On the complexity of proving functions*

by ANDY N. C. KANG

University of California
Berkeley, California

INTRODUCTION

Let \( f \) be a recursive function. We shall be interested in the following question: given \( x \) and \( y \), how difficult is it to decide whether \( f(x) = y \) or \( f(x) \neq y \)? Since the problem of deciding \( f(x) = y \) or \( f(x) \neq y \) is the same problem as that of computing the characteristic function \( C_f \) of the graph of \( f \), we can study the above question by looking into the complexity of computing \( C_f \).

We say that algorithm \( j \) serves to prove (or disprove) \( f(x) = y \) if \( c_P/j \in \mathcal{R} \) and \( c_P/j \) computes the characteristic function, \( C_f \), of the graph of \( f \) and \( c_P/j(x, y) = 1 \) if and only if \( f(x) = y \).

We shall show that the complexity of computing functions and the complexity of proving them are approximately equal modulo some recursive function \( h \).

Let \( g \) and \( f \) be recursive functions. We say that \( f \) is difficult to prove almost everywhere (infinitely often) modulo \( g \) if every algorithm which computes \( C_f \) takes more than \( g(x, y) \) steps to output a 1 for almost all \( x \) and all \( y \). We say that \( f \) is difficult to disprove almost everywhere (infinitely often) modulo \( g \) if every algorithm which computes \( C_f \) takes more than \( g(x, y) \) steps to output a 0 for almost all \( x \) and at least one \( y \).

Based on these definitions, we prove the following results: (1) A function is difficult to prove infinitely often if and only if it is difficult to disprove infinitely often. (2) There exists a function which is difficult to prove almost everywhere, but, surprisingly, it is not difficult to disprove almost everywhere. (3) There exists a function which is difficult to disprove almost everywhere, but it is not difficult to prove almost everywhere.

Before proceeding with our study, we give some preliminaries. Let \( R_n \) be the set of recursive functions of \( n \) variables. Let \( \phi^{(2)}, \phi^{(3)}, \ldots \) be an acceptable Gödel numbering of all the partial recursive functions of two variables [Ref. 4]. A partial recursive function \( \Phi_i^{(2)} \), the "step counting function," is associated with each \( \phi^{(2)} \). The set of partial recursive functions \( \{ \Phi_i^{(2)} \}_{i \geq 0} \) is arbitrary save for two axioms:

1. \( \Phi_i^{(2)}(x, y) \) converges if and only if \( \phi^{(2)}(x, y) \) converges, and
2. the function \( M(i, x, y, z) = \begin{cases} 1 & \text{if } \Phi_i^{(2)}(x, y) = z \\ 0 & \text{otherwise} \end{cases} \) is recursive.

Intuitively \( \Phi_i^{(2)}(x, y) \) represents the amount of time (or space) used by program \( i \) when it finally halts after receiving inputs \( x \) and \( y \).

THE COMPLEXITY OF COMPUTING VERSUS PROVING FUNCTIONS

Definition 1:

Let \( f \in \mathcal{R} \). \( \phi^{(2)} \) is a characteristic function for the graph of \( f \) if

\[
\phi^{(2)}(x, y) = \begin{cases} 1 & \text{if } f(x) = y \\ 0 & \text{if } f(x) \neq y \end{cases}
\]

We write \( \phi^{(2)} = C_f \), where \( C_f \) is the characteristic function for the graph of \( f \).

Definition 2:

\( \Phi^{(2)}(x, y) \) is the complexity of algorithm \( k \) of disproving \( f(x) = y \) if \( \phi^{(2)} = C_f \) and \( \phi^{(2)}(x, y) = 0 \).

Definition 3:

\( \Phi^{(2)}(x, y) \) is the complexity of algorithm \( k \) of proving \( f(x) = y \) if \( \phi^{(2)} = C_f \) and \( \phi^{(2)}(x, y) = 1 \).

The following lemma asserts that the complexity of computing \( f(x) \) and the complexity of proving \( f(x) = y \) are approximately equal modulo some recursive function \( h \).

* Research sponsored by the National Science Foundation, Grant GJ-708.
Lemma 1:

Let $\Phi$ be any complexity measure. There exist three functions $h \in R_3$, $\gamma \in R_3$, $\sigma \in R_3$ such that for any given $f \in R_3$:

(a) If $\phi_1 = f$, then $\Phi_{\gamma}(x) = C_f$ and

\[ \forall x \forall y[f(x) = y \rightarrow \Phi_{\gamma}(x, y) \leq h(x, y, \Phi_1(x))]. \]

(b) If $\phi_1 = C_f$, then $\phi_1 = f$ and

\[ \forall x \forall y[f(x) = y \rightarrow \Phi_{\gamma}(x, y) \leq h(x, y, \Phi_2(x, y))]. \]

Proof:

(a) Any algorithm to compute $f$ can obviously be used to obtain an algorithm for proving $f$. The complexity of the algorithm for proving $f$ is then bounded by the complexity of the algorithm for computing $f$. Formally: let $\gamma$ be a recursive function such that:

\[ \Phi_{\gamma}(x, y) = \begin{cases} 1 & \text{if } \phi_1(x) \text{ converges and } \phi_2(x) = y \\ 0 & \text{if } \phi_1(x) \text{ converges and } \phi_1(x) \neq y \\ \text{diverge otherwise} & \end{cases} \]

If $\phi_1 = f$, then $\Phi_{\gamma}(x)$ is recursive since $\phi_1$ always converges. Then $\Phi_{\gamma}(x, y) = 1 \Rightarrow \phi_1(x) = y$, i.e., $\Phi_{\gamma}(x) = C_f$.

Let

\[ p_1(i, x, y, z) = \begin{cases} \Phi_{\gamma}(i, x, y) & \text{if } \phi_1(x) = z \text{ and } \phi_1(x) = y \\ 0 & \text{otherwise} \end{cases} \]

$p_1$ is recursive, since $\lambda z [\Phi_1(x) = z]$ is a recursive predicate, and $\phi_1(x)$ convergent implies $\Phi_{\gamma}(x, y)$ converges.

Define $h_1(x, y, z) = \max \{p_1(i, x, y, z) \mid i \leq x \}$.

