the construction it is clear that for all x, $x\epsilon A$ if and only if $G(M,x) \subseteq B$ and $x\epsilon\overline{A}$ if and only if $G(M,x) \subseteq \overline{B}$. Thus, $A \leq \gamma B$ via the nondeterministic transducer M′ which was informally described earlier in the sketch of the proof of Theorem 2.

We will now argue that if $A \leq_m^P B$ then some restriction of f can be computed in deterministic polynomial time. Notice that when the construction cancels an index, say j, at some stage, say x, this cancellation occurs either at I.1, I.2, or I.3 inside of case I. In each of these places x is assigned to A or $\overline{A}$ in such a way that $x\epsilon A$ if and only if $g_j(x) \epsilon B$ is not true. Thus, when j is cancelled x witnesses $A \not\leq_m^P B$ via g_j. It follows that if the construction cancels every index then $A \not\leq_m^P B$. Conversely, if $A \leq_m^P B$ then some index is not cancelled by the construction.

Now assume that $A \leq_m^P B$ and let j be the smallest index not cancelled by the construction. Let j_0 be the first stage where the construction enters step 2 and discovers that j is the smallest uncancelled index. Let F be the set of elements not assigned to A or $\overline{A}$ during all stages prior to stage j_0.

CLAIM: If $x\epsilon F$ then x and only x is assigned to A or $\overline{A}$ during stage x.

Proof: The proof is by induction on the elements of F. The smallest element in F is in fact j_0 and by the choice of j_0, stage j_0 enters step 3. All possible cases inside of step 3 assign j_0 to A or $\overline{A}$ so that j_0 is assigned to A or $\overline{A}$ during stage j_0.

If stage j_0 assigns some string $y \neq j_0$ to A or $\overline{A}$, this assignment would have to be made at case I.3. But at caseI.3, stage j_0 would cancel j. By assumption j is never cancelled, so only j_0 is assigned to A or $\overline{A}$ during stage j_0.

Now assume that the claim is true for the first k elements of F, say $x_1, x_2, \ldots, x_k$ and consider the k+1 element in F, x_{k+1}. By the induction assumption and the definition of F, stage x_{k+1} has to enter step 3. The proof for x_{k+1} now proceeds just as the proof for j_0. QED

It follows from the claim that for all $x\epsilon F$, stage x enters step 3. Because j is never cancelled each of these stages must go to case II. Thus, for all $x\epsilon F$, $g_j(x) \epsilon G(M,x)$. Therefore, g_j can be used to compute a restriction of a in deterministic polynomial time in the following way:

On input x, if $x \notin F$ then output an element of $f(x)$ using a finite table.

If $x \varepsilon F$, compute $\pi_2(g_j(x))$ to obtain an element of $f(x)$.

Hence, if $A \leq_m^P B$ then there is a $g \varepsilon PE^P$ such that for all x, $g(x) \varepsilon f(x)$.

THEOREM 4: If $A \leq \gamma B$ implies that $A \leq_T^P B$ for all sets A and B, then $NPC_t^P \subseteq PE^P$.

Proof: The construction assigns strings to A, $\overline{A}$, B, and $\overline{B}$ in stages with string x being assigned to A or $\overline{A}$ at or before stage x and string $y \varepsilon T(M,x)$ being assigned to B or $\overline{B}$ at or before stage x. To start the construction assign λ to A, let $\overline{A} = \emptyset$, assign all of $T(M,\lambda)$ to B, let $\overline{B} = \overline{T(M)}$, and GOTO stage 0.

STAGE X:

(1) If x was assigned to A or $\overline{A}$ at an earlier stage then GOTO stage $x \oplus 1$.

(2) Otherwise, find the smallest index not cancelled at an earlier stage, say j. (We let $M_0, M_1, M_2, \ldots$ be an effective enumeration of the oracle Turing machines which run in polynomial time.)

(3) Begin simulating M_j on input x.

(*) If the first string queried by M_j is an element of $T(M,x)$ then:

Assign x to A and all of $T(M,x)$ to B; GOTO stage $x \oplus 1$.

Otherwise, proceed with the simulation of M_j answering all queries about any strings u according to the following rules:

(a) if $u \notin T(M)$ then answer NO. (Recall that $B \subseteq T(M)$.)

(b) If $u \varepsilon T(M)$ and u was assigned to B or $\overline{B}$ at an earlier stage, answer the query YES if $u \varepsilon B$ and answer NO if $u \varepsilon \overline{B}$.

(c) If $u \varepsilon T(M,x)$ then:

Assign x to $\overline{A}$; Assign all of $T(M,x)$ to $\overline{B}$; Release all temporary assignments made during stage x; GOTO stage $x \oplus 1$.

COMMENT: In this case u is not the first string queried by M_j on input x or else (*) would have applied and the construction would have left stage x. Also, because (1) did not apply, (b) did not apply and x and $T(M,x)$ had not yet been assigned, respectively, to A and B or $\overline{A}$ and $\overline{B}$.

(d) If $u = \langle y,z,w \rangle \varepsilon T(M,y)$ and (b) and (c) do not apply (that is, $y \neq x$), then do the following:

Begin simulating M_j on input y. If the first string queried during this new simulation is an element of $T(M,y)$ then temporarily:

Assign y to A; Assign all of $T(M,y)$ including u, to B.

