Cook Reducibility Is Faster than
Karp Reducibility in NP

Luc Longpré* and Paul Young†

Department of Computer Science, FR-35,
University of Washington, Seattle, Washington 98195

Received October 25, 1988; revised August 25, 1989

It is unknown whether Cook reducibility of a set \( A \) to a set \( B \)—that is reduction of \( A \) to \( B \) via a Turing machine operating in polynomial time with “free” procedural calls to an algorithm for \( B \)—is more general than Karp reducibility—that is reduction of \( A \) to \( B \) via a function computable in polynomial time—on sets in NP. While we conjecture that Cook reducibility is indeed a more general notion than Karp reducibility on sets in NP, proving this would imply that \( P \neq NP \). Here we investigate more tractable subcases of the problem. For example, we prove that Cook reducibility is much faster than Karp reducibility on some classes of NP-complete sets. We also prove that for some classes of NP-complete sets Cook reductions between members of the classes can be both linearly fast and “linearly honest” while any Karp reduction must be highly “dishonest.” © 1990 Academic Press, Inc.

1. INTRODUCTION

Intuitively, we say that a set \( C \) is NP-complete if \( C \in NP \) and if in addition \( C \) can be shown to be at least as “hard” as every member of NP. The latter condition may be formalized by saying that every member of NP is reducible to \( C \) in polynomial time. In most textbooks, e.g., [GJ79, HU79, LPSl], and in many papers, the notion of reducibility used in this context is that of a polynomial time many-one reduction, i.e., a functional reduction that can be computed in polynomial time. These are the so-called “Karp” reductions. However, if one is looking for an “absolute” definition of NP-complete, then one wants to fully capture the idea that if \( C \) is decidable in polynomial time then every other member of NP is also decidable in polynomial time. In this context the philosophically “correct” reductions to use in defining NP-complete sets are general polynomial time Turing reductions. These reductions are basically reductions by programs having “free” access to procedures which answer arbitrary membership questions about the complete set \( C \). These are the so-called “Cook” reductions.

The general purpose of this paper is to investigate what “harm” is caused by using only Karp reductions rather than using more general Cook reductions in...
studying completeness notions and reductions on sets in NP. Ladner, Lynch, and Selman \cite{LLS75} were the first to show that these reducibilities, as well as intermediate reducibilities such as polynomial truth-table-reducibilities, differ, although not in NP. Later, Selman \cite{Sel79, Sel82} showed differences for sets in NP under reasonable assumptions about class separations, although Selman’s methods cannot be applied to complete sets. In \cite{KM81}, Ko and Moore proved that Cook and Karp reducibilities give different complete sets for exponential time, $E = \defeq \bigcup_c \text{TIME}[2^c]$. A more recent investigation by Watanabe \cite{Wat87a}, has further shown that various polynomial time truth-table-reducibilities also generate successively more general classes of complete sets not only for $E$ but also for \text{NP} : \defeq \bigcup \text{NTIME}[2^c] \cite{Wat87b}. In his classic 1944 paper, \cite{Pos44}, Post showed that successively more general notions of computable reductions generate successively larger classes of sets complete for the collection of all recursively enumerable (r.e.) sets.

We believe that, just as for the corresponding questions about polynomial time reductions on sets in E and about computable reductions on the r.e. sets, there are surely sets in NP that are complete for polynomial time Turing reductions but not complete for polynomial time functional reductions. Proving this would, in a certain sense, prove that Cook reductions are the “correct” reductions to use in studying, and in defining, NP-complete sets. This is a long term goal: proving such a result for NP would also prove \text{P} \neq \text{NP}.

Thus we settle here for several lesser results along these lines, answering questions raised seven years ago in \cite{You83}. The first of these tells us that Cook reductions are in fact much faster than Karp reductions on some classes of NP-complete sets. Specifically, given any polynomial $p$, we construct NP-complete sets $A$ and $B$ such that any Karp reduction of $A$ to $B$ requires time at least $p$, but $A$ can be reduced to $B$ by a (simple) Cook reduction that takes only linear time. The second of these results constructs complete sets $A$ and $B$ which have similar simple Cook reductions between them, but, given any polynomial $p$, any Karp reduction between the sets must be at least $p$ “dishonest.” While results of this form are not surprising, they do give new, and absolute, knowledge of how Cook and Karp reducibilities differ in NP, knowledge that yields basic new information about how reducibilities work in NP.