Then we have that for all $i$:

\[ \forall x \forall y[f(x) = y \rightarrow \Phi_{\gamma}(x, y) \leq h_1(x, y, \Phi_1(x))]. \]

If $\phi_1 = f$. Then

\[ \forall x \forall y[f(x) = y \rightarrow \Phi_{\gamma}(x, y) \leq h_1(x, y, \Phi_1(x))]. \]

(b) Given an oracle $\phi_2$ for the graph of $f$, a program can compute $f$ by asking questions about $\phi_2$. Because the graph of $f$ is single-valued, an affirmative answer from $\phi_2$ gives the value of $f$. Formally: let $\sigma$ be a recursive function such that:

$\phi_2(x)(y) = \text{"With input } x, \text{ compute } \phi_2(x, 0), \phi_2(x, 1), \ldots \text{ by dovetailing, until a } y \text{ appears such that } \phi_2(x, y) = 1; \text{ let the output be } y."

If $\phi_2$ computes $C_f$. Then $\phi_2$ is recursive and

\[ \exists x[x \in f_2(x) = y \rightarrow \Phi_{\gamma}(x, y) = 1 \Rightarrow f(x) = y] \]

whence $\phi_1 = f$. Let

\[ p_2(k, x, y, z) = \begin{cases} \Phi_2(x) & \text{if } \phi_2(x, y) = z \text{ and } \phi_2(x, y) = 1 \\ 0 & \text{otherwise} \end{cases} \]

$p_2$ is recursive, since $\lambda z [\phi_2(x, y) = z]$ is a recursive predicate, and $\phi_2(x, y) = 1$ implies $\Phi_2(x, y)$ converges.

Define $h_2(x, y, z) = \max \{p_2(k, x, y, z) \mid k \leq x \}$.

Then we have that for all $k$:

\[ \forall x \forall y[f(x) = y \rightarrow \Phi_{\gamma}(x, y) \leq h_2(x, y, \Phi_2(x, y))]. \]

If $\phi_2 = C_f$. Then

\[ \forall x \forall y[f(x) = y \rightarrow \Phi_{\gamma}(x, y) \leq h_2(x, y, \Phi_2(x, y))]. \]

$h$ in the lemma is defined by

\[ h(x, y, z) = \max \{h_1(x, y, z), h_2(x, y, z)\} \]

for all $x$, $y$, and $z$.

Notice that $h$ is independent of the choice of $f$.

Let $(\phi, \Phi)$ be the class of multitape Turing machines with step counting measure. Then the function $h$ of lemma 1 is roughly given by $h(x, y, z) = z^2$. This can be done by a straightforward construction of a Turing machine.

We will now show that a recursive $f$ exists for which there is no lower bound on the complexity for proving $f(x) = y$. This result basically follows from the speed-up theorem [Ref. 2] and lemma 1.

Theorem 1:

Let $\Phi$ be any complexity measure. For all $r \in R_3$ there is a $0-1$ valued recursive function $f \in R_1$ such that:

\[ \forall k [\Phi_{\gamma}(x) = C_f] \exists j [\phi_2(x) = C_f] \forall x \forall y[f(x) = y \rightarrow \Phi_{\gamma}(x, y) > r(\Phi_2(x, y))]. \]

Proof:

Without loss generality, we assume $r$ to be monotone increasing. Let $h \in R_3$ be any sufficiently large recursive function monotone increasing in the third variable ($h$ need only be large enough to satisfy lemma 1).

Define

\[ r'(x, z) = \max \{h(x, y, r(h(x, y, z))) \mid y = 0, 1\} \tag{1} \]

By the speed-up theorem, there exists a $0-1$ valued function $f \in R_1$ such that:

\[ \forall i [\phi_1 = f] \exists j [\phi_2 = f] \forall x [\Phi_2(x) > r'(x, \Phi_1(x))]. \tag{2} \]

Given $\phi_2 = C_f$, it follows from lemma 1 that there
exists an \( i \) such that:
\[
\phi_i = f
\]
and
\[
\forall x \forall y [f(x) = y \rightarrow \Phi_i(x) \leq h(x, y, \Phi_i^{(2)}(x, y))]
\]
(3)

Then by Eq. (2) there is an \( l \) such that \( \phi_l = f \) and
\[
\forall x [\Phi_i(x) > r'(x, \Phi_l(x))]
\]
(4)

Applying lemma 1 again, we get \( j = \gamma(l) \) such that:
\[
\phi_j^{(2)} = C_f
\]
and
\[
\forall x \forall y [f(x) = y \rightarrow \Phi_j^{(2)}(x, y) \leq h(x, y, \Phi_l(x))]
\]
(5)

Therefore we have:
\[
\forall x \forall y [f(x) = y \rightarrow h(x, y, \Phi_j^{(2)}(x, y))]
\]
\[
\geq \Phi_i(x) \quad \text{(by Eq. (3))}
\]
\[
> r'(x, \Phi_l(x)) \quad \text{(by Eq. (4))}
\]
\[
\geq h(x, y, r(h(x, y, \Phi_l(x))))) \quad \text{(by Eq. (1) and the fact that } y = f(x) \in \{0, 1\})
\]
\[
> h(x, y, r(\Phi_i^{(2)}(x, y)))) \quad \text{(by Eq. (5) and the fact that } r \text{ is monotone increasing.)}
\]

This implies that
\[
\forall x \forall y [f(x) = y \rightarrow \Phi_j^{(2)}(x, y) > r(\Phi_i^{(2)}(x, y))]
\]

since \( h \) is monotone increasing in the third variable.

Although the faster program exists, we cannot effectively get such a program. This fact follows from the theorem in Ref. (3).

THE COMPLEXITY OF PROVING VERSUS DISPROVING FUNCTIONS

Next we are going to investigate the complexity for proving and disproving functions. First we make a definition.

Definition 4:

Let \( g \in R_s \), \( f \in R_1 \). We say \( f \) is "difficult to prove almost everywhere (infinitely often) modulo \( g' \)" if every \( \Phi_i^{(2)} \) computing \( C_f \) has the property that for almost all \( x \) (infinitely many \( x \)) and all \( y, f(x) = y \) implies \( \Phi_i^{(2)}(x, y) > g(x, y) \).

