ON COMPUTATIONAL SPEED-UP

By

Albert R. Meyer
Carnegie-Mellon University

Patrick C. Fischer
University of British Columbia

May, 1968

This work was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-146) monitored by the Air Force Office of Scientific Research, and in part by NSF grant GP-7701. Distribution of this document is unlimited.

ON COMPUTATIONAL SPEED-UP

ABSTRACT

Let $\underset{\sim}{F}$ be any effective mapping from total functions on the integers to total functions. Composition and iterated composition are examples of such mappings. The "operator speed-up" theorem in this paper establishes the existence of a computable function f such that for any program computing f(x) in $p_1(x)$ steps for all x, there is another program computing f(x) in $p_2(x)$ steps and $\underset{\sim}{F}(p_2) < p_1$ almost everywhere. Thus, there is no best program for f. The notions of "program" and "number of steps" are treated axiomatically, so that the theorem is independent of any particular model of a computing machine. An example of speed-up for Turing machines is considered.

Key words: computable, computational complexity, Turing machines, recursive functions, speed-up, effective operator, recursion theorem, measure on computation, measure of computation, complexity measure, axioms.

TABLE OF CONTENTS

1. INTRODUCTION

The complexity of a computable function can be measured by considering the time or space required to compute its values. Particular notions of time and space arising from variants of Turing machines have been investigated by R. W. Ritchie [1963], Hartmanis and Stearns [1965], and Arbib and Blum [1965], among others. General properties of such complexity measures have been characterized axiomatically by Rabin [1960] and Blum [1967].

In this paper the speed-up and super speed-up theorems of Blum [1967] are generalized to speed-up by arbitrary general recursive operators. The significance of such theorems is that one cannot equate the complexity of a computable function with the running time of its fastest program, for the simple reason that there are computable functions which in a very strong sense have no fastest programs. However, the structure of our proof suggests the possibility of defining the computational complexity of a function in terms of a recursively enumerable sequence of partial recursive functions.

Rogers' [1958] axioms for Gödel numberings and Blum's axioms for measures on computation are repeated in the next section. The operator speed-up theorem and the lemmas from which it follows are stated in Section 3. The proofs of the lemmas are given in the appendix. An example of operator speed-up for the case of Turing machines and the notion of complexity sequences are considered in Section 4.

2. PRELIMINARIES

Let $\mathbb{N}$ be the non-negative integers, and $\mathcal{P}_n$ ($\mathcal{R}_n$) be the partial recursive (recursive) functions of n variables. A Gödel numbering is a mapping from $\mathbb{N}$ onto $\mathcal{P}_1$ satisfying the universal Turing machine (or normal form) and iteration (or S_n^m) theorems: let $\varphi_i \in \mathcal{P}_1$ be the image of $i \in \mathbb{N}$ under the Gödel numbering, then

(1) (Normal Form Theorem) as a function of i and x, $\varphi_i(x)$ is partial recursive, that is, $\lambda ix[\varphi_i(x)] \in \mathcal{P}_2$, and

(2) (Iteration Theorem) there is a function $\sigma \in \mathcal{R}_2$ and a 1-1 onto function $\tau \in \mathcal{R}_2$ such that $\varphi_i(\tau(x,y)) = \varphi_{\sigma(i,x)}(y)$ for all $i,x,y \in \mathbb{N}$.

The function τ is usually called a pairing function, and "$\langle x,y \rangle$" will be written instead of "$\tau(x,y)$". Computation measuring functions Φ_i for $i \in \mathbb{N}$ are characterized by two further axioms:

(3) $\varphi_i(x)$ converges (i.e., $x \in$ domain (φ_i)) $\Leftrightarrow$ $\Phi_i(x)$ converges, and

(4) the function

$$M(i,x,m) = \begin{cases} 1 & \text{if } \Phi_i(x) = m \\ 0 & \text{otherwise} \end{cases}$$

is in $\mathcal{R}_3$.

Axiom (4) trivially implies that $\lambda ix[\Phi_i(x)] \in \mathcal{P}_2$, and that the predicates $[\Phi_i(x)=m]$ and $[\Phi_i(x) \leq m]$ are recursive in all three arguments i,x, and m. Intuitively, $\Phi_i(x)$ may be thought of as the number of steps required by the i^{th} Turing machine (in a standard enumeration of Turing machines) to halt for input x.

A mapping $\underset{\sim}{F}$ from functions of one variable to functions of one variable will be called an operator. For any function φ, the value of $\underset{\sim}{F}(\varphi)$ at x will be written $\underset{\sim}{F}(\varphi,x)$. If $\varphi: \mathbb{N} \to \mathbb{N}$, let $\bar{\varphi}(x)$ be an effective encoding of φ restricted to the domain $\{0,1,\ldots,x\}$.[1] Let $\mathbb{N}^{\mathbb{N}}$ be the set of total functions from $\mathbb{N}$ into $\mathbb{N}$.

Definition 1. An operator $\underset{\sim}{F}: \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ is continuous if there is a function f of two arguments (f is called the associate of $\underset{\sim}{F}$) such that for every $\varphi \in \mathbb{N}^{\mathbb{N}}$, $x \in \mathbb{N}$,

$$\underset{\sim}{F}(\varphi,x) = f(\bar{\varphi}(z), x) - 1$$

where z is the least y such that $f(\bar{\varphi}(y),x) \neq 0$.