Furthermore, the techniques used in proving our main theorems turn out to be more subtle than one might expect. In our first proof, one has to diagonalize against all polynomially time computable functions of complexity $\leq n^k$, while maintaining a linear time Turing relationship between $A$ and $B$. Traditionally, in a diagonalization of this kind, one has to perform complementations in order to realize the diagonalizations. But since sets in NP are presumably not closed under complements, standard diagonalization arguments do not work to construct sets in NP. In \cite{Coo73}, Cook overcomes this problem by giving a diagonalization proving that $\text{NP}[n^{O}] \subseteq \text{NP}[n^{r}]$ for any real constants $0 \leq r_0 < r - 1$. Discussing this problem in a different nondeterministic context, \cite{KMR86} call for the development of new techniques not directly dependent on complementation. In the proof
of our main theorem we use a diagonalization technique that directly exploits the nondeterminism of sets in NP. In fact, by exploiting a combinatorial argument, we are able to make direct use of nondeterminism instead of direct complementation in order to realize the necessary diagonalizations. The proof of our second main result is by a priority method. The use of priority methods is rare in concrete complexity theory.

**Definition 1.** A set $A$ Turing reduces to $B$ ($A \leq_T B$) in time $T(n)$ if there is an oracle Turing machine $M$ which accepts $A$ in time $T(n)$ using $B$ as oracle. If $T(n)$ is bounded by a polynomial we say that $A$ is Cook reducible to $B$ (or $A \leq^P_T B$).

**Definition 2.** A set $A$ many-one reduces to $B$ ($A \leq_m B$) in time $T(n)$ if there is a function $f$ computable in time $T(n)$ such that $x \in A \iff f(x) \in B$. If $T(n)$ is bounded by a polynomial we say that $A$ is Karp reducible to $B$ (or $A \leq^P_m B$).

We will also consider $k - TT$ reductions between sets, with particular emphasis on disjunctive truth-table reducibilities:

**Definition 3.** For any constant $k$, we say that $A \leq^P_{k - TT} B$ in time $T(n)$ if there is a function which, on input $x$, in time $T(|x|)$, computes $k$ strings $x_1, \ldots, x_k$ and a predicate, $PRED$, of $k$ boolean variables, such that $x \in A \iff PRED(a_1, a_2, \ldots, a_k)$, where $a_i$ is a boolean value which is true $\iff x_i \in B$. When $PRED(x_1, \ldots, x_k)$ is always the disjunction $"x_1 \lor \ldots \lor x_k"$ we say that $A$ is $k$-disjunctive truth-table reducible to $B$, and we write $A \leq^P_{k - disj} B$.

Clearly, for any $k$, $\leq^P_m \Rightarrow \leq^P_{k - disj} \Rightarrow \leq^P_{k - TT} \Rightarrow \leq^P_T$, either for polynomial or for more general computable reductions.

2. **Disjunctive Reductions and Polynomial Time Many-One Reductions**

Our main theorem compares $\leq^P_{2 - disj}$ and $\leq^P_m$ reducibilities among Karp-complete sets.

**Theorem 4.** For all $k \geq 1$, there exist sets $A$ and $B$ which are Karp-complete for NP and such that $A \leq^P_{2 - disj} B$ in linear time, but $A \leq^P_m B$ requires time at least $n^k$.

**Proof.** We will construct the sets $A$ and $B$ in stages. We begin by defining a function $g$ used to separate strings so that various stages of the construction do not unduly interfere with each other. We define $g$ by $g(1) = 0$ and $g(i + 1) = (g(i) + 2)k + 1$. We let $\varphi_1, \varphi_2, \varphi_3, \ldots$ be an enumeration of all $n^k$ time bounded functions. We will give a direct construction of $A$, letting the set $B$ be constructed from $A$. The construction of $A$ will be by stages. At stage $i$, we will determine membership in $A$ for all strings of length $g(i)$, $g(i) + 1, \ldots, (g(i) + 2)^k = g(i + 1) - 1$. Our chief goal at stage $i$ will be to make sure that the $i$th $n^k$ many-one reduction does
not reduce $A$ to $B$. To accomplish this, we spoil $\phi_i$ by making sure that, for at least one string $w$ of length $g(i) + 2$, either $w \in A$ and $\phi_i(w) \notin B$, or $w \notin A$ and $\phi_i(w) \in B$. Obviously doing this for every value of $i$ will guarantee that $A$ has no "fast," i.e., no $n^k$, functional reduction to $B$. To keep the set $A$ complete, the strings of size $g(i)$ will be used to encode a standard NP-complete set into $A$. The strings for the remaining sizes will be put into $A$. This will make some cases of the diagonalization easy. The NP-completeness of $B$ will be an easy consequence of how $B$ depends on $A$.

