# Comparison of Polynomial- Time Reducibilities

Richard Ladner
University of Washington
Seattle, Wash.

Nancy Lynch
University of Southern California
Los Angeles, Cal.

Alan Selman
Florida State University
Tallahassee, Fla.

Abstract

Comparison of the polynomial-time-bounded reducibilities introduced by Cook [1] and Karp [4] leads naturally to the definition of several intermediate truth-table reducibilities. We give definitions and comparisons for these reducibilities ; we note, in particular, that all reducibilities of this type which do not have obvious implication relationships are in fact distinct in a strong sense. Proofs are by simultaneous diagonalization and encoding constructions.

Work of Meyer and Stockmeyer [7] and Gill [2] then leads us to define nondeterministic versions of all of our reducibilities. Although many of the definitions degenerate, comparison of the remaining nondeterministic reducibilities among themselves and with the corresponding deterministic reducibilities yields some interesting relationships.

## I. Introduction

Computation-resource-bounded reducibilities play a role in the theory of computational complexity which is analogous to, and perhaps as important as, the various kinds of effective reducibilities used in recursive function theory. Just as the effective reducibilities are used to classify problems according to their degrees of unsolvability, [9] space- and time- bounded reducibilities may be used to classify problems according to their complexity level.

The most fruitful resource-bounded reducibilities thus far have been the polynomial-time-bounded reducibilities of Cook [1] and Karp [4], corresponding respectively to Turing and many-one reducibilities in recursive function theory. Other resource-bounded reducibilities have been defined and used as well [3] [5] [6] [8]; they differ from Cook's or Karp's only in the bound on time or space allowed for the reduction, and thus they also correspond to Turing or many-one reducibility.

We begin by comparing Cook's and Karp's reducibilities in Section II; as examination of our proof that they are distinct shows that a simple kind of poynomial-bounded truth-table reducibility is actually involved. This leads us to study, in Section III, polynomial-time-bounded analogues of all the usual kinds of truth-table reducibilities. Thus, we are studying restrictions on the form of allowable reduction procedures, as well as on their complexity. Besides basic results about the individual reducibilities, the primary type of result we show is a strong form of distinctness among the various relations.

Further impetus for studying polynomial-time truth-table reducibilities is the hope that exploiting the analogy between recursive function theory and the theory of polynomial-computable functions may help to sove problems such as $P \stackrel{?}{=} NP$. Various results suggest a parallel between the class of recursive sets and P(the class of polynomial-computable sets), as well as between the class of recursively enumerable sets and NP (the class of nondeterministic polynomial-computable sets). For example, we note Cook's characterization of NP [1] using existential quantification. Although the usual argument for showing "recursively enumerable $\not\Rightarrow$ recursive" does not seem to apply in showing $P \neq NP$, we hope that further study of the analogy may provide useful insight into the problem.

This analogy also leads us to wonder whether our results could be strengthened to show that any of the defined reducibilities are distinct on NP. Similar resursive function theory results generally show distinctness on the class of recursively enumerable sets [9]. Of course, this strengthening would require a prior demonstration that $P \neq NP$.

In Section IV, we study nondeterministic versions of polynomial truth-table reducibilities. This investigation is influenced by Gill's work using nondeterministic polynomial Turing reducibility [2], which gives important evidence for the uselessness of two common techniques (simulation and diagonalization) for the solution of $P \stackrel{?}{=} NP$. The structure of the nondeterministic reducibilities turns out to be interesting in itself.

Finally, in Section V, we present open questions arising from this work.

## II Polynomial-time Turing and many-one Reducibilities

We define $\leq_T^P$ and $\leq_m^P$ (polynomial-time Turing reducibility and polynomial-time many-one reducibility) to be the reducibilities used by Cook and Karp respectively. Specifically, we restrict the sets involved in our reducibilities to be recursive sets of strings over the alphabet $\{0,1\}$. We write $|x|$ for the length of string x. Then we write :

$A \leq_T^P B$ iff there is an oracle Turing machine M and a polynomial p such that $x \in A$ exactly if M accepts x with B as its oracle, within $p(|x|)$ steps.

We write:

$A \leq_m^P B$ iff there is a function $f\{0,1\}^* \rightarrow \{0,1\}^*$ computable in polynomial time such that $x \in A$ exactly if $f(x) \in B$.