Definition 2. An operator $\underset{\sim}{F}: \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ is general recursive if it is continuous and has an associate in $\mathcal{R}_2$.

Definition 3. An operator $\underset{\sim}{F}: \mathcal{P}_1 \to \mathcal{P}_1$ is effective providing there exists a function $\alpha \in \mathcal{R}_1$ such that $\underset{\sim}{F}(\varphi_i) = \varphi_{\alpha(i)}$ for all $i \in \mathbb{N}$. An effective operator $\underset{\sim}{F}$ is total providing that $\underset{\sim}{F}(\varphi_i)$ is total whenever φ_i is total (that is, $\underset{\sim}{F}(\mathcal{R}_1) \subset \mathcal{R}_1$).

We remark that the restriction to $\mathcal{R}_1$ of any general recursive operator is equal to the restriction to $\mathcal{R}_1$ of some total effective operator (cf. Rogers [1967], ch. 11).

For any function g let $g^{(0)}$ be the identity function and $g^{(y+1)}$ be the composition of g and $g^{(y)}$. Then $\underset{\sim}{I}(g) = \lambda x[g^{(x)}(x)]$ defines a general recursive operator. Also, $\underset{\sim}{G}(g) = \lambda x[\underset{\sim}{\max}\ y \leq x\ \{g(y),x+2\}]$ is general recursive, as is the operator $\underset{\sim}{H} = \underset{\sim}{I} \circ \underset{\sim}{G}$. Finally, extending the superscript

notation to operators, define $\underset{\sim}{F}(g) = \lambda x[[\underset{\sim}{H}^{(x)}(g)](x)]$. $\underset{\sim}{F}$ is a general recursive operator with the property that $\underset{\sim}{F}(g)$ is almost everywhere greater than any function which is primitive recursive in g.

3. OPERATOR SPEED-UP

Theorem (Operator Speed-up). Let g be any function in $\mathcal{R}_1$ and $\underset{\sim}{F}$ be any general recursive operator. There is a 0-1 valued function $f \in \mathcal{R}_1$ such that if $\varphi_i = f$, then (1) $\Phi_i > g$ almost everywhere, and (2) there is a $j \in \mathbb{N}$ such that $\varphi_j = f$ and $\underset{\sim}{F}(\Phi_j) < \Phi_i$ almost everywhere.

The intended interpretation of such speed-up theorems is that there are computable functions f with the pathological property that, given any program (with index i) for computing f, there is another program (with index j) which computes f by an algorithm which is vastly quicker to perform. For example, if we consider the operator $\underset{\sim}{F}$ described at the end of the previous section, then the running time of program i will be almost everywhere greater than any function primitive recursive in the running time of program j.

For any function $r \in \mathcal{R}_2$, the operators mapping φ into $\lambda x[r(x,\varphi(x))]$ or $\lambda x[\varphi(\varphi(x))]$ are general recursive, so that Blum's speed-up and super speed-up theorems follow as special cases of operator speed-up.

The proof of the theorem roughly follows Blum's outline for the super speed-up theorem. Let $P = \{p_i\}_{i=0}^{\infty}$ be a sequence of functions such that $\lambda ix[p_i(x)] \in \mathcal{R}_2$. The function f is computed at x by first computing $\varphi_i(x)$ for $p_i(x)$ steps, $0 \leq i \leq x$, and cancelling the least $i \leq x$ (that is, making $f(x) \neq \varphi_i(x)$) such that $\varphi_i(x)$ converges in the alloted time and i has not been cancelled previously. If no such i exists, f(x) is

set to zero, so that f is total. By construction, if φ_j converges in $\leq p_j$ steps infinitely often, then j will be cancelled. Hence, $\varphi_i=f$ implies $\Phi_i > p_i$ almost everywhere.

Since every index gets cancelled at most once, all indices less than any given u which ever get cancelled will have been cancelled during the computations of f(0), f(1),...,f(v-1) for some $v \in \mathbb{N}$. For $x \geq v$, it is thus possible to compute f(x) by only computing $\varphi_i(x)$ for $p_i(x)$ steps, $u \leq i \leq x$. This procedure is more efficient since it eliminates the computation of $p_i(x)$ for $0 \leq i < u$. The original procedure for f is identical to this procedure with $u = v = 0$.

The preceding "u-v procedure" for computing f is uniform in the parameters $u,v \in \mathbb{N}$. In fact, it is uniform in an index ℓ of the sequence $p_0,p_1,\ldots$, where p_i is, by definition, $\lambda x[\varphi_\ell(\langle i,x\rangle)]$. The index of the u-v-$\ell$ procedure is given by a function $t \in \mathcal{R}_3$. Formally, this is summarized by the following two lemmas:

Lemma 1. There is a function $t \in \mathcal{R}_3$ such that if $\varphi_\ell \in \mathcal{R}_1$ and $f = \varphi_{t(0,0,\ell)}$, then $f \in \mathcal{R}_1$, f is 0-1 valued, and $\varphi_i=f$ implies $\Phi_i > p_i$ almost everywhere (where $p_i = \lambda x[\varphi_\ell(\langle i,x\rangle)]$).