Let the sets $C_i$ be the strings used for coding and the sets $D_i$ be used for the diagonalization:

$$C_i = \{w : |w| = g(i)\}$$

$$C = \bigcup_i C_i$$

$$D_i = \{w : |w| = g(i) + 2\}$$

$$D = \bigcup_i D_i.$$  

The following sets will be used for diagonalization:

$$I_0 = \{0w : \phi_i(w) = 0w \text{ and } w \in D_i\}$$

$$I_1 = \{1w : \phi_i(w) = 1w \text{ and } w \in D_i\}$$

$$I = I_0 \cup I_1$$

Clearly, $D \in P$, $C \in P$, and $I \in P$, where $P$ is the class of sets recognizable in polynomial time. Notice also that for each $w$ it is not possible to have both $0w \in I$ and $1w \in I$.

The set $B$ will be $(0A \cup 1A) - I$. Because $I$ cannot contain both $0w$ and $1w$, we will thus have that if $w \in A$, either $0w \in B$ or $1w \in B$. Moreover, if $w \notin A$, then $0w \notin B$ and $1w \notin B$. This gives us $A \leq_{2 \text{- disj}}^P B$ in linear time by the following reduction:

$$w \in A \iff [0w \in B \text{ or } 1w \in B].$$

Now, if $A$ is NP-complete, then

1. $B$ is in NP, because $B = (0A \cup 1A) - I$ and $I \in P$.

2. $B$ is Cook-complete, because $A \leq_{2 \text{- disj}}^P B$ in linear time. In fact, since $I \in P$, it is easily seen that $A \leq_m^P B$ (although the reduction will take time at least $n^k$), so if $A$ is Karp-complete $B$ will also be Karp-complete.

The set $A$ is defined by the nondeterministic polynomial time algorithm below. After defining $A$, we will show that $A$ is complete and that $A \not\leq_m^P B$ in time $n^k$.

To determine whether a given string $w$ is in $A$, compute $i$ such that $g(i) \leq |w| < g(i + 1)$. See if $w$ is in $C_i$ or $D_i$. If $w$ is in $C_i$, then accept $w$ iff $w \in 0^*1\text{ SAT}$. If $w$
is neither in $C_j$ nor in $D_i$, then accept $w$. If $w$ is in $D_i$, our aim is to spoil the reduction at $w$, if possible. There are now three cases to consider, depending on the size of $\varphi_i(w)$:

1. $|\varphi_i(w)| = |w| + 1$,
2. $|\varphi_i(w)| < |w|$, 
3. $|\varphi_i(w)| = |w|$ or $|\varphi_i(w)| > |w| + 1$.

We use these three cases to construct $A$ as follows:

In case 3, reject $w$.

In case 2, guess a string $x$ such that $|x| = |w|$, and accept $w$ iff $x < w$ and $\varphi_i(x) = \varphi_i(w)$.

In case 1, acceptance of $w$ depends on the value of $\varphi_i(w)$. Let $\varphi_i(w) = 0y$ or $1y$, where $|y| = |w|$. If case 2 or case 3 applies to $y$, then accept $w$. In the remaining case accept $w$ iff $w \leq y$. (Note that this includes the case $w = y$; i.e., it includes the case in which $\varphi_i(w) \in I$.)

This finishes the construction of $A$.

**Claim 4.1.** The set $A$ is NP-complete.

Clearly $A$ is in NP by the way it is defined. To see that $A$ is complete, we reduce SAT to $A$ through a many-one reduction, thus proving that both $A$ and $B$ are Karp-complete. Let $F$ be a formula such that $|F| = n$. We compute the first value $n' > n$ such that $n' = g(i)$, for some $i$. Then, we pad $F$ with a string in $0^*1$ to get a string $F'$ of size $n'$. By direct construction we have that $F \in SAT \iff F' \in A$.

**Claim 4.2.** For every $i$, $\varphi_i$ is not a reduction from $A$ to $B$.

We show that the reduction $\varphi_i$ is spoiled for at least one value in $D_i$. For any string $w \in D_i$, we first assume that case 3 applies. In this case we have that $w \notin A$. We also have that $\varphi_i(w) - 0y$ or $1y$, and we now show that $y \notin C$ and $y \notin D$. Since $w \in D_i$, $|w| = g(i) + 2$. Since case 3 applies, $|\varphi_i(w)| = |w|$ or $|\varphi_i(w)| > |w| + 1$. If $|\varphi_i(w)| = |w|$, then $|y| = |w| - 1 = g(i) + 1$, so $y \notin C$ and $y \notin D$. If $|\varphi_i(w)| > |w| + 1$, then $|y| \geq |w| + 1$ so $|y| \geq g(i) + 3$. Since the computation of the function $\varphi_i$ is limited in time by $|w|^k = (g(i) + 2)^k = g(i + 1) - 1$, we have $|y| < g(i + 1)$. Again, $y \notin C$ and $y \notin D$.

Because all strings in $\Sigma^* - C - D$ are in $A$, $y \in A$. Thus by the definition of $B$ and the relation between $y$ and $\varphi_i(w)$, $\varphi_i(w) \in B$. Thus if there is a $w \in D_i$ for which case 3 holds, then the reduction $\varphi_i$ is spoiled at $w$. So, for the following, we may now assume that case 3 does not arise for any string in $D_i$.