In all our notation for reducibilities, the subscript (T or m, for example) will indicate the form of the reduction procedure, while the superscript (P, for example) refers to the time bound. We note that our definitions are independent of standard Turing machine conventions, a fact which is often convenient in our proofs; we can use simpler machine models in a diagonalization and more complex models in a simulation construction.

We may easily obtain the following basic facts, similar to basic results about $\leq_T$ and $\leq_m$ in [9]:

Theorem 1: (a) $A \leq_T^P B, B \in P \Rightarrow A \in P$.

(b) $A \leq_m^P B \Rightarrow A \leq_T^P B$.

(c) $\leq_T^P$ and $\leq_m^P$ are reflexive and transitive relations.

(d) $A \leq_m^P B \Leftrightarrow \bar{A} \leq_m^P \bar{B}$.

(e) $A \leq_T^P B \Leftrightarrow \bar{A} \leq_T^P B \Leftrightarrow A \leq_T^P \bar{B} \Leftrightarrow \bar{A} \leq_T^P \bar{B}$

(f) If $A \leq_T^P B$ and for each string $x$, $x \in B$ is decidable in time $\leq t(|x|)$, then for some polynomial $p$, $x \in A$ is decidable in time $\leq p(|x|) + p(|x|) \max \{t(|y|) \mid |y| \leq p(|x|)\}$.

(g) If $A \leq_m^P B$ and for each string $x$, $x \in B$ is decidable nondeterministically in time $\leq t(|x|)$, then for some polynomial $p$, $x \in A$ is decidable nondeterministically in time < $\leq p(|x|) + \max \{t(|y|) \mid |y| \leq p(|x|)\}$.

(h) $A \leq_m^P B$, $B \in NP \Rightarrow A \in NP$.

All of these results have elementary proofs, and many have been previously noted. The key idea in (a), (f), (g) and (h) is direct simulation of the oracle. (g) and (h) are not known to hold for $\leq_T^P$. ((h) for $\leq_T^P$ would imply that NP is closed under complement.) This fact, together with the following, provides good reason for considering reducibilities other that $\leq_T^P$:

Proposition: (Meyer) Let A be any $\leq_m$-complete set in NP. Then A and $\bar{A}$ are m-comparable iff NP is closed under complement.

Degree-theoretic results about $\leq_T^P$ and $\leq_m^P$ are explored in [5]. We now wish to show that $\leq_T^P$ and $\leq_m^P$ are distinct; to do so we use the following notion of distinctness:

Definition: Given any 2 reducibilities, $\leq_1$ and $\leq_2$, we say "$\leq_1$ transcends $\leq_2$" if there exist recursive sets A and B such that $A \leq_1 B$, $B \leq_1 A$, $A \not\leq_2 B$ and $B \not\leq_2 A$. (That is, A and B are 1-equivalent but 2-incomparable.)

Theorem 2: $\leq_T$ transcends $\leq_m^P$

Proof: Immediate from Theorem 1 and the following Lemma:

Lemma: There exists an infinite, coinfinite recursive set A such that $\bar{A} \not\leq_m A$.

Proof of the Lemma: The set A is constructed in stages numbered 0, 1, 2, ... At each stage, we attempt to diagonalize over a many-one reduction procedure. Thus, we need an effective enumeration of polynomial-time-bounded reduction procedures; it is sufficient to select some recursive function b which is eventally greater than each polynomial (that is, $b: \{0,1\}^* \to N$, and ($\forall p$, a polynomial) $(\exists x)(\forall x | |x| \geq \ell)[b(x) \geq p(|x|)]$ ), and use it as a bound on the number of steps in the computation of Turing machines in some ordinary Gödel numbering $\{M_i\}$. Since we only know that $b(x)$ is eventually greater than $p(|x|)$, we return to consider each machine $M_i$ infinitely often. Let $\pi_1$ and $\pi_2$ be the projection functions for some pairing function $N \times N \to N$ (e.g., see [9])

Stage y: Let x be the first string (in a natural ordering of binary strings) whose membership in A is not yet determined. Let $i = \pi_1(y)$.

See if $M_i$ on input x converges within $b(x)$ steps. If not, define $x \in A \Leftrightarrow y$ is even, and go on to stage $y + 1$. Otherwise, let $\phi_i(x)$ be the output. We wish to falsify: $x \in \overline{A} \Leftrightarrow \phi_i(x) \in A$.

If $\phi_i(x)$'s membership in A is already determined, we define: $x \in \overline{A} \Leftrightarrow \phi_i(x) \in A$. Otherwise, we define $x \in A$ and $\phi_i(x) \in A$. Go on to stage $y + 1$.