Lemma 2. If $\varphi_\ell \in \mathcal{R}_1$, then for every $u \in \mathbb{N}$ there exists a $v \in \mathbb{N}$ such that $f = \varphi_{t(u,v,\ell)}$ (where f and t are as in lemma 1).

To compute $\varphi_{t(u,v,\ell)}(x)$, the functions $p_u,p_{u+1},\ldots,p_x$ must be computed at x. However, in order to tell which indices have been previously cancelled it is necessary to recompute $\varphi_{t(u,v,\ell)}(y)$ for $y < x$, so that in general $p_j(y)$ must be computed for $u \leq j \leq y \leq x$. For values of $x < v$, the u-v procedure reduces to the 0-0 procedure, so $p_j(y)$ for $0 \leq j \leq y \leq v$

must also be computed. Defining $T_{x,u} = \{(y,j) \mid u \leq j \leq y \leq x\}$, we have (see Figure 1.):

Lemma 3. If $p_j(y)$ converges for all $(y,j) \in T_{x,u} \cup T_{v,0}$, then $\varphi_{t(u,v,\ell)}(x)$ converges.

Intuitively, the computation of $\varphi_{t(u,v,\ell)}(x)$ consists of running the cancelling procedure for a number of steps determined by the values of $p_j(y)$ in the region $T_{x,u} \cup T_{v,0}$. The number of steps required to compute the $p_j(y)$ depends in turn on the index ℓ.[2] Hence, $\Phi_{t(u,v,\ell)}(x)$ depends in a fixed way on Φ_ℓ in the region $T_{x,u}$ ($T_{v,0}$ may be ignored for large x). This should motivate:

Lemma 4. There is a function $\beta \in \mathcal{R}_2$ such that $\varphi_{\beta(u,\ell)}(x) = \max\{\Phi_\ell(\langle j,y\rangle) \mid (y,j) \in T_{x,u}\}$, and there is a function $h \in \mathcal{R}_2$, non-decreasing in its second argument, such that if $\varphi_\ell \in \mathcal{R}_1$, then

$$h(x,\varphi_{\beta(u,\ell)}(x)) > \Phi_{t(u,v,\ell)}(x)$$

for all $u,v \in \mathbb{N}$ and almost all x.

At this point the proof of operator speed-up reduces to constructing an appropriately pathological sequence P and an index ℓ for the sequence so that p_i is almost everywhere vastly largely than both p_{i+1} and the number of steps needed to compute p_{i+1} using program ℓ. In particular, the following lemma is sufficient:

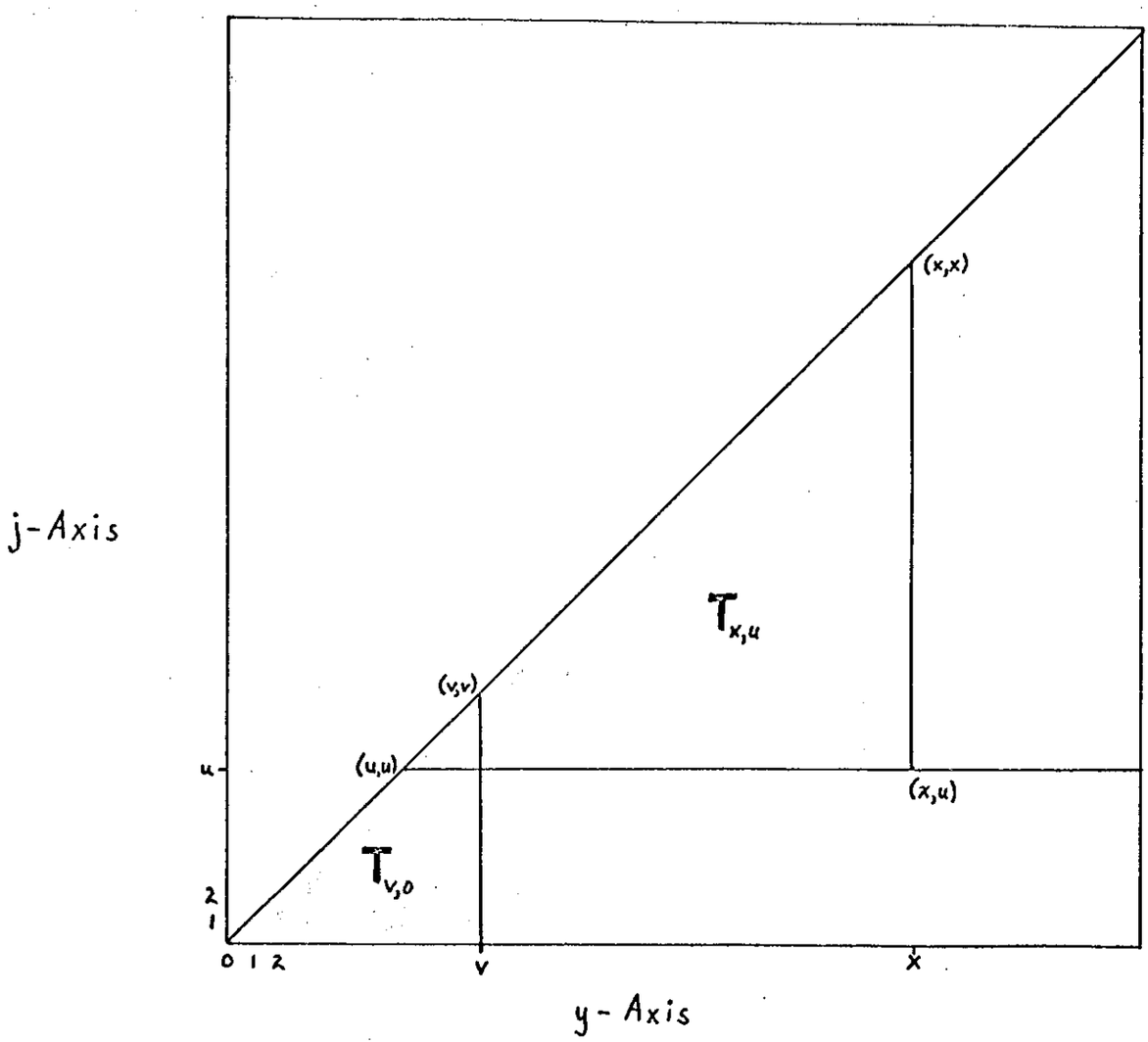

Figure 1. Regions $T_{x,u}$ and $T_{v,0}$.

Lemma 5. For any functions $h \in \mathcal{R}_2$, $g \in \mathcal{R}_1$, and total effective operator $\underset{\sim}{F}$, there is a sequence of functions $P = \{p_i\}_{i=0}^{\infty}$ with index ℓ such that $\varphi_\ell \in \mathcal{R}_1$ and for all $i \in \mathbb{N}$ the following inequalities hold almost everywhere:

5.1) $p_i > g$,

5.2) $p_i > \lambda x[h(x,p_{i+1}(x))]$,

5.3) $p_i > \varphi_{\beta(i+1,\ell)}$ (with β given as in Lemma 4),

5.4) $p_i > \underset{\sim}{F}(p_{i+1})$.

Proof of the theorem: Given $g \in \mathcal{R}_1$ and a total effective operator $\underset{\sim}{F}$, let P be the sequence with index ℓ given by Lemma 5, with $h \in \mathcal{R}_2$ chosen as in Lemma 4. Let $f = \varphi_{t(0,0,\ell)}$, and suppose $\varphi_i = f$.

Since $\varphi_\ell \in \mathcal{R}_1$, Lemma 1 implies that $f \in \mathcal{R}_1$, f is 0-1 valued and $\Phi_i > p_i$ almost everywhere. Lemma 5.1 implies $p_i > g$ almost everywhere, so f satisfies the first clause of the theorem.

Lemma 2 (with $u = i + 3$) implies that $f = \varphi_{t(i+3,v,\ell)}$ for some $v \in \mathbb{N}$. Let $j = t(i + 3,v,\ell)$, so that $f = \varphi_j$; then

$$\Phi_j < h(x,\varphi_{\beta(i+3,\ell)}(x))$$

almost everywhere by Lemma 4. Lemma 5.3 and the fact that h is non-decreasing imply that

$$h(x,\varphi_{\beta(i+3,\ell)}(x)) \leq h(x,p_{i+2}(x))$$

almost everywhere. Lemma 5.2 and the above inequalities finally imply that

$$\Phi_j < p_{i+1}$$

almost everywhere.

Assume for the moment that $\underset{\sim}{F}$ is dominance-preserving.

Definition 4. An operator $\underset{\sim}{F}$ is dominance-preserving if for all total functions $r,s \in$ domain $(\underset{\sim}{F})$, $r \geq s$ almost everywhere implies $\underset{\sim}{F}(r) \geq \underset{\sim}{F}(s)$ almost everywhere.

The preceding inequality now implies

$$\underset{\sim}{F}(\Phi_j) \leq \underset{\sim}{F}(p_{i+1})$$

almost everywhere. Lemma 5.4 and Lemma 1 imply

$$\underset{\sim}{F}(p_{i+1}) < p_i < \Phi_i$$

almost everywhere, and so almost everywhere

$$\underset{\sim}{F}(\Phi_j) < \Phi_i.$$

This proves the theorem for dominance-preserving total effective operators. Moreover, if $\underset{\sim}{F}'(r) \geq \underset{\sim}{F}(r)$ for all $r \in \mathcal{R}_1$, then $\underset{\sim}{F}'$ speed-up trivially implies $\underset{\sim}{F}$ speed-up. The following lemma therefore completes the proof:

Lemma 6. For any general recursive operator $\underset{\sim}{F}$ there is a dominance-preserving general recursive (and hence total effective) operator $\underset{\sim}{F}'$ such that $\underset{\sim}{F}'(r) \geq \underset{\sim}{F}(r)$ for all $r \in \mathcal{R}_1$.

4. COMPLEXITY SEQUENCES

The function f constructed to satisfy the speed-up theorem has no best program or fastest running time, but its complexity can be described by a sequence of possible running times.

Definition 5. A sequence $P = \{p_i\}_{i=0}^{\infty}$ of functions $p_i \in \mathcal{P}_1$ is a complexity sequence for a function $f \in \mathcal{P}_1$ if and only if

(1) domain $(p_i) \supset$ domain (f) for all $i \in \mathcal{N}$,

(2) if $\varphi_j = f$, then for some $i, \Phi_j(x) \geq p_i(x)$ for almost all $x \in$ domain (f).