Let \( f \in R_1 \). For each \( x \) there is only one \( y \) with the property that \( y = f(x) \). Hence it is ambiguous to say \( f \) is difficult or easy to disprove at \( x \), since the complexity of disproving \( f \) at \( x \) also depends on argument \( y \).

For example, Rabin's 0—1 valued recursive function \( f \) [Ref. 1] is difficult to compute, hence by lemma 1, \( f \) is difficult to prove. As to the complexity of disproving \( f \): if \( y \in \{0, 1\} \), then we know immediately that \( f(x) \neq y \). However, if \( y \in \{0, 1\} \), then \( f(x) \neq y \) is not easy to verify; it is easy to show that this must be the case, i.e., if \( y \in \{0, 1\} \), then to prove \( f(x) = y \) or to disprove \( f(x) = y \) is of the same difficulty.

Since functions, like Rabin’s 0—1 valued function, are generally treated as difficult, we have the following definition for the complexity of disproving functions.

Definition 5:

Let \( g \in R_s \), we say that \( f \) is "easy to disprove almost everywhere (infinitely often) modulo \( g' \)" if some \( \phi_i^{(2)} = C_f \) has the property that for almost all \( x \) (infinitely many \( x \)) and all \( y, f(x) \neq y \rightarrow \Phi_i^{(2)}(x, y) \leq g(x, y) \).

The complement of this definition is:

Definition 6:

Let \( g \in R_s \), we say that \( f \) is "difficult to disprove almost everywhere (infinitely often) modulo \( g' \)" if every \( \phi_i^{(2)} = C_f \) has the property that for almost all \( x \) (infinitely many \( x \)) and some \( y, f(x) \neq y \) and \( \Phi_i^{(2)}(x, y) > g(x, y) \).

The following theorem asserts that if function \( f \) is difficult to prove infinitely often, then it is also difficult to disprove infinitely often.

Theorem 2:

Let \( \Phi \) be any complexity measure. There is an \( h \in R_s \) such that for every step counting function \( g \in R_s \) and for all \( f \in R_1 \), if \( f \) is difficult to prove infinitely often modulo \( \lambda y [h(x, y, g(x, y))] \), then \( f \) is difficult to disprove infinitely often modulo \( g \).

Proof:

Assume to the contrary that there is a program for \( C_f \), which disproves \( f \) easily almost everywhere modulo \( g \), i.e.,
\[
\exists j \in \{0, 1\} \forall x \forall y [f(x) \neq y \rightarrow \Phi_j^{(2)}(x, y) \leq g(x, y)]
\]

(1)

Then we may use this program to construct a program which easily proves \( f \) almost everywhere. This relies on \( g \) being a step counting function. With inputs \( x \) and \( y \), our program simply computes \( g(x, y) \), then starts computing the given program \( \Phi_j^{(2)}(x, y) \). When

* If this were "almost all \( y \), then every \( f \) would be easy to disprove almost everywhere.
Let $f(x) = y$ immediately. Otherwise $f(x) = y$ takes more than $g(x, y)$ steps to converge. In that case we can prove or disprove $f(x) = y$ according to the value of $\Phi^{(2)}_{x, y}(x, y)$. Now we are going to give a formal proof of the above outline.

Let $\alpha$ be a recursive function such that:

$$
\Phi_{\alpha(i,j,l)}^{(2)}(x, y) = \begin{cases} 
1 & \text{if } x \leq l \text{ and } \Phi^{(2)}_{\alpha(i,j,l)}(x, y) \leq \Phi^{(2)}_{\alpha(i,j,l)}(x, y) \\
0 & \text{otherwise} 
\end{cases}
$$

**Lemma:**

If $g = \Phi^{(2)}_{\alpha(i)}$ and if $j = j_0$ is given by Eq. (1), then for some $l_0$, $\Phi_{\alpha(i,j_0,l_0)}^{(2)} = C_f$.

**Proof:**

Since $\Phi_{\alpha(i)}^{(2)}$ and $\Phi_{\alpha(j)}^{(2)}$ are total, $\Phi_{\alpha(i,j)}^{(2)}$ is total. By Eq. (1) there is a number $l_0$ such that for all $x$ and all $y$, if $x \geq l_0$ and $f(x) \neq y$, then $\Phi_{\alpha(i)}^{(2)}(x, y) \leq g(x, y)$. Therefore $\Phi_{\alpha(i,j,l_0)}^{(2)}$ computes $C_f$. End of lemma.

Now we shall construct the $h$ of the theorem. Define: $p(i, j, l, x, y, z)$

$$
p(i, j, l, x, y, z) = \begin{cases} 
0 & \text{if } x < l \\
\Phi_{\alpha(i,j,l)}^{(2)}(x, y) & \text{if } x \geq l \text{ and } \Phi^{(2)}_{\alpha(i,j,l)}(x, y) = z \\
0 & \text{otherwise} 
\end{cases}
$$

$p$ is recursive, for if $x \geq l$ and $\Phi^{(2)}_{\alpha(i,j,l)}(x, y)$ converges, then $\Phi_{\alpha(i,j,l)}^{(2)}(x, y)$ converges.

Let $h(x, y, z) = \max\{p(i, j, l, x, y, z) | \max\{i, j, l\} \leq z\}$. Thus we have that for all $i, j, l$:

$$
\forall x \forall y[h(x, y, \Phi^{(2)}_{\alpha(i,j,l)}(x, y)) \geq \Phi_{\alpha(i,j,l)}^{(2)}(x, y)].
$$

Hence, particularly for $i = i_0, j = j_0, l = l_0$:

$$
\forall x \forall y[h(x, y, g(x, y)) \geq \Phi_{\alpha(i_0,j_0,l_0)}^{(2)}(x, y)].
$$

since $\Phi_{\alpha^{(2)}(x, y)} = g(x, y)$.

This contradicts the hypothesis that $f$ is difficult to prove infinitely often modulo $\lambda x y[h(x, y, g(x, y))]$, since by the lemma: $\Phi_{\alpha(i_0,j_0,l_0)}^{(2)}$ is a program that computes $C_f$.

Conversely, we can show that if $f$ is difficult to disprove infinitely often, then it is also difficult to prove infinitely often.