END OF CONSTRUCTION

A is clearly recursive, and the reader may verify that for pairing functions chosen as in [9], for example, A is infinite and coninfinite. Now if $\overline{A} \leq^P_m A$ via the polynomial computable function f, then $(\exists i)(\exists p$, a polynomial) $[x \in \overline{A} \Leftrightarrow \phi_i(x) \in A$, and $M_i(x)$ runs in $\leq p(|x|)$ steps ] But for x sufficiently long, $b(x) \geq p(|x|)$, and for some y sufficiently large, $\pi_1(y) = i$, causing the condition $[x \in \overline{A} \Leftrightarrow \phi_i(x) \in A]$ to be falsified at stage y. ☒

The above proof is simple but is presented since many of the results to follow can be proved by essentially similar ideas. This Lemma shows that Theorem 1 (d) cannot be strengthened analogously to (e).

As noted in Section I, it would be desirable to know on what complexity classes of sets the reducibilities can be shown to differ For example, can a set A as in the Lemma be constructed with $A \in NP$? More tractably, can we show that $P \neq NP$ would imply the existence of such a set A in NP? If we naively measure the complexity of the set A constructed in the Lemma, we note that it is roughly $2^{2^{|x|}}$ on argument x, since this much time is required to simulate and keep track of the results of enough stages in the construction to determine if $x \in A$. However, the diagonalization construction is very "loose," in that there are few constraints on our choice of x at each stage. Thus, by choosing the values of x to be sufficiently separated (a technique due to Machtey) it requires sufficiently less time to simulate the computations of preceding stages to bring the complexity down to $2^{|x|}$. (Strings not used in the diagonalization can have their membership in A determined arbitrarily). So $\leq^P_T$ and $\leq^P_m$ can at least be shown to differ on the exponentially-computable sets. The same technique could also be applied to all transcendence results in

Sections III and IV, reducing the complexity of all relevant sets to $2^{|x|}$.

The technique used in the proof of the Lemma actually yields results more powerful than stated. First, the function b may be chosen as large as we like; for example, if we choose b so that b is eventually greater than each primitive recursive function of the length of its argument, then a set A is produced with $\bar{A}$ not many-one reducible to A in primitive recursive time. Second, we see that it is not only $\underset{T}{\leq}^P$ that transcends $\underset{m}{\leq}^P$, but a very simple form of polynomial-bounded procedure, involving asking only a single oracle question. Since this is an obvious analogue to $\underset{1\text{-}tt}{\leq}$ (one-question truth-table reducibility), we are led to define polynomial-bounded truth-table reducibilities:

## III Polynomial-time Truth-table Reducibilities

We recall that A is tt - reducible (truth-table reducible) to B if, given any x, one can effectively compute both a finite set of arguments $x_1, x_2, \ldots, x_k$, and a Boolean function $\alpha$ such that:

$$x \in A \Leftrightarrow \alpha(C_B(x_1), C_B(x_2), \ldots, C_B(x_k)) = 1,$$

where $C_B$ is the characteristic function of B. This differs from Turing reducibility in allowing one, given any x, to effectively compute ahead of time the entire set of questions that might be asked during the computation. That is, the choice of questions to ask cannot depend on the oracle set B. Our definition of polynomial-time tt- reducibility requires that both the generation of the set and function, and the computation of $\alpha$, must be polynomial-time-bounded. If we were to restrict our attention to a specific representation of Boolean functions, say one using only the symbols $\wedge$, $\vee$ and $\neg$, then the polynomial bound on the generation of the set and function is a sufficient requirement for our defintion, as it implies a polynomial bound on the evaluation time. However, we wish to leave the representation of the function arbitrary, so both restrictions are needed. It is unknown whether a less general reducibility would result by restriction to $\wedge$, $\vee$ and $\neg$.

We let $\Delta$ be a fixed finite alphabet, for the encoding of Boolean functions, and let $c \notin \Delta \cup \{0,1\}$.

Definition: A tt-condition is a member of $\Delta^* c(c\{0,1\}^*)^*$ A tt-condition generator is a recursive mapping from $\{0,1\}^*$ into $\Delta^* c(c\{0,1\}^*)^*$.

A tt-condition evaluator is a recursive mapping from $\Delta^* c\{0,1\}^*$ into $\{0,1\}$.

Let e be a tt-condition evaluator.