(3) for every i, there is a j such that $\varphi_j = f$ and $\Phi_j(x) \leq p_i(x)$ for almost all $x \in$ domain (f).

The major part of our proof of operator speed-up amounted to proving that, given any recursively enumerable sequence P of recursive functions satisfying certain simple conditions (viz., Lemma 5.2 and 5.3), one can construct an $f \in \mathcal{R}_1$, with P as its complexity sequence. Thus, although f can be sped up to an extreme degree, its complexity can still be described in a highly constructive manner. Note that any $f \in \mathcal{P}_1$ has a complexity sequence consisting of the elements of $\{\Phi_i \mid \varphi_i = f\}$, but this sequence cannot be recursively enumerable. It is an interesting question whether or not every function in $\mathcal{P}_1$ has some recursively enumerable complexity sequence.

Although the sequence P of Lemma 5 describes the complexity of f, the behavior of P itself is somewhat obscured by the use of the recursion theorem in proving Lemma 5 (cf. the appendix). Henceforth in this section, let $\Phi_i(x)$ be the number of steps required by the i^{th} Turing machine[3] to halt (and print an output) given input x. D. M. Ritchie [1968] has obtained a refined example of speed-up for Turing machines and the operator $\underset{AM}{F}(f) = f \circ f$.

Definition 6. Let $t(x) = 2^x$. A function $h \in \mathcal{R}_1$ is honest if and only if for some i, $h = \varphi_i$ and $\Phi_i(x) \leq t^{(k)}(\max\{h(x),x\})$ for some $k \in \mathcal{N}$ and all $x \in \mathcal{N}$.

The purpose of the definition of honesty is to isolate a class of functions whose size and complexity are approximately the same. For example, Φ_i is always honest whenever it is total, whereas a difficult

to compute 0-1 valued function is highly dishonest.

Theorem (D. M. Ritchie). For any honest, strictly increasing function $g \in \mathcal{R}_1$ such that $g(x) \geq 2^x$, and any unbounded, non-decreasing function $r \in \mathcal{R}_1$, there is a 0-1 valued function $f \in \mathcal{R}_1$ such that

(1) if $\varphi_i = f$, then $\Phi_i > g$ almost everywhere,

(2) if $\varphi_i = f$, then there is a $\varphi_j = f$ such that $\Phi_j \circ \Phi_j < \Phi_i$, almost everywhere (in fact $\Phi_j^{(k)} < \Phi_i$ almost everywhere for any $k \in N$),

(3) there is a $\varphi_i = f$ such that $\Phi_i(x) < g^{(r(x))}(x)$ for almost all x.

To prove the theorem, the sequence P of Lemma 5 is obtained directly (without appeal to the recursion theorem) using a construction discovered jointly by the first author and D. M. Ritchie.[4] The object is to construct a sequence P of honest functions (honesty will yield part (3) of Lemma 5) such that p_i is greater than g and is much greater than p_{i+1} almost everywhere. It is not hard to find a sequence Q of very rapidly increasing honest functions such that q_i is much smaller than q_{i+1}. Taking inverses ($q^{-1}(x)$ = the least z such that $q(z) \geq x$) yields a sequence R of honest, unbounded, non-decreasing functions such that r_i grows more rapidly than r_{i+1}. Set $p_i = \lambda x[g^{(r_i(x))}(x)]$. If the functions q_i are sufficiently large, then the functions r_i will grow more slowly than the given function r. Moreover, r_{i+1} will grow slowly enough that p_i will not only be greater than p_{i+1} but also greater than $p_{i+1}^{(k)}$ for any fixed $k \in N$. We refer the reader to Ritchie [1968] for the complete proof.

Clearly, in order for f to satisfy (1) and (2) of the theorem, the running time of any program for f must almost everywhere exceed $g^{(k)}$ for

any fixed k. Part (3) of the theorem may therefore be interpreted to mean that f is as simple to compute as it possibly could be while satisfying (1) and (2). For example, if g is primitive recursive, the function f will also be primitive recursive. It seems likely that this result can be extended to a more general theorem relating the complexity of f to that of g and $\underset{\sim}{F}$, for arbitrary general recursive operators $\underset{\sim}{F}$.

APPENDIX

PROOFS OF THE LEMMAS

The following definition provides a detailed procedure for computing a partial function $\psi(u,v,\ell,x)$ which equals $\varphi_{t(u,v,\ell)}(x)$ as described in Section 3. It is convenient to define simultaneously a function $L(u,v,\ell,x)$ whose values are finite sets of cancelled indices.[5] As in Section 3, p_i is the function $\lambda x[\varphi_\ell(\langle i,x\rangle)]$.