**Theorem 3:**

Let $\Phi$ be any complexity measure. There is an $h \in R_2$ such that for every step counting function $g \in R_2$ and for all $f \in R_2$, if $f$ is difficult to disprove infinitely often modulo $\lambda x y[h(x, y, g(x, y))], then $f$ is difficult to prove infinitely often modulo $g$.

**Proof:**

The proof is similar to the proof of theorem 2. Notice that if there is a program $\Phi^{(2)}_{x}$ for $C_f$ which proves $f$ easily almost everywhere modulo $g$, then by definition this means that for almost all $x$ there is a $y$ such that $f(x) = y$ and $\Phi^{(2)}_{\alpha^{(2)}(x, y)} \leq g(x, y)$. Since the graph of $f$ is single-valued, if $x$ is large and $\Phi^{(2)}_{\alpha^{(2)}(x, y)} > g(x, y)$, then $f(x)$ cannot possibly be equal to $y$.

Theorems 2 and 3 show that a function is difficult to prove infinitely often if and only if it is difficult to disprove infinitely often.

Naturally, we may ask the question: if $f(x) = y$ is difficult to prove for almost all $x$, then does it follow that for each such $x$ there is a $y$ such that $f(x) = y$ is also difficult to disprove? This is false by the following theorem.

**Theorem 4:**

Let $\Phi$ be any complexity measure. There is a function $b \in R_2$ such that for all function $g \in R_2$, there exists a function $f \in R_2$ with two properties:

(a) $f$ is difficult to prove almost everywhere modulo $g$.

(b) $f$ is easy to disprove infinitely often modulo $b$.

**Proof:**

The theorem follows from the fact that even if a function $f$ has the property that there is a bound $b(x, y)$ on the number of steps to disprove $f(x) = y$ for infinitely many $x$ and all $y$, it may still be difficult to locate these $x$ for which $f(x) = y$ is easy to disprove.

Let $g \in R_2$ be given. Let $h \in R_3$ be any sufficiently large function monotone increasing in the third variable ($h$ need only be large enough to satisfy lemma 1). Let $\lambda x[a(x)]$ be a $0-1$ valued recursive function such that:

$$
\forall i(\phi_{i} = a) \forall x[\Phi_{i}(x) > h(x, 0, g(x, 0))].
$$

(1)

The existence of $\lambda x[a(x)]$ follows from Rabin's theorem [Ref. 1].

Let $\phi_{\alpha}$ be a fixed program for $\lambda x[a(x)]$. We define $f$ as follows: $f(x) = y$ With input $x$, first compute $f(y)$ for all $y < x$. (In the process of doing this, some finite set of $\phi_{\alpha}$ will be cancelled.) Second compute $\phi_{\alpha}(x)$.

Case 1: $\phi_{\alpha}(x) = 0$, let the output be 0.
Case 2: \( \Phi_i(x) = 1 \), look for \( j = n \) \( j \leq x \) and \( j \) is not cancelled and \( \Phi_j(x) \leq 1 + \max \{ h(x, y, g(x, y)) \mid y \in \{ \Phi_i(x) + 1, \Phi_i(x) + 2 \} \} \).

If no such \( j \) is found, let the output be \( \Phi_i(x) + 1 \). Otherwise, let the output be \( \Phi_i(x) + 2 \) if \( \Phi_i(x) = \Phi_i(x) + 1 \). Let the output be \( \Phi_i(x) + 1 \) if \( \Phi_i(x) \neq \Phi_i(x) + 1 \).

Then cancel \( \Phi_i \) from the standard list.""

Let us see why \( f \) works:

(b) We know that for any sufficiently difficult \( 0 \rightarrow 1 \)
function \( a, a(x) = 0 \) for infinitely many \( x \). From the construction of \( f \), \( a(x) = 0 \) implies \( f(x) = 0 \). We shall exhibit a program which has the property that it disproves \( f(x) = y \) quickly whenever \( a(x) = 0 \) and \( y \neq 0 \). In fact, we will define a recursive function \( b \), which does not depend on \( f \) such that \( b \) is an upper bound for the number of steps to disprove \( f(x) = y \) whenever \( a(x) = 0 \) and \( y \neq 0 \).

First we construct \( \Phi_{a(i, b)} \). We will show that \( \Phi_{a(i, b)} \) is the program that disproves \( f \) quickly when \( k \) is an index for \( f \) and \( i \) is \( i_o \), where \( \Phi_i \) is the program used in the construction of \( f \).

Let \( \sigma \) be a recursive function such that:

\[
\begin{align*}
\phi_{a(i, b)}(x, y) = & \begin{cases} 
1 & \text{if } y = 0 \text{ and } \phi_{a}(x) \text{ converges } \\
0 & \text{if } y \neq 0 \text{ and } \\
\phi_{a}(x) \text{ converges and } \phi_{a}(x) \neq y \\
1 & \text{if } y \neq 0 \text{ and } \phi_{a}(x) \leq y \text{ and } \phi_{a}(x) \neq 0 \text{ and } \\
\phi_{a}(x) \text{ converges and } \phi_{a}(x) = y \\
\text{diverge} & \text{otherwise}
\end{cases}
\end{align*}
\]

Lemma 4.1:

If \( k \) is an index for \( f \) and \( \phi_{a}(x) \) is the program used in the construction of \( f \), then \( \phi_{a(i, b)}(x) = C_f \).

Proof:

Observe that \( \phi_{a} \) is total implies that \( \phi_{a(i, b)}(x) = \) total.

Case 1: \( y = 0 \).

\[
\phi_{a(i, b)}(x, y) = \begin{cases} 
1 & \text{if } \phi_{a}(x) = 0 \\
0 & \text{if } \phi_{a}(x) \neq 0
\end{cases}
\]

\[
\phi_{a(i, b)}(x, 0) = C_f(x, 0) \text{ for all } x.
\]

Case 2: \( y \neq 0 \).

Subcase 1: \( \Phi_i(x) > y \). In this subcase,

\[
\phi_{a(i, b)}(x, y) = 0.
\]

Now look at the construction of \( f \). \( f(x) \in \{ 0, \Phi_i(x) + 1, \Phi_i(x) + 2 \} \). Therefore \( f(x) \) will either be 0 or will be greater than \( y \). Therefore \( f(x) \neq y \).