A tt-condition $\alpha c c x_1 c x_2 c \cdots c x_k$ is e-satisfied by $B \subseteq \{0,1\}^*$ iff $e(\alpha c\, C_B(x_1) C_B(x_2) \cdots C_B(x_k)) = 1$.

$A \underset{tt}{\leq}^P B$ iff there exist a polynomial-time computable generator g and a polynomial-time computable evaluator e such that $x \in A \Leftrightarrow g(x)$ is e-satisfied by B.

Polynomial analogues of various special cases of tt-reducibilities may now be defined by placing restrictions on the generator, the evaluator, or both:

Definition: $A \underset{btt}{\leq}^P B$ (A is polynomial-time bounded-truth-table reducible to B) provided

$A \leq^P_{tt} B$ by a generator g and an evaluator e, such that g produces words with a bounded number of c's.

$A \leq^P_{k\text{-}tt} B$ (A is polynomial-time k-question truth-table reducible to B) for any integer k,

if $A \leq^P_{tt} B$ via g and e, and

$g: \{0,1\}^* \to \Delta^* c\, (c\, \{0,1\}^*)^k$.

$A \leq^P_p B$ (A is polynomial-time positive reducible to B) if the evaluator e has the property that

$[e(\alpha c\, \sigma_1 \sigma_2 \cdots \sigma_k) = 1 \wedge (\sigma_i = 1 \Rightarrow \tau_i = 1)]$

$\Rightarrow [\, e(\alpha c\, \tau_1 \tau_2 \cdots \tau_k) = 1\,]$.

$A \leq^P_c B$ (A is polynomial-time conjunctive reducible to B) if the evaluator e has the property that

$e(\alpha c\, \sigma_1 \sigma_2 \cdots \sigma_k) = 1 \Leftrightarrow \sigma_1 = \sigma_2 = \cdots = \sigma_k = 1$.

$A \leq^P_d B$ (A is polynomial-time disjuctive reducible to B) if $e(\alpha c\, \sigma_1 \sigma_2 \cdots \sigma_k) = 0$

$\Leftrightarrow \sigma_1 = \sigma_2 = \cdots = \sigma_k = 0$.

Corresponding to Theorem 1, we obtain:

<u>Theorem 3:</u> (a) For any $k \geq 1$, we have the implications:

$A \leq^P_m B \Rightarrow A \leq^P_{k\text{-}tt} B \Rightarrow A \leq^P_{k+1\text{-}tt} B \Rightarrow A \leq^P_{btt} B \Rightarrow$

$A \leq^P_{tt} B \Rightarrow A \leq^P_T B$.

(b)
$$\begin{array}{ccccc} & & A \leq^P_c B & & \\ & \nearrow & & \searrow & \\ A \leq^P_m B & \rightarrow A \leq^P_d B & \nearrow & A \leq^P_p B & \Rightarrow A \leq^P_{tt} B. \end{array}$$

(c) All are transitive.

(d) $A \leq^P_{tt} B \Leftrightarrow \bar{A} \leq^P_{tt} B \Leftrightarrow A \leq^P_{tt} \bar{B} \Leftrightarrow \bar{A} \leq^P_{tt} \bar{B}$.

(The same is true for btt and k-tt.)

(e) $A \leq^P_p B \Leftrightarrow \bar{A} \leq^P_p \bar{B}$.

(f) $A \leq^P_c B \Leftrightarrow \bar{A} \leq^P_d \bar{B}$.

(g) $A \leq^P_p B,\ B \in NP \Rightarrow A \in NP$.

For implications not given in Theorem 3, we obtain the following transcendence results:

<u>Theorem 4:</u> (a) $\leq^P_{1\text{-}tt}$ transcends $\leq^P_m$.

(b) For any k, $\leq^P_{k+1\text{-}c}$ transcends $\leq^P_{k\text{-}tt}$

$\leq^P_{k+1\text{-}d}$ transcends $\leq^P_{k\text{-}tt}$.

(k+1-c and k+1-d refer to k+1-question conjuctive and disjunctive reducibilities respectively, defined by the obvious restrictions on the generator and evaluator.)

(c) $\leq^P_{2\text{-}c}$ transcends $\leq^P_d$; $\leq^P_{2\text{-}d}$ transcends $\leq^P_c$.

(d) $\leq^P_{4\text{-}p}$ transcends both $\leq^P_c$ and $\leq^P_d$, (and both may be done with the same pair of sets).