Next suppose that for two different strings $w$ and $w'$ in $D_i$, case 2 applies and $\varphi_i(w) = \varphi_i(w')$. Without loss of generality, assume $w < w'$ and $w$ is the smallest
value in $D_i$ such that $\varphi_i(w) = \varphi_i(w')$. We have that $w \notin A$ because the guess in the
procedure for $A$ will fail and $w$ will be rejected. We also have that $w' \in A$ because
the guess in the procedure will succeed ($w'$ will be found) and $w'$ will be accepted.
So, no matter whether $\varphi_i(w) = \varphi_i(w')$ is in $B$ or in $\bar{B}$, the reduction will be spoiled,
either at $w$ or at $w'$. So, we may now assume that case 2 arises only in a one-to-one
fashion, and in consequence, for each string $w$ for which case 2 arises, we will have
that $w \notin A$.

Because we now need consider case 2 only when it arises in a one-to-one fashion,
and because there are more strings in $D_i$ than there are strings of smaller size, we
are now guaranteed that case 1 must happen for at least one string in $D_i$.

So now suppose that $w$ is any string in $D_i$ for which case 1 applies. Let
$\varphi_i(w) = z = 0y$ or $1y$. We have that $|y| = |w|$ since case 1 applies to $w$. We observe
first that we can assume that $y \neq w$, since if $y = w$ then $w$ is placed into $A$ but $\varphi_i(w)$
is in $I$, which keeps $\varphi_i(w)$ out of $B$, spoiling the reduction. Also, as mentioned
above, if case 3 had applied to $y$ or if case 2 applies to $y$ but not in a one-to-one
fashion, the reduction is already spoiled somewhere. So we may assume that if
case 2 applies to $y$ then it applies in a one-to-one fashion. In this case the preceding
analysis shows that $y \notin A$. But this in turn implies that neither $0y$ nor $1y$ is in $B$,
forcing $\varphi_i(w) \notin B$. However, since case 2 applied to $y$, we put $w$ into $A$, spoiling $\varphi_i$
at $w$.

So now we may assume that case 1 also applies to $y$, and since we've already
disposed of the case $w = y$, we may assume that for all $w$ in $A$ for which case 1
occurs, the corresponding $y$ with $\varphi_i(w)$ equal to $0y$ or $1y$ also has case 1 applying
to $y$ and $w \neq y$. It follows that in this final case the mapping which takes $w$ to the
 corresponding $y$ such that $\varphi_i(w) = 0y$ or $= 1y$ must be stable among all of the $y$'s
of length $|w|$ to which case 1 applies, and that this mapping leaves no elements
fixed. If we let $w$ be the smallest member of $D_i$ for which case 1 applies and for
which there is a string $w' > w$ such that $\varphi(w')$ is either $0w$ or $1w$, we have that case 1
applies both to $w$ and to $w'$. We know that in this case $w \in A$, forcing both $0w$ and
$1w$ into $B$, and also forcing $w' \notin A$. This spoils $\varphi_i$ at $w'$ since $\varphi_i(w') = 0w$ or $1w$.

Osamu Watanabe has pointed out to us that by using the methods of his proof
that Karp reducibility and Cook reducibility yield different completeness notions
for $E$, [Wat87a], and for $NE$, [Wat87b], one can obtain a weaker version of our
Theorem 4, where the set $B$ is only 2-disjunctive complete instead of many-one
complete. We would like to emphasize that normal diagonalization techniques (see,
e.g., [KLD87, KMR86, Wat87a, Wat87b]) seem always to be limited to 2-disj
completeness because, when making sure that an element is not in the set, one
needs to put some other known element into the set to keep it complete. Thus the
set remains complete under 2-disj reductions since one can ask for the "or" of the
two elements. Our technique seems original in its ability to keep both constructed
sets many-one complete.
3. Disjunctive Reductions and Honest Many-One Reductions

Returning to our ultimate goal of showing that Cook and Karp reducibilities differ on sets in NP, we now consider circumstances under which we can prove a result similar to Theorem 4, but force a condition that requires some large reasonable class of many-one reductions of \( A \) to \( B \) to always require a non-polynomial amount of time. The following result, which is here a corollary to both Theorem 4 and to Theorem 7, was announced without proof in [You83]:

**Corollary 5.** For all \( k < 1 \), there exist sets \( A \) and \( B \) which are Karp-complete for NP and such that \( A \leq_{\text{z-disj}}^p B \) in linear time, but \( A \leq_m^p B \) requires at least \( n^k \) time if the reduction is \( n^k \) honest.