Subcase 2: \( \Phi_i(x) \leq y \) and \( \Phi_i(x) = 0 \). Again,

\[
\phi_{a(i, b)}(x, y) = 0.
\]

Look at the program for \( f \). \( \phi_{a}(x) = 0 \) implies \( f(x) = 0 \), therefore: \( f(x) \neq y \).

Subcase 3: \( \Phi_i(x) \leq y \) and \( \phi_{a}(x) \neq 0 \) and \( \phi_{a}(x) = y \) implies \( \phi_{a(i, b)}(x, y) = 1. \) \( \phi_{a(i, b)}(x, y) \) gives the correct answer. End of lemma 4.1.

Define \( b \) as follows:

\[
b(x, y) = \begin{cases} 
\max \{ \Phi_{a(i, b)}(x, y) \mid i \in A \text{ and } k \leq x \} & \text{if } y = 0 \text{ and } A \neq \emptyset, \text{ where} \\
A \mid i \leq x \text{ and } [\Phi_i(x) > y \text{ or } i, 0 \mid (x) = 0] & \text{otherwise}
\end{cases}
\]

\( b \) is recursive since one can effectively decide whether or not \( A = \emptyset \), and in case \( y = 0 \text{ and } A \neq \emptyset \), then by definition of \( \sigma, \phi_{a(i, b)}(x, y) = 0 \), so \( \Phi_{a(i, b)}(x, y) \) converges for all \( x \).

Lemma 4.2:

\[
\forall i \forall k \forall x \forall y [\phi_{a}(x) = 0 \text{ and } y = 0 \rightarrow \Phi_{a(i, b)}(x, y) \leq b(x, y)] \quad (2)
\]

Proof:

Let \( x \geq \max \{ i, k \} \). If \( \phi_{a}(x) = 0 \), then \( i \) will be in \( A \). If \( i \) is in \( A \) and \( y \neq 0 \), then \( b(x, y) \geq \Phi_{a(i, b)}(x, y) \). End of lemma 4.2.

Let \( i = i_o \), and let \( k \) be an index for \( f \). By lemma 4.1, \( \phi_{a(i, b)}(x) = C_f \). By lemma 4.2,

\[
\forall x \forall y [\phi_{a}(x) = 0 \text{ and } y = 0 \rightarrow \Phi_{a(i, b)}(x, y) \leq b(x, y)].
\]

By the definition of \( f \), we know that \( \phi_{a}(x) = 0 \Rightarrow f(x) = 0 \). Because \( \phi_{a}(x) \) is a sufficiently complex \( 0 \rightarrow 1 \) valued function, we know \( \phi_{a}(x) = 0 \) for infinitely many \( x \). Hence, we have:

\[
\exists x \forall y [f(x) = 0 \Rightarrow \Phi_{a(i, b)}(x, y) \leq b(x, y)].
\]

This finishes the proof of part (b).

The proof of part (a) is easier: Let \( \phi_{a} \) be any program which computes \( f \).

Lemma 4.3:

\[
\forall x [\phi_{a}(x) > h(x, f(x), g(x, f(x)))]
\]

Proof:

Case 1: \( f(x) = 0 \)

Case 2: \( f(x) = y \)

Case 3: \( f(x) \neq 0 \) and \( f(x) \neq y \)
Assume to the contrary that
\[ \exists \mathbf{x} [ \Phi_b(\mathbf{x}) \leq h(x, 0, g(x, 0)) ] \]

By the definition of \( f \), we know \( f(x) = 0 \) or \( \Phi_b(\mathbf{x}) = 0 \). By assumption, program \( \Phi_b \) computes \( f \) in less than \( h(x, 0, g(x, 0)) \) steps infinitely often when \( f(x) = 0 \). Hence, it computes \( \Phi_b(\mathbf{x}) \) in less than \( h(x, 0, g(x, 0)) \) steps infinitely often when \( \Phi_b(\mathbf{x}) = 0 \). This is a contradiction, since
\[ \forall \mathbf{i} (\Phi_i = \Phi_b) \forall \mathbf{x} [ \Phi_b(\mathbf{x}) > h(x, 0, g(x, 0))] \]
by Eq. (1).

Case 2: \( f(x) \neq 0 \).

Assume to the contrary that
\[ \exists \mathbf{x} [ \Phi_b(\mathbf{x}) \leq h(x, f(x), g(x, f(x))) ] \]

Notice that \( f(x) \neq 0 \) implies \( f(x) \in \{ \Phi_b(\mathbf{x}) + 1, \Phi_b(\mathbf{x}) + 2 \} \). Look at the definition of \( f \); there are infinitely many chances for \( k \) to be cancelled. Since at most \( k \) programs with index less than \( k \) may be cancelled, \( \Phi_b \) will eventually be cancelled and the value of \( f \) will be made different from \( \Phi_b \). This contradicts \( \Phi_b = f \).

End of lemma 4.3.

Let \( \Phi_b = C_x \). By lemma 1, there is a \( k \) such that \( \Phi_b = f \) and
\[ \forall \mathbf{x} \forall \mathbf{y} [ f(x) = y \Rightarrow \Phi_b(x) \leq h(x, y, \Phi_b(x, y))] \]  
Combining Eq. (3) and Eq. (4), we have:
\[ \forall \mathbf{x} \forall \mathbf{y} [ f(x) = y \Rightarrow h(x, y, \Phi_b(x, y))] \geq \Phi_b(x) > h(x, y, g(x, y))] \]

If follows that:
\[ \forall \mathbf{x} \forall \mathbf{y} [ f(x) = y \Rightarrow \Phi_b(x, y) > g(x, y)] \]

since \( h \) is monotone increasing in the third variable.

This finishes the proof of part (a).

Let \( (\phi, \Phi) \) be the class of multitape Turing machines with step counting measure. Then the function \( b \) of theorem 4 is given by \( b(x, y) = C \cdot y \), where \( C \) is a constant. To see this, we simply present a program to prove \( f(x) \neq y \) in \( C \cdot y \) steps when \( \phi_b(x) = 0 \) (note that \( \phi_b(x) = 0 \) occurs for infinitely many \( x \)), informally: given inputs \( x \) and \( y \), if \( y \neq 0 \), run \( y \) steps of \( \phi_b(x) \). If \( \phi_b(x) \) does not converge in \( y \) steps, we know immediately \( f(x) \neq y \) (since \( f(x) \in \{ 0, \Phi_b(x) + 1, \Phi_b(x) + 2 \} \) ).