(e) $\leq^P_{1\text{-}tt}$ transcends $\leq^P_p$.

(f) $\leq^P_{tt}$ transcends $\leq^P_{btt}$.

<u>Notes on the proofs:</u> (a) was proved for Theorem 2. (e) is proved similarly, by constructing an infinite, coinfinite set A with $\bar{A} \not\leq^P_p A$. For the other cases, we need to

construct a pair of sets A and B, preserving a 2-sided reducibility of the first type, while conducting a 2-sided diagonalization over reducibilities of the second type. For example, in constructing sets A and B for (d), we preserve the conditions:

$$z11 \in A \Leftrightarrow (z110 \in B \wedge z1100 \in B) \vee (z11000 \in B \wedge z110000 \in B)$$

$$z01 \in B \Leftrightarrow (z010 \in A \wedge z0100 \in A) \vee (z01000 \in A \wedge z010000 \in A)$$

$$\left.\begin{array}{l} z110^k \in A \Leftrightarrow z110^k \in B \\ z010^k \in A \Leftrightarrow z010^k \in B \end{array}\right\} \quad \text{for } 1 \le k \le 4,$$

for all strings z. All membership questions not specifically mentioned will be answered negatively. These conditions are strong enough to force $A \le^P_{4-p} B$ and $B \le^P_{4-p} A$, and weak enough to allow us to diagonalize over all $\le^P_c$ and $\le^P_d$ procedures. For instance, to show $A \not\le^P_c B$, we define:

Stage 4y: Let $i = \pi_1(y)$. Let x be the next string of the form z11 such that none of the questions $z11 \in A$, $z110^k \in B$, $z110^k \in A$, $1 \le k \le 4$ have been answered. Consider $M_i$ on input x for b(x) steps, as before. If it halts, we consider the set Q of questions it outputs. If we already have, or if it is possible to define $q_0 \notin B$ for some $q_0 \in Q$, then we define $q_0 \notin B$, $x \in A$, $q \in B$ for all undetermined $q \in Q$, $q \ne q_0$, $z110^k \in B$ for $1 \le k \le 4$ and $z110^k \ne q_0$, and other values as required to preserve the above conditions. Otherwise, we define $x \notin A$, $q \notin B$ for all undetermined $q \in Q$, $z110^k \notin B$ for $1 \le k \le 4$, and other values as required. END OF CONSTRUCTION

The reader may complete the proof and verification. ☒

Again, judicious choice of arguments on which to diagonalize will bring the complexity of A and B down to $2^{|x|}$. Also, as before, all the results in Theorem 4 may be strengthened by making the bound b as large as desired. The same is not true for the following result:

Theorem 5: $\le^P_T$ transcends $\le^P_{tt}$.

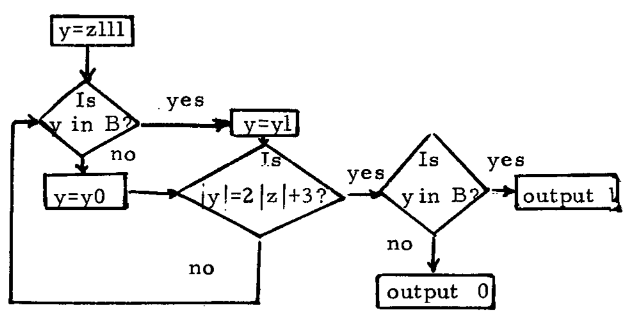
Proof: We must use a small function for b because of the following:

Note: $A \le^P_T B \Rightarrow A \le^E_{tt} B$ (where $\le^E_{tt}$ refers to exponential-time tt-reducibility, defined analogously to $\le^P_{tt}$, but with a time bound of the form $2^{p(|x|)}$ rather than $p(|x|)$.)

Thus, we let $b(x) = 2^{|x|} - 1$. We will obtain $\le^P_T$ by preserving the following conditions:

If $z \in (\{0,1\}0)^*$, then:

$z \in A \Leftrightarrow 1$ is outputted by the following procedure:




$z \in B \Leftrightarrow 1$ is outputted by the same flowchart, starting with y=z110, using oracle A.