Honesty here means that the computation time is polynomially bounded by the size of the output and that the size of the outputs, although they can be arbitrarily large, cannot be smaller than the size of the input by more than a polynomial factor.

Returning to our main program, we will next see that Corollary 5 can be strengthened so that the many-one reductions which are spoiled include arbitrarily difficult reductions, including all polynomial time reductions, provided we limit the many-one reductions which are considered to those which are polynomially honest.

Most reductions one uses in practice are honest and, beginning with [MR72 and Mac75], the notion of honesty has had a persistent history in the study of complexity theory. A computation is thought of a dishonest if it takes a very long time to produce a very small result. By way of contrast, a computation is thought of a honestly difficult if the difficulty of the computation is merely caused by, or can reasonably be predicted from, the size of the output being produced. Largely for technical reasons, one also wants to regard computations which are easy simply because they fail to read their inputs as dishonest. These considerations lead to the following definition.

**Definition 6.** A many-one reduction \( f \) is \( T(n) \) honest if

1. for all \( x, f(x) \) can be computed in time \( T(|f(x)|) \) and
2. for all \( x, |x| \leq T(|f(x)|) \).

Similar definitions of honesty have been used for truth-table and Turing reductions in [You83], and will be used in an informal way below. We refer the reader to [Hom87, HL87] for a formal definition of honest Turing reductions. It is impor-

---

1 If, unlike in our conditions for Theorem 7, the many-one reductions under consideration are all computable in polynomial time, then condition (a) is superfluous. Thus when one is dealing only with polynomially computable functions, condition (b) is sometimes taken as the sole defining characteristic of polynomially honest functions. (We might also observe that in practice one seldom needs the full force of condition (b), but rather one needs only that \( |x| \) is bounded by some polynomially honest, unbounded, (and perhaps monotone), function from \( |f(x)| \).)
tant to note that in practice it usually seems possible to make most standard polynomial reductions polynomially honest.

The following theorem gives NP-complete sets which are easily reduced via honest Cook reductions but which have no \( n^k \)-honest computable reductions, including none which can be computed in polynomial time, no matter how large the polynomial. The proof will be a priority argument.\(^2\)

**Theorem 7.** For all \( k \leq 1 \), there exist sets \( A \) and \( B \) which are Karp-complete for NP and such that \( A \leq_2 \text{disj} B \) in linear time via a linearly honest reduction, but \( A \leq_m B \) only through reductions that are not \( n^k \)-honest.

**Proof.** We begin by coding any NP-complete set into \( C \), a subset of all strings of even length; in doing this encoding we require that if the strings \( 0y \) and \( 1y \) have even length then \( 0y \in C \iff 1y \in C \). Details of this are straightforward, and are left to the reader. Strings of the form \( 0y \) and \( 1y \) will be called *pairs*, and each of these strings will be called the *partner* of the other. We take \( A \) to be \( C \cup \text{ODDS} \), where \text{ODDS} is the set of all strings of odd length. Clearly, by appropriate choice of \( C \), we can guarantee that the set \( A \) will be Karp-complete for NP.

We will define a polynomially decidable set \( P_0 \) which for every \( y \) has the property that exactly one of \( 0y \) and \( 1y \) belong to \( P_0 \). \( B \) will then be \( A \cap P_0 \). Thus, just as in the proof of Theorem 4, for all strings \( y \), \( y \in A \) iff \( y \) or the partner of \( y \) is in \( B \). Clearly this shows that \( A \leq^P \text{disj} B \) in linear time, and that with any reasonable definition of honesty, the reduction is linearly honest. Equally clearly, \( B \in \text{NP} \) from the definition, and \( A \leq^P \text{disj} B \) in polynomial time since \( P_0 \) will be polynomially decidable. This will guarantee that both \( A \) and \( B \) are Karp-complete.

We now let \( \phi_0, \phi_1, \phi_2, \ldots \) be an enumeration of all \( n^k \) honest functions. Note that such an enumeration, to be effective, must contain some *partial* functions. Basically, to obtain this enumeration, one starts enumerating all partial recursive functions, but when an output appears, if the time required to get the output is not \( n^k \) bounded from the size of the output or if the output is too small compared with the size

---

\(^2\) The proof envisioned for proving Corollary 5 in [You83] was also by a priority argument. While Corollary 5 is in fact fairly directly proven without the use of priority methods, its generalization to Theorem 7 does seem to require a simple application of this recursion theoretic method. [You83] was a survey, intended to demonstrate the use of a fairly wide variety of such methods, including priority methods, in studying that were there called “structural” properties of sets in NP. The emphasis of that paper was to point out the use of such methods in proving *nonoracle* results about NP and other concrete complexity classes. Other structural methods described in [You83] included use of the recursion theorem, use of creative sets, and use of cylinders and immune sets in NP. These other uses of structural methods in NP described in [You83] have now all been reported in [JY85] and in [MY85]. A very elegant use of priority methods may be found in [KMR86], where priority methods are used to construct a polynomial many-one degree in exponential time which collapses to a single polynomial time isomorphism type. The referee suggests that the reader wishing to learn more about priority methods in a recursion theoretic context consult [Soa87]. A classical source would be [Rog67]. The use of priority methods in concrete complexity theory is not common, except perhaps for oracle constructions.
of the input, one transforms the output into a large enough value to achieve the required polynomial bound.