Otherwise \( \phi_b(x) \) converges in \( y \) steps, and in case \( \phi_b(x) = 0 \), then we conclude that \( f(x) \neq y \) (since \( \phi_b(x) = 0 \)).

Parallel to the question asked before theorem 4, we may ask the question: if \( f \) is difficult to disprove almost everywhere, is it also difficult to prove almost everywhere? The answer is again negative by a similar construction of \( f \) as in theorem 4.

**Theorem 5:**
Let \( \Phi \) be any complexity measure. There is a function \( b \in R_3 \) such that for all function \( g \in R_2 \), there exists a function \( f \in R_1 \) with two properties:

1. \( f \) is difficult to disprove almost everywhere modulo \( g \).
2. \( f \) is easy to prove infinitely often modulo \( b \).

**Proof:**

The proof is similar to that of theorem 4. First we define a function \( h_1 \in R_3 \), the reason we define \( h_1 \) will be clear later.

Let \( \delta \) be a recursive function such that
\[ \phi_{h_1, \delta}^{(3)}(x, y) : \]

"With input \( x \) and \( y \), Compute \( \phi_0(x) \). If and when \( \phi_0(x) \) converges. Check following 3 cases.

Case 1: \( \psi(x) = 0 \); give output 1
Case 2: \( \psi(x) = 1 \); compute \( \phi^{(2)}(x, 0) \) and \( \phi^{(2)}(x, 1) \) simultaneously until one of them converges.

Subcase 1: \( \phi^{(2)}(x, 0) = 0 \) or \( \phi^{(2)}(x, 1) = 1 \); give output 1
Subcase 2: \( \phi^{(2)}(x, 0) = 1 \) or \( \phi^{(2)}(x, 1) = 0 \); give output 1
Subcase 3: \( \phi^{(2)}(x, 0) \) converges and \( \phi^{(2)}(x, 0) \notin \{ 0, 1 \} \) or \( \phi^{(2)}(x, 1) \) converges and \( \phi^{(2)}(x, 1) \notin \{ 0, 1 \} \); give output 0

Case 3: \( \phi(x) \) converges and \( \phi(x) \notin \{ 0, 1 \} \); give output 0."

Define:
\[ h_1(x, y, z) = \begin{cases} \max \{\phi_{h_1, \delta}^{(3)}(x, y) \mid (i, j) \in A \} & \text{if } A \neq \phi, \\ \phi(x) & \text{where} \\ A = \{ (i, j) \mid i \leq x \text{ and } j \leq z \} \text{ and } \phi(x) = w \text{ and } \phi^{(2)}(x, 0) \leq z \text{ or } \phi^{(2)}(x, 1) \leq z \} & \\ 0 & \text{otherwise} \end{cases} \]

\( h_1 \) is recursive, since we can effectively decide whether \( (i, j) \in A \) or not. And in case \( (i, j) \in A \), \( \phi_{h_1, \delta}^{(3)}(x, y) \) converges.

**Lemma 5.1:**
\[ \forall \mathbf{i} \forall \mathbf{j} \forall \mathbf{y} [ \phi_i(x) \text{ converges and } \phi^{(2)}(x, 0) \text{ converges or } \phi^{(2)}(x, 1) \text{ converges} ] \]
\[ \Rightarrow \phi_{h_1, \delta}^{(3)}(x, y) \leq h_1(x, y, \phi_i(x), \min \{ \phi^{(2)}(x, 0), \phi^{(2)}(x, 1) \}) \]
On the Complexity of Proving Functions

Proof:

Let \( x \geq \max \{ i,j \} \). Since either

\[
\Phi^{(3)}(x, 0) \leq \min \{ \Phi^{(2)}(x, 0), \Phi^{(2)}(x, 1) \}
\]

or

\[
\Phi^{(3)}(x, 1) \leq \min \{ \Phi^{(2)}(x, 0), \Phi^{(2)}(x, 1) \},
\]

by the definition of \( A \) we know that \( (i,j) \in A \). Therefore, for all \( x \geq \max \{ i,j \} \) and all \( y \)

\[
[\Phi^{(3)}(x, y) \leq \min \{ \Phi^{(2)}(x, 0), \Phi^{(2)}(x, 1) \}].
\]

End of lemma 5.1.

Now we are going to define \( f \). Let \( h_i \) be any sufficiently large function monotone increasing in the fourth variable (each need only be large enough to satisfy lemma 5.1 above). Let \( g \in R_4 \) be given. Let \( h \in R_4 \) be any sufficiently large function monotone increasing in the third variable (h need only be large enough to satisfy lemma 1). Let \( \lambda \alpha[x(a)] \) be a 0–1 valued recursive function such that

\[
\forall (\phi_{a(x)} \in \lambda \alpha[x(a)] \exists \forall (\phi_{a(x)}(x/y)) > 2 + \max \{ g(x, 0), g(x, 1) \}]. \quad (1)
\]

The existence of \( \lambda \alpha[x(a)] \) follows from Rabin’s theorem [Ref. 1].

Let \( \phi_{a(x)} \) be a fixed program for \( \lambda \alpha[x(a)] \). We define \( f \) as follows: \( f(x/y) = \text{"With input } x, \text{ first compute } f(y) \) for all \( y < x \). (In the process of doing this, some finite set of \( \phi_{a(x)} \) will be cancelled.) Second compute \( \phi_{a(x)}(x) \).

Case 1: \( \phi_{a(x)}(x) = 0 \), let the output be \( \Phi_{a(x)}(x) \).