Definition A. For $u,v,\ell,x \in \mathbb{N}$ the values of $\psi(u,v,\ell,x)$ and $L(u,v,\ell,x)$ are given according to the rules:

1. If $x = 0$ or $v < u$, then $\psi(u,v,\ell,x) = 0$ and $L(u,v,\ell,x) = \emptyset$.
2. If $x \neq 0$ and $v \geq u$ and $v > x$, then $\psi(u,v,\ell,x) = \psi(0,0,\ell,x)$ and $L(u,v,\ell,x) = L(0,0,\ell,x)$.
3. If $x \neq 0$ and $x \geq v \geq u$, compute $p_i(x)$ for $u \leq i \leq x$ and compute $L(u,v,\ell,x-1)$. If any of these computations fail to converge, then $\psi(u,v,\ell,x)$ and $L(u,v,\ell,x)$ are undefined. Otherwise, let n be the least number, if any, such that $u \leq n \leq x$, $\Phi_n(x) \leq p_n(x)$, and $n \notin L(u,v,\ell,x-1)$. If such an n exists, then $\psi(u,v,\ell,x) = 1 \dot{-} \varphi_n(x)$[6] and $L(u,v,\ell,x) = L(u,v,\ell,x-1) \cup \{n\}$. If no such n exists, then $\psi(u,v,\ell,x) = 0$ and $L(u,v,\ell,x) = L(u,v,\ell,x-1)$.

Proofs of Lemmas 1-3: $\psi(0,0,\ell,0)$ and $L(0,0,\ell,0)$ are defined by rule 1 in Definition A for all $\ell \in \mathcal{N}$. Straightforward induction on x implies that if $p_j(y)$ converges for $(y,j) \in T_{x,0}$, then $\psi(0,0,\ell,x)$ and $L(0,0,\ell,x)$ are defined. (Note that $\varphi_n(x)$ is evaluated only when $\Phi_n(x)$ is bounded, so that axiom 3 implies that $\varphi_n(x)$ converges in this case.) Using this result, another induction on x implies that for all $u,v \in \mathcal{N}$, if $p_j(y)$ converges for $(y,j) \in T_{x,u} \cup T_{v,0}$, then $\psi(u,v,\ell,x)$ and $L(u,v,\ell,x)$ are defined.

The rules defining ψ and L are clearly effective. (Note that the existence of n in rule 3 is effectively decidable, given that $p_i(x)$ for $u \le i \le x$ and $L(u,v,\ell,x-1)$ converge, since the predicate $[\Phi_i(x) \le m]$ is recursive by axiom 4.) Thus we conclude that $\psi \in \mathcal{P}_4$. The iteration theorem implies that $\psi(u,v,\ell,x) = \varphi_{t(u,v,\ell)}(x)$ for some $t \in \mathcal{R}_3$ and all $u,v,\ell,x \in \mathcal{N}$.

In particular, if $\varphi_\ell \in \mathcal{R}_1$, then $p_j(y)$ converges for all (y,j), so that $\varphi_{t(u,v,\ell)} \in \mathcal{R}_1$ for all $u,v \in \mathcal{N}$. Definition A guarantees that $\varphi_{t(u,v,\ell)}$ is 0-1 valued wherever it is defined. This proves Lemma 3 and the first part of Lemma 1.

Given ℓ such that $\varphi_\ell \in \mathcal{R}_1$, a number n is said to be <u>cancelled at x</u> if $n \in L(0,0,\ell,x) - L(0,0,\ell,x-1)$. If n is cancelled at some x, then $\varphi_{t(0,0,\ell)} \neq \varphi_n$ since rule 3 forces them to differ at x. Moreover, if $\Phi_n \le p_n$ infinitely often, then rule 3 causes n to be cancelled eventually. Hence, $\varphi_i = \varphi_{t(0,0,\ell)}$ implies $\Phi_i > p_i$ almost everywhere, which proves Lemma 1.

For the above ℓ, rules 1 and 2 imply that $L(0,0,\ell,x) = L(u,v,\ell,x)$ whenever $v > x$ and $v \ge u$. If v is also large enough that every number $n < u$ which gets cancelled does so at some number less than v, then induction on x implies that $L(u,v,\ell,x) = L(0,0,\ell,x)$ and $\varphi_{t(u,v,\ell)}(x) = \varphi_{t(0,0,\ell)}(x)$ for all x. This proves Lemma 2. Q.E.D.

Proof of Lemma 4. The function $\lambda u \ell x[\underline{\max}\{\Phi_\ell(\langle j,y\rangle) \mid (y,j) \in T_{x,u}\}]$ is in $\mathcal{P}_3$ (since the measure function Φ is partial recursive by axiom 4). The iteration theorem implies that it equals $\varphi_{\beta(u,\ell)}(x)$ for some $\beta \in \mathcal{R}_2$.

Define a function h' by:

$$h'(u,v,\ell,x,m) = \begin{cases} \Phi_{t(u,v,\ell)}^{(x)} + 1 & \text{if } m \geq \underline{\max}\{\Phi_\ell(\langle j,y\rangle) \mid (y,j) \in T_{x,u} \cup T_{v,0}\}, \\ 0 & \text{otherwise.} \end{cases}$$

Axiom 4 implies that the predicate used in defining h' is a recursive predicate, so $h' \in \mathcal{P}_5$. Moreover, h' is totally defined since axiom ④³ and Lemma 3 imply that $\Phi_{t(u,v,\ell)}^{(x)}$ converges whenever the predicate is true.