The set \( P_0 \) will be defined using a priority argument that keeps every total \( \phi_i \) from being a many-one reduction of \( A \) to \( A \cap P_0 \) \((= B)\). The idea in this proof is to associate an element of odd length, \( y_i \), with every function \( \phi_i \). Since \( y_i \) has odd length, \( y_i \in A \). The hope therefore is to "spoil" \( \phi_i \) by keeping \( \phi_i(y_i) \) out of \( B = (A \cap P_0) \). The \( n^k \) honesty of \( \phi_i \) guarantees that, if \( \phi_i(y_i) \) cannot be too much bigger than \( \phi_i(y_i) \) and that \( \phi_i(y_i) \) can be computed in time polynomially bounded from \( |\phi_i(y_i)| \). This guarantees that for any \( z \), if we are trying to decide whether to put \( z \) into \( P_0 \) (and hence potentially into \( B \)), we can make the decision in polynomial time provided we do not have too many \( i \)'s such that \( \phi_i(y_i) \) might equal \( z \) and provided we can locate the appropriate \( y_i \)'s. Conflicts are caused, for example, when one value of \( \phi_i(y_i) \) needs to be kept out of \( B \) in order to spoil \( \phi_i \) but the partner of \( \phi_i(y_i) \) is the value of some \( \phi_j(y_j) \) which also must be kept out of \( B \) in order to spoil \( \phi_j \). This can contradict the requirement that for every \( z \) either \( z \) or the partner of \( z \) must be placed into \( P_0 \). These conflicts, and similar conflicts, are always resolved in favor of the function with smaller index, forcing the other function to find a new value of \( y \) for the attempted spoiling. The bookkeeping of finding these new values is easily accomplished via a straightforward movable markers, or "priority" argument.

We now give details of the construction: For any string \( z \) of length \( 2i \) or \( 2i + 1 \), the membership of \( z \) in \( P_0 \) is determined by running all stages \( j \) for \( j \leq i \) of a priority construction: At stage \( i \) of the construction, we say that a string \( y \) is free if neither \( y \) nor \( y \)'s partner has a marker beside it, if \( y \) has odd length, and if \( |y| \geq (2i + 1)^k \).

We begin stage \( i \) by placing a marker, designated by \( \Box_j \), beside some free \( y \), called \( y_j \), for each \( j \leq i \) which does not already have a marker \( \Box_j \) beside some \( y_j \). (Note that if \( \phi_i \) is \( n^k \)-honest, this guarantees that \( |\phi_i(y_j)| > 2i \). Note also that since the number of strings of length \( (2i + 1)^k \) that are not free at the beginning of stage \( i \) is at most \( 2i \), and since we need to mark at most \( i + 1 \) free strings, we can find the necessary free strings in time that is polynomial in \( i \) by running the first \( i \) stages of this construction to see which strings of length \( (2i + 1)^k \) have already been marked.

Next, for each \( j \leq i \) we do \( (2i + 1)^k \) steps in the computation of each \( \phi_j(y_j) \). (When the construction is completely defined, it will be clear that many of these computations are already known and need not really be repeated here.)

We will now be concerned only with those \( j \leq i \) for which \( \phi_j(y_j) \) has length \( 2i \) or \( 2i + 1 \). Such a \( j \) is said to be in conflict with a similar \( j' \neq j \) if either \( \phi_j(y_j) \) or the partner of \( \phi_j(y_j) \) is either \( y_j \) or \( \phi_j(y_j) \) or the partner of one of these latter two strings. For any such pair \( j < j' \leq i \) for which \( j \) is in conflict with \( j' \) or \( j' \) is in conflict with \( j \), resolve the conflict by removing the marker \( \Box_j \) from \( y_j \), and no longer considering \( \phi_j \) during stage \( i \). (Note that this will cause \( \Box_j \) to be reintroduced next to some new \( y_j \) at the beginning of stage \( i + 1 \).)