Case 2: \( \phi_{a(x)}(x) = 1 \), look for \( j_0 = \mu j [j \leq x \text{ and } j \text{ is not cancelled and} \]

\[
\Phi_{a(x)}(x) \leq \max \{ h(x, y), \max \{ h(x, w, \Phi_{a(x)}(x),
\]

\[
\max \{ g(x, 0), g(x, 1) \} \} \}
\]

If no such \( j_0 \) is found, let the output be 0. Otherwise, let the output be \( 1 - \phi_{a(x)}(x) \). Then cancel \( \phi_{a(x)} \) from the standard list.17

Let us see why \( f \) works:

(a) We will show that for all sufficiently large \( x \), there exists some \( y \) such that \( f(x/y) \neq y \) and \( \Phi^{(3)}(x, y) > g(x, y) \) for every \( \Phi^{(2)} \) computing \( C_f \).

Lemma 5.2:

For all sufficiently large \( x \) such that \( \phi_{a(x)}(x) = 0 \), \( f(x/y) \neq y \) is difficult to disprove modulo \( g(x, y) \) for some \( y \), i.e.,

\[
\forall (\phi_{a(x)}(x) = 0) \forall y \forall (\phi_{a(x)}(x) = 0 - \exists y \forall (f(x) \neq y)
\]

and

\[
\Phi^{(3)}(x, y) > g(x, y). \]

Proof:

Assume to the contrary that: \( \phi_{a(x)}(x) = C_f \) and

\[
\exists x[\phi_{a(x)}(x) = 0
\]

and

\[
\forall y[f(x) \neq y \rightarrow \phi_{a(x)}(x, y) \leq g(x, y)] \quad (2)
\]

From the construction of \( f \), we know that \( \phi_{a(x)}(x) = 0 \rightarrow f(x) > 1 \). Therefore, if we can decide \( f(x) > 1 \) in less than \( \max \{ g(x, 0), g(x, 1) \} \) steps for infinitely many \( x \), then we can compute \( \phi_{a(x)}(x) \) in less than \( \max \{ g(x, 0), g(x, 1) \} \) steps for infinitely many \( x \). This will be a contradiction to Eq. (1). Since \( \phi_{a(x)}(x) \) computes \( C_f, \) a way to decide whether \( f(x) > 1 \) or not is to compute \( \phi_{a(x)}(x, 0) \) and \( \phi_{a(x)}(x, 1) \) simultaneously. For those \( x \) satisfying Eq. (2), the computation of \( \phi_{a(x)}(x, 0) \) and \( \phi_{a(x)}(x, 1) \) will converge in less than \( g(x, 0) \) and \( g(x, 1) \) steps respectively, since by the definition of \( f \) for those \( x \) with the property that \( \phi_{a(x)}(x) = 0 \), we know that \( f(x) \neq 0 \) and \( f(x) \neq 1 \). Therefore, we can decide \( f(x) > 1 \) in less than \( \max \{ g(x, 0), g(x, 1) \} \) steps for infinitely many \( x \). End of lemma 5.2.

Lemma 5.3:

For all sufficiently large \( x \) such that \( \phi_{a(x)}(x) = 1 \), \( f(x/y) = y \) is difficult to disprove modulo \( g(x, y) \) for some \( y \), i.e.,

\[
\forall (\phi_{a(x)}(x) = 1) \forall x[\phi_{a(x)}(x) = 0 \rightarrow \exists y \forall (f(x) = y)
\]

and

\[
\Phi_{a(x)}(x, y) > g(x, y) \]

Proof:

Claim:

For all sufficiently large \( x \) such that \( \phi_{a(x)}(x) = 1 \), \( f(x/y) = y \) is difficult to prove modulo

\[
\lambda \alpha [\max \{ h(x, w, \Phi_{a(x)}(x), \max \{ g(x, 0), g(x, 1) \} \}]
\]

for all \( y \), i.e.,

\[
\forall (\phi_{a(x)}(x) = 1) \forall x[\phi_{a(x)}(x) = 0 \rightarrow \exists y (f(x/y) = y - \Phi_{a(x)}(x, y) \rightarrow \max \{ h(x, w, \Phi_{a(x)}(x), \max \{ g(x, 0), g(x, 1) \} \}]
\]

Proof:

First we show that: for every \( \Phi_{a(x)} \) computing \( f(x/y) = y \) is difficult to disprove modulo

\[
\forall x[\phi_{a(x)}(x) = 1 \rightarrow [\Phi_{a(x)}(x) = 0 \rightarrow \exists y (f(x/y) = y - \Phi_{a(x)}(x, y) \rightarrow \max \{ h(x, w, \Phi_{a(x)}(x), \max \{ g(x, 0), g(x, 1) \} \}]
\]

(3)
Since otherwise there is a $\phi_\delta$ such that $\phi_\delta = f$ and
\[ \exists x [\phi_\delta(x) = 1] \]
and
\[ f_\delta(x) \leq \max \{ h(x, y, \max \{ h_1(x, w, \Phi_\delta(x), \ldots \}) \} \]

Look at the definition of $f$, then there are infinitely many chances for $k$ to be cancelled and the value of $f$ will be made different from $\Phi_\delta$. This contradicts that $\Phi_\delta$ computes $f$.

Second, let $\phi_\delta(x) = C_f$. By lemma 1 there is a $k$ such that $\phi_\delta = f$ and
\[ \forall x \forall y [f(x) = y \rightarrow \Phi_\delta(x) \leq h(x, y, \phi_\delta(y, x))] \quad (4) \]

Therefore, we have:
\[ \forall x [\phi_\delta(x) = 1 \rightarrow \forall y [f(x) = y \rightarrow h(x, y, \phi_\delta(x)) \geq \Phi_\delta(x)] \]

(4)
\[ \geq h(x, y, \max \{ h_1(x, w, \Phi_\delta(x), \ldots \}) \]

(by Eq. (4))
\[ \max \{ g(x, 0), g(x, 1) \} \}

(by Eq. (3) and $\Phi_\delta(x) = 1 \rightarrow y = f(x) \in \{0, 1\}$)

It follows that:
\[ \forall x \forall y [f(x) = y \rightarrow \Phi_\delta(x) \leq g(x, y, \phi_\delta(x))] \]

since $h$ is monotone in the third variable. End of claim.