If $w \in \bigcup_{1 \le k \le |z|+1} z11\{0,1\}^k$, then $w \in A \Leftrightarrow w \in B$. Otherwise, membership questions will be answered negatively. To obtain $A \not\le^P_{tt} B$ :

Stage 2y: Let $i = \pi_1(\pi_1(y)), j=\pi_2(\pi_1(y))$. Let $x$ be the next string in $(\{0,1\}0)^*$ such that none of the question $x \in A$, $w \in A$ or $B$ for $w \in \bigcup_{0 \le k \le |x|} x111\{0,1\}^k$ have been answered.

If $M_i$ on input $x$ converges within $b(x)$ steps, consider the tt-condition $\phi_i(x)$. Note that at most $b(x) = 2^{|x|}-1$ questions are represented in the truth table, so some $w \in x111\{0,1\}^{|x|}$ is not in the truth table. We define membership in $B$ of elements of $\bigcup_{0 \le k < |x|} x111\{0,1\}^k$ and of values in the truth table in some way that causes the above flowchart, for $z = x$, to eventually ask whether $w \in B$.

Say $\phi_i(x) = \alpha c c x_1 c x_2 c \cdots c x_n$. Then if $M_j$ on input $\alpha c\, C_B(x_1) C_B(x_2) \cdots C_B(x_n)$ converges within $b(x)$ steps, we define:

$w \in B \Leftrightarrow x \in A \Leftrightarrow \phi_j(\alpha c C_B(x_1) C_B(x_2) \cdots C_B(x_n)) = 0$,

and other values as required to preserve the above conditions.

END OF CONSTRUCTION

As before, we leave the reader to complete the construction and verification. ☒

We complete our consideration of deterministic tt-reducibilities by noting the following two equivalent formulations of our definition of $\le^P_{tt}$:

Definition: $A \le^P_{tt} B$ iff there exists a polynomial-time computable function $f: \{0,1\}^* \to \Delta^* c(c\{0,1\}^*)^*$ such that if $f(x) = \alpha c c x_1 c \cdots c x_k$ then $\alpha$ is a combinational circuit (having only $\wedge, \vee$ and $\neg$ gates) [10] and $x \in A \Leftrightarrow \alpha(C_B(x_1), \cdots C_B(x_k)) = 1$.

Definition: $A \le^P_{tt} B$ iff there is an oracle Turing machine $M$ and a polynomial-time computable function $f: \{0,1\}^* \to (c\{0,1\}^*)^*$ such that $M$ reduces $A$ to $B$ within polynomial time, and for this computation, on input $x$, $M$ only asks questions that are on the list $f(x)$.

We note that this last definition describes a sort of weak truth-table reducibility.

## IV Nondeterministic Reducibilities

A natural way to generalize the definitions in Sections 2 and 3 is to allow nondeterminism in the reducibility procedures. The first place in which an interesting application of a nondeterministic reducibility appears is in G ll's paper [2]. He shows, for $\le^{NP}_T$ a reasonable notion of nondeterministic Turing reducibility, that

(1) there exist recursive sets $B$ with $A \le^P_T B \Leftrightarrow A \le^{NP}_T B$, and

(2) there exist recursive sets $B$ with the above equivalence false.

Since both diagonalization and simulation proofs generally extend from the non-oracle to the oracle case, these results seem to show that neither a diagonalization nor a simulation will probably be useful in deciding whether P=NP.

We define nondeterministic reducibilities:

Definition: $A \leq_T^{NP} B$ (A is nondeterministic polynomial-time Turing reducible to B) iff there is a nondeterministic oracle Turing machine M and a polynomial p such that with oracle B, M runs in time bounded by p for all possible courses of computation, and $x \in A \Leftrightarrow$ M with oracle B, input x has some accepting computation.

$A \leq_m^{NP} B$ iff there is a nondeterministic Turing machine transducer M and a polynomial p such that M runs in time bounded by p for all possible courses of computation, and

$x \in A \Leftrightarrow$ some course of M's computation on x yields as output a value $y \in B$.

$A \leq_{tt}^{NP} B$ iff there is a nondeterministic Turing machine transducer M, polynomial-bounded as above, and a polynomial-time-computable evaluator e such that

$x \in A \Leftrightarrow$ some possible computation of M on input x generates a tt-condition which is e-satisfied by B.

Note: This definition allows nondeterminism to be introduced into the generator but not into the evaluator. Allowing nondeterminism in the evaluator as well adds no extra pairs to the reducibility.

We make appropriate modifications in the last definition to obtain definitions for

$\leq_{btt}^{NP}$, $\leq_{k\text{-}tt}^{NP}$, $\leq_p^{NP}$, $\leq_c^{NP}$ and $\leq_d^{NP}$.