Now for the remaining \( \phi_j \)'s with \( |\phi_j(y_j)| \) either \( 2i \) or \( 2i + 1 \), place the partner of \( \phi_j(y_j) \) into \( P_0 \). Since \( y_j \) has odd length, \( y_j \) is automatically placed into \( A \), and placing the partner of \( \phi_j(y_j) \) into \( P_0 \) keeps \( \phi_j(y_j) \in P_0 \), and hence \( \phi_j(y_j) \notin B \). Thus
we say that this spoils $\phi_j$, since it keeps $\phi_j$ from reducing $A$ to $B$. For any string $y$ of length $2i$ or $2i+1$ for which neither $y$ nor the partner of $y$ has been placed into $P_0$ by this process, place the smaller of $y$ and the partner of $y$ into $P_0$.

It is clear that this procedure gives a polynomial time algorithm for deciding membership in $P_0$, and that for each $y$, exactly one of $y$ and the partner of $y$ is placed into $P_0$. Thus to complete the proof, we must show that if $\phi_j$ is total then $\phi_j$ does not witness that $A \leq_m B$.

Now it is obvious that if $\phi_j$ is spoiled at some stage $i$ then $\phi_j$ cannot many-one reduce $A$ to $B$ since $y_j \in A$ but $\phi_j(y_j) \notin B$. Next we observe that if $\square_j$ is introduced beside $y_j$ during stage $i$, then unless we move $\square_j$ in the process of resolving a conflict with a smaller $j'$, if $\phi_j(y_j)$ is defined then we must spoil $\phi_j$ at stage $i$ or some following stage since, by choice of $y_j$ and the honesty of $\phi_j$, $|\phi_j(y_j)| \geq 2i$. But since resolving such conflicts can only involve a smaller $j'$, and since in resolving such a conflict $\phi_j$ gets spoiled for this smaller $j'$, for any $j$ such conflicts can force a movement of $\square_j$ only finitely often.

Thus each $\square_j$ must come eventually to rest, so $\phi_j$ must eventually be spoiled if $\phi_j$ is total.

4. RELATED RESULTS AND OPEN PROBLEMS

In Theorem 4, we were able to diagonalize over all $n^k$ time many-one reductions. In Theorem 7, we were able to diagonalize over all $n^k$ honest many-one reductions. Notice that an $n^k$ honest reduction need not be computable in time $n^k$ and vice versa. Thus, we can introduce a reduction which is more general than both reductions. We say that a function is $T(n)$ largely-honest if for all $x$, $f(x)$ can be computed in time $\max(T(|x|), T(|f(x)|))$. Notice that any $T(n)$ time computable or $T(n)$ honest function is $T(n)$ largely-honest. So diagonalizing against $n^k$ largely-honest functions actually diagonalizes against $n^k$ time functions, $n^k$ honest functions, and more.

One might conjecture that the combination of the techniques used in Theorems 4 and 7 would allow diagonalization over all $n^k$ largely honest many-one reductions. But we have been unable to prove this for complete sets in NP, even if we only require that the sets $A$ and $B$ be Cook-complete. A partial result in this direction can be seen in [Lon90], where DP-complete sets ([PaP84]) are considered instead of NP-complete sets.

It is also natural to ask about results similar to Theorems 4 and 7 in NP, and in particular to ask about the relation between other types of reductions. We can formalize such a discussion for reductions having a polynomial time bound by

**Definition 8.** We say that a reduction $R_1$ is faster than a reduction $R_2$ for sets that are $R_1$-complete for a class $\mathcal{C}$ if for all $k$, there exist sets $A$ and $B$ that are $R_3$-complete for $\mathcal{C}$ such that $AR_1B$ in linear time, but $AR_2B$ requires $n^k$ time.
In this notation, Theorem 4 says that $\leq_{2\text{-disj}}^p$ reductions are faster than $\leq_{m}^p$ reductions on sets that are Karp-complete for NP. We may ask about other types of reduction. For example, we can compare polynomial many-one and polynomial one-one, reducibility:

**Theorem 9.** Many-one ($\leq_{m}^p$) reductions are faster than one-one ($\leq_{1}^p$) reductions for sets that are Karp-complete for NP.

**Proof.** We use the same setting as in the proof of Theorem 4. But the technique here is simpler. We just need to make $A$ much more dense than $B$ to spoil any one-one $n^k$ reduction.

Let $g(1) = 0$ and $g(i+1) = (g(i) + 2)^k + 1$. As before, let the sets $C_i = \{w : |w| = g(i)\}$ and the sets $D_i = \{w : |w| = g(i) + 2\}$. Let $C = \bigcup_i C_i$ and $D = \bigcup_i D_i$.

Let $A = (C \cap \text{SAT}) \cup (D \cap 1\Sigma^*)$ and $B = (C \cap \text{SAT}) \cup (D \cap 1\Sigma^*)$. Both $A$ and $B$ are Karp-complete because SAT has been coded in the $C_i$ regions, as we did in Theorem 4. Furthermore, $A \leq_{m}^p B$ through the following function $f$. If $w$ starts with a 0, then $f(w) = w$. Otherwise, $f(w) = 1^{w_1}$.