Next we assume to the contrary to lemma 5.3 that there is a $\phi_\delta(x)$ computes $C_f$ has the property that:
\[ \exists x [\phi(x) = 1] \]
and
\[ \forall y [f(x) = y \rightarrow \Phi_\delta(x) \leq g(x, y, \phi_\delta(x))] \quad (5) \]

We will use $\phi_\delta$ to construct a program which proves $f(x) = y$ in less than
\[ \max \{ h_1(x, w, \Phi_\delta(x), \max \{ g(x, 0), g(x, 1) \} \} \]

steps for infinitely many $x$ such that $\phi_\delta(x) = 1$. Then this is a contradiction to the claim above.

We claim $\phi_{(\delta_1, \delta_2)}$ will do: from the definition of $\delta_1$, it is easy to see that $\phi_{(\delta_1, \delta_2)}$ computes $C_f$, since $\phi_\delta$ computes $C_f$ and $\Phi_\delta$ is used in defining $f$.

Since $\phi_{(\delta_1, \delta_2)}$ is total and $\phi_\delta(x) = 1$ implies that $f(x) \in \{0, 1\}$. From lemma 5.1 and Eq. (5) above we have the result that for infinitely many $x$ such that $\phi_\delta(x) = 1$ and for $y \in \{0, 1\}$,
\[ \Phi_{(\delta_1, \delta_2)}(x, y) \leq h(x, y, \phi_\delta(x)) \]

\[ \min\{\Phi_\delta(x, 0), \Phi_\delta(x, 1)\} \leq \max \{ h_1(x, w, \Phi_\delta(x), \ldots) \} \]

(5)
\[ \min\{\Phi_\delta(x, 0), \Phi_\delta(x, 1)\} \leq \max \{ h_1(x, w, \Phi_\delta(x), \ldots) \}

(by Eq. (5)).

This is a contradiction to the claim above. End of lemma 5.3.

Combining lemma 5.2 and lemma 5.3, we have the proof of part (a).

(b) We will construct $\phi_{(i, k)}$. When we choose $i = \delta_1$ and $k$ to be an index for $f$, then $\phi_{(\delta_1, k)}$ computes $C_f$. In addition, we will show that there is a recursive $b$, which does not depend on $f$ and $g$ such that $b$ is an upper bound for the number of steps to compute $\phi_{(\delta_1, k)}(x, y)$ whenever $\phi_\delta(x) = 0$ and $y = \phi_\delta(x)$. Look at the definition of $f$, $f(x) = \Phi_\delta(x)$ when $\phi_\delta(x) = 0$.

Therefore $b$ is an upper bound for the number of steps to prove $f(x) = y$ by algorithm $\sigma(i, k)$ when $\phi_\delta(x) = 0$.

Since $\phi_\delta$ is a sufficiently complex 0-1 valued function, we know $\phi_\delta(x) = 0$ for infinitely many $x$.

Let $\sigma$ be a recursive function such that:
\[ \phi_{(i, k)}(x, y) \]
\[ \begin{cases} 1 - |y - \phi(x)| & \text{if } y \leq 1 \text{ and } \phi(x) \text{ converges} \\ 1 & \text{if } y > 1 \text{ and } \Phi_\delta(x) = y \\ 0 & \text{if } y > 1 \text{ and } (\Phi_\delta(x) > y) \text{ or } (\Phi_\delta(x) = y) \\text{or } (\phi(x) \neq 0) \text{ or } (\phi(x) < y) \end{cases} \]

Lemma 5.4:

If $k$ is an index for $f$ and if $\phi_\delta$ is the program used in the construction of $f$, then $\phi_{(i, k)} = C_f$.

Proof:

Observe that $\phi_\delta$ total implies that $\phi_{(i, k)}$ is total.

It is obvious to see $\phi_{(i, k)}(x, y) = C_f(x, y)$ when $y \leq 1$. By definition of $f$, $f(x) = \phi(z)$ only when $\phi(z) = 0$. End of lemma 5.4.
Define \( b \) as follows:

\[
b(x, y) = \begin{cases} 
\max \{ \Phi_{i(k,x)}(x, y) \mid k \leq x \} & \text{if } A \neq \emptyset, \\
\emptyset & \text{otherwise}
\end{cases}
\]

where \( i \in A \). If \( A \neq \emptyset \), then by definition of \( \sigma \), \( \Phi_{i(k,x)}(x, y) = 1 \). So \( \Phi_{i(k,x)}(x, y) \) converges.

Lemma 5.5:

\[
\forall i \forall k \forall x \forall y \left( y > 1 \text{ and } \phi_i(x) = 0 \right) \Rightarrow \left( \Phi_i(x) = y \rightarrow \Phi_{i(k,x)}(x, y) \leq b(x, y) \right).
\]

Proof:

Let \( x \geq \max \{ i, k \} \). If \( y > 1 \) and \( \phi_i(x) = 0 \) and \( \Phi_i(x) = y \), then \( i \) will be in \( A \) then \( b(x, y) \geq \Phi_{i(k,x)}(x, y) \). End of lemma 5.5.

Let \( i = i_0 \) and let \( k \) be an index for \( f \). By lemma 5.4, \( \Phi_{i(k,x)}(x, y) = C_f \). Because \( \phi_{i_0} \) is a sufficiently complex 0–1 valued function, we know \( \Phi_{i_0}(x) = 0 \) for infinitely many \( x \). By definition of \( f \), \( \phi_{i_0}(x) = 0 \) implies \( f(x) = \Phi_{i_0}(x) \) and \( \Phi_{i_0}(x) > 1 \) by Eq. (1). Hence by lemma 5.5, we have:

\[
\exists x \forall y \left( f(x) = y \text{ and } \Phi_{i(k,x)}(x, y) \leq b(x, y) \right)
\]

i.e., we show that \( f \) is easy to prove infinitely often modulo \( b \). This ends the proof of part (b).

ACKNOWLEDGMENTS

The author is deeply indebted to Professor M. Blum for suggesting the topic and close supervision throughout the whole study, especially in the proof of theorem 4.

REFERENCES

1. M. Rabin
   Degree of difficulty of computing a function and a partial ordering on recursive sets
   Technical Report No 2 Hebrew University April 1960

2. M. Blum
   A machine independent theory of computational complexity
   JACM 14 322–336 1967

3. On effective procedure for speeding up algorithms
   ACM Symposium on Theory of Computing Marina del Ray May 1969

4. H. Rogers
   Theory of recursive functions and effective computability