We first note that a collapse occurs which is very different from the deterministic case. In part, the following theorem suggests that nondeterminism recovers the power of using disjunctions:

Theorem 6: $A \leq_T^{NP} B \Leftrightarrow A \leq_{tt}^{NP} B$

$$A \leq_p^{NP} B \Leftrightarrow A \leq_c^{NP} B$$

$$A \leq_d^{NP} B \Leftrightarrow A \leq_m^{NP} B$$

Proof: For the second equivalence, for example, assume $A \leq_p^{NP} B$. Given x, nondeterministically generate all the appropriate positive tt-conditions. For each such condition, $\alpha c c x_1 c \cdots c x_k$, nondeterministically assign values $v(x_i) = 0$ or 1 to each $x_i$, and simulate the evaluator e on input $\alpha c\, v(x_0) \cdots v(x_k)$. For each such assignment of values of v, output the appropriate form of:

$$\begin{cases} a \wedge \neg a & \text{(for some string a) if } e(\alpha c v(x_0) \cdots v(x_k)) = 0, \\ \bigwedge_{x_i \mid v(x_i) = 1} x_i & \text{if } e(\alpha c v(x_0) \cdots v(x_k)) = 1. \end{cases}$$

This procedure provides a nondeterministic generator witnessing $A \leq_c^{NP} B$; we leave the reader to supply the remaining details as well as the similar proofs for the other two equivalences. ☒

For transitivity, nondeterministic results again differ from deterministic results:

Theorem 7: (a) $\leq_m^{NP}$ and $\leq_c^{NP}$ are transitive.

(b) $\leq_{tt}^{NP}$, $\leq_{btt}^{NP}$ and $\leq_{k\text{-}tt}^{NP}$ (for any k) fail to be transitive.

Notes on the proof: (a) is by straightforward simulation. For (b), we prove the lemma:

Lemma: There exist recursive sets A, B, C with $A \leq^{NP}_{1-tt} B$, $B \leq^{NP}_{1-tt} C$ but $A \not\leq^{NP}_{tt} C$.

Proof of Lemma: We preserve the conditions

$x \in A \Leftrightarrow (\exists y)[\,|y| = |x| \text{ and } y \in \overline{B}]$

$x \in B \Leftrightarrow (\exists y)[\,|y| = |x| \text{ and } y \in \overline{C}\,]$.

Within this framework, we diagonalize as in previous proofs, over $\leq^{NP}_{tt}$-procedures. The bound b in this proof is chosen to be small ($2^{|x|}-1$, for example), a necessity because of the following limiting result:

$A \leq^{NP}_{1-tt} B$, $B \leq^{NP}_{1-tt} C \Rightarrow A \leq^{NE}_{tt} C$, (where $\leq^{NE}_{tt}$ refers to nondeterministic exponential-time tt-reduciblity, defined analogously to $\leq^{NP}_{tt}$, but with a time bound of the form $2^{p(|x|)}$ rather than $p(|x|)$).

Again, remaining details are left to the reader. ☒

Note: We may also easily show $A \leq^{NP}_{c} B$, $B \in NP \Rightarrow A \in NP$.

We conjecture, but have not yet proved, that $\leq^{NP}_{c}$ is a maximal transtive subset of $\leq^{NP}_{T}$.

For nondeterministic reducibilities whose definitions do not collapse, transcendence results become stronger than in the deterministic case. Namely, we show that deterministic reducibilities transcend the appropriate nondeterministic reducibilities: Compare with Theorem 4:

Theorem 8: (a) $\leq^{P}_{1-tt}$ transcends $\leq^{NP}_{m}$.

(b) For any k, $\leq^{P}_{k+1-c}$ transcends $\leq^{NP}_{k-tt}$.

(The corresponding statement is false for $\leq^{P}_{k+1-d}$, by Theorem 6.)

(c) $\leq^{P}_{2-c}$ transcends $\leq^{NP}_{d}$.

(The corresponding statement is false for c and d interchanged.)

(d) $\leq^{P}_{1-tt}$ transcends $\leq^{NP}_{p}$.

(e) $\leq^{P}_{tt}$ transcends $\leq^{NP}_{btt}$.