If the first string queried is not an element of $T(M,y)$ then temporarily:

Assign y to $\overline{A}$; Assign all of $T(M,y)$, including u, to $\overline{B}$.

Answer YES to the query about u generated by the simulation of M_j on input x if u was just temporarily assigned to B and answer NO if u was just temporarily assigned to $\overline{B}$.

COMMENT: In this case, because $u \varepsilon T(M,y)$ and (b) did not apply, y and $T(M,y)$ had not yet been assigned, respectively, to A and B or $\overline{A}$ and $\overline{B}$.

(4) If the outer simulation which began in step (3) completes the computation of M_j on input x without (*) or (c) applying, then no member of $T(M,x)$ was queried during the

simulation. In this case do the following:

(i) Make all temporary assignments in (d) permanent assignments.

COMMENT: This guarantees that the simulation of M_j on input x answered all queries consistently with the oracle set being B.

(ii) If the simulated path of M_j on input x accepted, then assign x to $\overline{A}$ and all of T(M,x) to $\overline{B}$. If the simulated path rejected, then assign x to A and all of T(M,x) to B.

COMMENT: In this case, because (1), (*), and (c) never applied, x and T(M,x) had not yet been assigned, respectively, to A and B or $\overline{A}$ and $\overline{B}$. Also, we have just made x a witness to $A \not\leq_T^P B$ via M_j.

(iii) CANCEL j and GOTO stage $x \oplus 1$.

<u>END OF STAGE X</u>

It is clear from the construction that for any string x, $x \in A$ if and only if $T(M,x) \subseteq B$ and and $x \in \overline{A}$ if and only if $T(M,x) \subseteq \overline{B}$. Therefore, $A \leq_\gamma B$ via the transducer M′ described earlier in the sketch of the proof of Theorem 4.

By the comments in the construction, if index j is cancelled then $A \not\leq_T^P B$ via M_j. Thus, if $A \leq_T^P B$, some index is not cancelled by the construction. Assuming that $A \leq_T^P B$, let j be the smallest index not cancelled in the construction. We now show that M_j can be used to compute a restriction of f in deterministic polynomial time.

Let j_0 be the first stage where the construction enters step 2 and discovers that j is the smallest uncancelled index. Let F be the set of elements not assigned to A or $\overline{A}$ during stages prior to stage j_0.

PROPERTY 1: If $x \in F$ then x and only x is assigned to A or $\overline{A}$ during stage x.

Proof: Property 1 is proved by induction on the elements of F. j_0 is the smallest element in F and by the definition of j_0, stage j_0 enters step (3). The construction leaves stage j_0 at (*), (c), or in step (4). In each of these places j_0 is assigned to A or $\overline{A}$ during stage j_0.

Additionally, only in (d) could a string other than j_0 be assigned to A or $\overline{A}$. But the assignments in (d) are only temporary and if they become permanent during stage j_0, then stage j_0 entered step (4) and cancelled j. Since, by assumption, j is never cancelled, the assignments in (d) are not made permanent during stage j_0. Thus, only j_0 is assigned to A or $\overline{A}$ during stage j_0.

Assume that property 1 holds for the first k elements of F, $x_1, x_2, x_3, \ldots, x_k$, and consider the element x_{k+1}. By the induction assumption and the definition of F, stage x_{k+1} enters step (3). The argument that x_{k+1} and only x_{k+1} is assigned to A or $\overline{A}$ during stage x_{k+1} is now the same as the argument for j_0. QED

PROPERTY 2: If $x \in F$ then $x \in A$ if and only if x is assigned at (*) during stage x and $x \in \overline{A}$ if and only if x is assigned at (c) during stage x.

Proof: Let $x \in F$. By Property 1 x is assigned to A or $\overline{A}$ during stage x. x can only be assigned at (*), (c), or in (4). If x is assigned in (4) then j is cancelled. By assumption, j is never cancelled so x is assigned at (*) or (c). The proof is now immediate from the way in which strings are assigned to A or $\overline{A}$ at (*) and (c). QED

Now consider the following description of a transducer, say M_j'.

On input x, if $x \notin F$, then output an element of f(x) using a finite table. If $x \in F$, then begin simulating M_j on input x. If M_j queries a string $u \in T(M,x)$, then compute $\pi_3(u)$ to obtain an element of f(x). If M_j queries a string $u \notin T(M,x)$ then answer the query according to the following rules and proceed with the simulation.

(i) If $u \notin T(M)$ then answer NO.

(ii) If $u = \langle y,z,w \rangle \in T(M)$ where $y \neq x$ and $y \in F$, then begin simulating M_j on input y. If the first string queried is an element of T(M,y) then answer YES, otherwise answer NO. If $y \notin F$ then answer the query about u YES if y was assigned to A and answer NO if y was assigned to $\overline{A}$.

By Property 2, the answers in (i) and (ii) are consistent with B. Also, these answers can be decided in deterministic polynomial time. To have M_j' computing a restriction of f in deterministic polynomial time, it only remains to show that when $x \in F$ then some element of T(M,x) is found during the computation of M_j' on input x. But, by Properties 1 and 2, M_j' on input $x \in F$ decides answers in (i) and (ii) just as the construction does during stage x. Thus, M_j' follows the same computation path of M_j as does stage x and stage x has to find an element of T(M,x) or else j is cancelled in step (4). Therefore, M_j' can be used to compute a restriction of f in deterministic polynomial time. QED