To show that $A$ cannot reduce to $B$ through a one-one $n^k$ function, notice that if $n = g(i) + 2$ for some $i > 1$, there are $2^{n-1}$ strings of size $n$ in $A$. There is one string of size $n$ in $B$, no string of size $n-1$, and fewer than $2^{n-1} - 1$ strings of size $\leq n - 2$. Moreover, $B$ has no strings of size $n + 1 \cdots n^k$. So, if a reduction is one-one, by the pigeon hole principle, it must map a string of $A$ onto a string of size $> n^k$. This cannot be done in time $n^k$.

It would be interesting to know whether $\leq_{(k + 1)\text{-disj}}^p$ reductions are faster than $\leq_{k\text{-disj}}^p$ reductions on NP-complete sets. Solving this question seems technically very difficult. It can be shown (see [Lon90]) that $\leq_{(k + 1)\text{-disj}}^p$ reductions are faster than $\leq_{k\text{-disj}}^p$ reductions for sets Karp-complete for NP. Any intermediate result, including proving that $(k + 1)$-conjunctive reductions are faster than $k$-conjunctive reductions might be interesting.

It would also be interesting to investigate this question for any other class below $E$, including the class $P$ of sets decidable in polynomial time. The techniques of this paper can be applied directly for many reductions in $P$, although in some cases simpler techniques suffice. For example, from the following theorem it follows that $\leq_{1\text{-rel}}^p$ reductions are faster than $\leq_{m}^p$ reductions for complete sets in $P$. Bearing in mind that the reduction $x \in A \iff x \notin \overline{A}$ is one of the simplest possible truth-table reductions (reducing $A$ to $\overline{A}$), the following straightforward result gives a particularly strong separation for many-one and simple truth-table reductions in $P$:

**Theorem 10.** There is a set $A$ in $P$ such that $A \leq_{m}^p \overline{A}$ requires time at least $n^k$. The set $A$ can have any reasonable density. For example, if $c(n)$ is a census function computable in time polynomial in $n$, then $A$ can be taken to have density $c(n) + 1$. Furthermore, $A$ can be taken to be complete under, e.g., logspace reducibility, if we allow the density to differ from $c(n)$ for a polynomial fraction of the integers $\leq n$.  

571/41/3-9
Proof. As before, let \( g(1) = 0 \) and \( g(i + 1) = (g(i) + 2)^k + 1 \). Let \( \varphi_1, \varphi_2, \varphi_3, \ldots \) be an enumeration of all \( n^k \) time bounded functions. We use the strings of the form \( 1g(i)+2 \) to diagonalize over \( \varphi_i \).

For any binary string \( w \) of length \( n \) not in \( 1^* \), put \( w \) in \( A \) if and only if \( w \notin c(n) \). This will ensure that \( A \) has density \( c(n) \pm 1 \). Since \( c(n) \) is computable in polynomial time, \( A - 1^* \in P \).

For strings in \( 1^* \),

\[
1^n \in A \iff n = g(i) + 2 \quad \text{and} \quad [\varphi_i(1^n) = 1^n \text{ or } \varphi_i(1^n) \in A].
\]

Because of our choice of \( g \) and because computing \( \varphi_i(1^n) \) cannot take more than \( n^k \) time, if \( \varphi_j(1^{g(i)+2}) = 1^{g(j)+2} \) for some \( j \neq i \), then \( g(j) < g(i) \). This means that the definition of \( A \) is not circular, and membership of \( 1^n \) in \( A \) can be computed in polynomial time using recursive calls.

To make \( A \) complete for logspace reducibilities, encode a complete language over the strings of size \( g(i) \), as we have done before. This will alter the density only for strings of size \( g(i) \). But integers of the form \( g(i) \) account for only a polynomial fraction of the integers \( \leq n \).

Aside from proving that Cook reducibility and Karp reducibility actually differ on NP, perhaps the most interesting problem would be to extend Theorem 7 by giving an example of two NP-complete sets which admit easy Cook reductions but which do not admit any polynomially honest Karp reduction. This would seem to be about as close as one could hope to come to proving that Cook reductions truly are a more general class of polynomial reductions on NP than Karp reductions without actually proving that Cook reductions are more general, and thus proving that \( P \neq NP \).

The reader should be warned that proving this is likely to be very difficult: Proving the existence of two sets which are complete for NP under Karp reductions but which have no polynomially honest Karp reductions from one of the sets to the other would prove that not all NP-complete sets are polynomially isomorphic. This is currently one of the most intensely studied problems in structural complexity theory. For detailed surveys of this isomorphism problem, the reader should consult [KMR90 and You90].

REFERENCES