Notes on proofs: Basically, within the frameworks used to preserve the reducibilities in Theorem 4, we are actually able to diagonalize over more procedures, nondeterministic as well as deterministic. For example, for (a) we construct an infinite, coinfinite set A with $\overline{A} \not\leq^{NP}_{m} A$. For (d), we construct an infinite, coinfinite set A with $A \not\leq^{NP}_{p} \overline{A}$ and $\overline{A} \not\leq^{NP}_{p} A$ (The diagonalization must be done in two directions). ☒

Finally, parallel to the existential quantifier characterization of NP given by Cook, we have the following equivalent formulation of the definition of $\leq^{NP}_{m}$:

Definition: $A \leq^{NP}_{m} B$ iff there is a polynomial p and a polynomial-time computable function f such that

$x \in A \Leftrightarrow (\exists y)[\,|y| \leq p(|x|) \text{ and } f(x, y) \in B]$

Similar characterizations exist for the other nondeterministic reducibilities.

## V Further Study

We know that our deterministic reducibilities differ on the exponential-computable sets. We would like to show that they differ on NP (for example, that there exist $A, B \in NP$ with $A \leq^P_T B$ but $A \not\leq^P_m B$). This, of course, would imply $P \neq NP$. Perhaps we can show:

$$P \neq NP \Rightarrow \leq^P_T \text{ and } \leq^P_m \text{ differ on } NP.$$

More strongly, perhaps we can show:

$P \neq NP \Rightarrow$ T-completeness and m-completeness differ. Same questions for the other deterministic reducibilities.

We would like to develop stronger notions of distinctness between reducibilities, than "transcendence." For example, can we show that $\leq^P_T$ and $\leq^P_m$ differ in the following way:

$$(\forall A \notin P)(\exists B)[A \leq^P_T B \text{ but } A \not\leq^P_m B] ?$$

In our definition of $\leq^P_{tt}$, if we restrict consideration to truth-table conditions with a specific representation (such as using $\wedge, \vee, \neg$ only), do we obtain as less general reducibility?

We may define a new reducibility, analogous to enumeration reducibility, as follows:

$$A \leq^{NP}_e B \quad \text{iff} \quad (\forall X)[B \leq^{NP}_T X \Rightarrow A \leq^{NP}_T X].$$

It is easy to show that $\leq^{NP}_c \subseteq \leq^{NP}_e \not\subseteq \leq^{NP}_T$.

Further, $\leq^{NP}_e$ is transitive and in fact is maximal transitive in the following sense:

$$\leq^{NP}_e \subseteq R \subseteq \leq^{NP}_T,\ R \text{ transitive} \Rightarrow R = \leq^{NP}_e.$$

We ask whether $\leq^{NP}_e = \leq^{NP}_c$; we conjecture that they are equal, which would make $\leq^{NP}_c$ a maximal transitive reducibility.

Degree-theoretic questions about all the reducibilities remain, as well as questions about complete sets at various complexity levels. These may someday prove relevant to a classification of natural problems by their complexity.

Acknowledgments: We would like to thank Albert Meyer and Michael Machtey for some very valuable suggestions on this work.

References:

1. Cook, S.A. The complexity of theorem-proving procedures. Third annual ACM Symposium on Theory of Computing (1971).

2. Gill, J.T. Axiomatic Independence of the question NP=P? Department of Electrical Engineering, Standford University. (1972).

3. Jones, N.D. Reducibility among combinatorial problems in log n space, Proceedings of Seventh Annual Princeton Conference on Information Sciences and Systems (1973)

4. Karp, R.M. Reducibility among combinatorial problems. Complexity of Computer Computations. Miller and Thatcher (eds.) Plenum Press(1973).

5. Ladner, R. E. Polynomial time reducibility, Fifth Annual ACM Symposium on Theory of Computing (1973).

6. Lynch, N.A. Relativization of the Theory of computational complexity, Ph.D thesis MIT (1972).

7. Meyer, A.R. and Stockmeyer, L.J. The equivalence problem for regular expressions with squaring requires exponential space, 13th Annual IEEE Symposium of Switching and Automata Theory (1972).

8. Meyer, A.R. and Stockmeyer, L.J. Word problems requiring exponential time. Fifth Annual Symposium on Theory of Computing (1973).

9. Rogers, H. Theory of recursive functions and effective computability. McGraw-Hill(1967).

10. Savage, J.E. Computational Work and Time on Finite Machines. JACM, Vol.19, No.4(1972).

## Addendum

Contemporary with Gill [2], parallel results have been independently obtained by T. Baker (Computational Complexity and Nondeterminism in Flowchart Programs, Ph.D. thesis, Cornell University, 1973).