Define a function h by the condition

$h(x,m) = \underline{\max}\{h'(z_1,\ldots,z_5) \mid 0 \leq z_1,\ldots,z_5 \leq x+m\}$; so $h \in \mathcal{R}_2$ and is non-decreasing. Let u,v,ℓ be given such that $\varphi_\ell \in \mathcal{R}_1$; then also $\varphi_{\beta(u,\ell)} \in \mathcal{R}_1$ and $\varphi_{t(u,v,\ell)} \in \mathcal{R}_1$. Let $k = \underline{\max}\{\Phi_\ell(\langle j,y\rangle) \mid (y,j) \in T_{v,0}\}$; then by definition $\varphi_{\beta(u,\ell)}(x) + k \geq \underline{\max}\{\Phi_\ell(\langle j,y\rangle) \mid (y,j) \in T_{x,u} \cup T_{v,0}\}$, and so $h'(u,v,\ell,x, \varphi_{\beta(u,\ell)}(x) + k) = \Phi_{t(u,v,\ell)}^{(x)} + 1$. Therefore, for $x \geq \max\{u,v,\ell,k\}$, we have by definition that $h(x,\varphi_{\beta(u,\ell)}(x)) > \Phi_{t(u,v,\ell)}(x)$.

Q.E.D.

Proof of Lemma 5. Let $\alpha \in \mathcal{R}_1$ be the function such that $\underset{\sim}{F}(\varphi_i) = \varphi_{\alpha(i)}$, and let $\sigma \in \mathcal{R}_2$ be the function of axiom 2. Define a function ψ of two variables as follows:

$$\psi(\ell,\langle i,x\rangle) = \begin{cases} 0 & \text{if } x < i \text{ or } (\exists n \leq i)[\Phi_\ell(\langle 0,n\rangle) \geq x], \\ g(x) + h(x,\varphi_\ell(\langle i+1,x\rangle) + \varphi_{\beta(i+1,\ell)}(x) + \varphi_{\alpha(\sigma(\ell,i+1))}(x) & \text{otherwise.} \end{cases}$$

Clearly, $\psi \in \mathcal{P}_2$. By the recursion theorem (cf. Rogers [1967], ch.11), there is an $\ell \in \mathbb{N}$ such that $\varphi_\ell(w) = \psi(\ell,w)$ for all $w \in \mathbb{N}$. For this ℓ, let $p_i = \lambda x[\varphi_\ell(\langle i,x\rangle)] = \varphi_{\sigma(\ell,i)}$; then the above definition becomes

$$p_i(x) = \begin{cases} 0 & \text{if } x < i \text{ or } (\exists n \le i)[p_0(n) \text{ does not converge in} < x \text{ steps}], \\ g(x) + h(x,\ p_{i+1}(x)) + \varphi_{\beta(i+1,\ell)}^{(x)} + \underset{\sim}{F}(p_{i+1},\ x) & \text{otherwise.} \end{cases}$$

Assume for the moment that $p_0 \in \mathcal{R}_1$. For any i and sufficiently large x, $p_i(x)$ will be defined by the second clause, and hence p_i satisfies the lemma providing it is total. Since p_0 itself is defined by the second clause for large x, it must be that $\varphi_{\beta(1,\ell)}^{(x)}$ converges for large x, and hence $\varphi_\ell(\langle j,y\rangle)$ converges for $(y,j) \in T_{x,1}$. By definition, $\varphi_\ell(\langle j,y\rangle) = 0$ for $y < j$ and by hypothesis $\varphi_\ell(\langle 0,x\rangle)$ converges for all x. In short, $\varphi_\ell \in \mathcal{R}_1$, $p_i \in \mathcal{R}_1$ for all i, and the lemma follows.

It remains to show that p_0 is total. Suppose $p_0(n)$ does not converge for some n; then by the first clause of the definition p_i is total (in fact, identically zero) for all $i \ge n$. In particular, for every x and every $(y,j) \in T_{x,n}$, $p_j(y)$ converges and this implies that $\varphi_{\beta(n,\ell)}$ is total. Also, $\underset{\sim}{F}(p_n)$ is total since $\underset{\sim}{F}$ is a total operator, and similarly, $\lambda x[h(x,p_n(x))]$ is total. Therefore, both clauses in the definition of p_{n-1} guarantee convergence, so p_{n-1} is total. Similarly, $p_{n-2}, p_{n-3}, \ldots, p_0$ must be total, a contradiction.

Q.E.D.

Proof of Lemma 6. Let $\underset{\sim}{F}$ be a general recursive operator. For $\varphi \in \mathbb{N}^{\mathbb{N}}$, $x \in \mathbb{N}$, let $\Psi(\varphi,x) = \{\psi \in \mathbb{N}^{\mathbb{N}} \mid \psi \le \lambda z[\underline{\max}\{\varphi(z),x\}]\}$.

Define an operator $\underset{\sim}{F}'$ with domain $\mathbb{N}^{\mathbb{N}}$ as follows:

$$\underset{\sim}{F}'(\varphi,x) = \underline{\max}\{\underset{\sim}{F}(\psi,y) \mid y \le x \text{ and } \psi \in \Psi(\varphi,x)\}.$$

The operator $\underset{\sim}{F}'$ maps $\mathbb{N}^{\mathbb{N}}$ into $\mathbb{N}^{\mathbb{N}}$. To show this, let $\psi^{[n]}$ be the restriction of $\psi \in \mathbb{N}^{\mathbb{N}}$ to $\{0,1,\ldots,n\}$, and let

$T(\varphi,x) = \{\psi^{[n]} \mid n \in \mathbb{N}, \psi \in \Psi(\varphi,x)\} \cup \{\emptyset\}$. Partially ordered under set inclusion, $T(\varphi,x)$ forms a tree with root $\emptyset$ (the empty set), and with only finitely many branches at each node. There is an obvious 1-1 correspondence between infinite paths through $T(\varphi,x)$ and elements of $\Psi(\varphi,x)$. The continuity of $\underset{\sim}{F}$ implies that for any function (path) ψ and $y \in \mathbb{N}$, the value of $\underset{\sim}{F}(\psi,y)$ is determined by $\psi^{[n]}$ for some n. The infinity lemma (also known as the fan theorem) now implies that for any $y \in \mathbb{N}$, the set $\{\underset{\sim}{F}(\psi,y) \mid \psi \in \Psi(\varphi,x)\}$ is finite and is determined by a finite subset of $T(\varphi,x)$. It follows that $\underset{\sim}{F}'(\varphi,x)$ is defined for all $\varphi \in \mathbb{N}^{\mathbb{N}}$, $x \in \mathbb{N}$.

Moreover, the fact that $\underset{\sim}{F}$ is a general recursive operator implies that $\underset{\sim}{F}'$ is also general recursive, as the reader can easily verify.

It is immediate from the definition of $\underset{\sim}{F}'$ that $\underset{\sim}{F}' \geq \underset{\sim}{F}$ on $\mathbb{N}^{\mathbb{N}}$ and $\underset{\sim}{F}'$ is dominance-preserving.

Q.E.D.

Notes

1. For example, define $\tilde{\varphi}(0) = \varphi(0)$, $\tilde{\varphi}(x+1) = \langle \tilde{\varphi}(x), \varphi(x+1) \rangle$, and $\bar{\varphi}(x) = \langle x, \tilde{\varphi}(x) \rangle$.

2. At this point we differ from the outline in Blum [1967] of the super speed-up theorem. Blum asserts that the running time of the u-v procedure depends on $p_j(y)$ rather than the number of steps required to compute $p_j(y)$. This will be true only for certain sequences whose members are the same size as their measure functions (cf. Lemma 5).

3. Any of the familiar formal definitions of Turing machines and length of computation (number of steps) may be chosen.

4. This construction actually arose independently of the present context (cf. Meyer and Ritchie [1968]).

5. For descriptive purposes we allow the values of L to be finite sets. In a formal definition by recursion, the values of L would be canonical indices of the finite sets (cf. Rogers [1967]).

6. For $y \in \mathbb{N}$, $1 \dot{-} y = \begin{cases} 0 & \text{if } y > 0 \\ 1 & \text{if } y = 0 \end{cases}$.

References

1. Arbib, M. A. and Blum, M. Machine dependence of degrees of difficulty. Proc. AMS, 16, 3 (June, 1965), 442-447.

2. Blum, M. A machine-independent theory of the complexity of recursive functions. J. ACM, 14, 2 (April, 1967), pp. 322-336.

3. Hartmanis, J. and Stearns, R. E. On the computational complexity of algorithms. Trans. AMS, 117, 5 (May, 1965), 285-306.

4. Meyer, A. R. and Ritchie, D. M. Primitive recursive hierarchies. to appear (1968).

5. Rabin, M. O. Degree of difficulty of computing a function and a partial ordering of recursive sets. Tech. Rep. No. 2, Hebrew University, Jerusalem (April, 1960).

6. Ritchie, D. M. Program structure and computational complexity. thesis, Harvard Univ. (1968).

7. Ritchie, R. W. Classes of predictably computable functions. Trans. AMS, 106, 1, (June, 1963), 139-173.

8. Rogers, H., Jr. Gödel numberings of partial recursive functions. J. Symbolic Logic, 23, 3 (Sept. 1958), pp. 331-341.

9. Rogers, H., Jr. Theory of recursive functions and effective computability, McGraw-Hill, N.Y., c.1967.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Carnegie-Mellon University Department of Computer Science Pittsburgh, Pennsylvania 15213	2a. REPORT SECURITY CLASSIFICATION UNCL 2b. GROUP

3. REPORT TITLE

ON COMPUTATIONAL SPEED-UP

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

Albert R. Meyer and Patrick C. Fischer

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
May 1968	21	9

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
SD-146 ARPA b. PROJECT NO. 9718 c. 6154501R d. 681304	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

--Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Air Force Office of Scientific Research 1400 Wilson Boulevard Arlington, Virginia 22209

13. ABSTRACT

Let F be any effective mapping from total functions on the integers to total functions. Composition and iterated composition are examples of such mappings. The "operator speed-up" theorem in this paper establishes the existence of a computable function f such that for any program computing f(x) in $p_1(x)$ steps for all x, there is another program computing f(x) in $p_2(x)$ steps and $F(p_2) < p_1$ almost everywhere. Thus, there is no best program for f. The notions of "program" and "number of steps" are treated axiomatically, so that the theorem is independent of any particular model of a computing machine. An example of speed-up for Turing machines is considered.

DD FORM 1 NOV 65 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT