PROPERTIES OF BOUNDS ON COMPUTATION

by

A. R. Meyer and E. M. McCreight

Carnegie-Mellon University
Department of Computer Science
Pittsburgh, Pa.
April 1969

This research was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense under Contract No. F 44620-67-C-0058 and is monitored by the Air Force Office of Scientific Research, and in part by the Fannie and John Hertz Foundation. This paper is a slightly revised version of a report which will appear in the Proceedings of the Princeton Symposium on Information Sciences, March, 1969.

ABSTRACT

Partial recursive functions which equal the amount of time or space required by computations have special properties which distinguish them from arbitrary partial recursive functions. Our main result illustrates a property of running times similar in interpretation to Borodin's gap theorem. The proof is based on the construction of difficult to compute characteristic functions which take the value one very infrequently.

1. INTRODUCTION

Partial recursive functions which equal the amount of time or space required by computations have special properties which distinguish them from arbitrary partial recursive functions. One can tell effectively whether a Turing machine halts in a given number of steps on a given argument. This means that the function equal to the running time of the machine has a recursive graph.

Borodin [2] has recently demonstrated that there are "gaps" between running times. There are, for example, rapidly increasing recursive functions t such that no running time function has values between $t(x)$ and $2^{t(x)}$ for infinitely many integers x. Hence a program whose running time is bounded above by 2^t must in fact be bounded above by t. Our main result, Theorem 2 below, illustrates a property of running times similar in interpretation to Borodin's gap theorem. Namely, we show that for <u>any</u> recursive function t, there are recursive functions t' which (except at a rapidly vanishing percentage of arguments) can be made arbitrarily much larger than t, such that any running time function bounded above by t' is in fact bounded above by t.

The proof is based on the construction of difficult to compute characteristic functions which take the value one very infrequently. We also note that there are difficult to compute characteristic functions which take the value one at half of their arguments. It follows that the computational complexity of characteristic functions is not related in any simple way to the density of ones among their values. This observation is used to emphasize the contrast between measuring the complexity of a

function in terms of programs which compute a function at all arguments, and programs or circuits which compute finite segments of functions.

2. PRELIMINARIES

N is the set of non-negative integers. "Function" in this paper means function from N to N. φ_i is the partial recursive function computed by the i^{th} Turing machine in a standard enumeration of Turing machines. Φ_i is the running time function of the i^{th} Turing machine; $\Phi_i(x)$ = the number of steps required by the i^{th} Turing machine operating with input x, or $\Phi_i(x)$ is undefined if the i^{th} Turing machine does not halt on input x. We assume that inputs and outputs are given in binary notation.

A function $c:N \to \{0,1\}$ is called a characteristic function. $1\dot{-}x$ is defined to be 1 if x=0, and 0 if $x > 0$. If P(x) is a predicate, then $(\overset{\infty}{\forall}x)[P(x)]$ is equivalent by definition to $(\exists n)(\forall x \geq n)[P(x)]$, and $(\overset{\infty}{\exists}x)[P(x)]$ is equivalent by definition to $(\forall n)(\exists x \geq n)[P(x)]$.

The phrase "there are arbitrarily complicated functions f..." means "for every recursive function g, there is a function f such that φ_i=f implies $(\overset{\infty}{\forall}x)[\Phi_i(x) \geq g(x)]$...." The phrase "for sufficiently complicated functions f..." means "there is a recursive function g, such that for functions f with the property that φ_i=f implies $(\overset{\infty}{\forall}x)[\Phi_i(x) \geq g(x)]$...."

The arguments of this paper apply equally well to the number of tape squares used by Turing machines, or to any measure on computation as characterized by Blum [1]. For simplicity we keep to the number-of-steps measure.

3. LARGE BOUNDS ON RUNNING TIME

If a Turing machine runs for y steps on some input, its output must be smaller than 2^y simply because the machine cannot have printed more than y ones on its output tape. It follows that if f and t are functions and $(\overset{\infty}{\exists}x)[f(x) \geq 2^{t(x)}]$, then f cannot be t-computable since t(x) steps are insufficient to print output f(x). This elementary observation yields

Theorem 1. For every function $t:N \to N$ and every (total) running time function Φ_i, if $(\overset{\infty}{\exists}x)[\Phi_i(x) \geq 2^{t(x)}]$, then there is a function which is Φ_i-computable [6] but not t-computable [6].

Proof. It is easy to show that Φ_i is Φ_i-computable for any total running time function Φ_i (cf. Fischer, Meyer, Rosenberg[4]). By the preceding remarks, Φ_i cannot be t-computable. □

Theorem 1 breaks down completely if we replace the running time function Φ_i by arbitrary recursive functions. In fact if t is any recursive function, there are recursive functions t' which are never smaller than t, are much bigger than t at most arguments, and yet t and t' bound exactly the same set of running times.

Definition. Let $c:N \to \{0,1\}$ be a characteristic function, and let $d:N \to N$ be any function. Then c is d-sparse iff $(\forall x)[c(x)=1 \Rightarrow \forall y(x < y \leq d(x)+x)[c(y)=0]]$.

By choosing d to be rapidly increasing, we guarantee that a d-sparse characteristic function takes the value one at a rapidly vanishing fraction of its first n arguments.

Theorem 2. Let t, g, and d be recursive functions. There is a d-sparse recursive characteristic function c and a recursive function t' such that

(i) $(\forall x)[t'(x) \geq t(x)]$,

(ii) $(\forall x)[c(x)=0 \Rightarrow t'(x) \geq g(x)]$,

(iii) $(\forall i)[(\overset{\infty}{\forall} x)[\Phi_i(x) \leq t(x)] \Leftrightarrow (\overset{\infty}{\forall} x)[\Phi_i(x) \leq t'(x)]]$.

The proof of Theorem 2 follows directly from two lemmas. The following lemma is due to Michael J. Fischer who observed that it was implicit in our original proof of Theorem 2.

Lemma 1. (M. Fischer) Let t be a recursive function. For every sufficiently complicated recursive characteristic function c and every $i \in N$, $(\overset{\infty}{\forall} x)[c(x)=1 \Rightarrow \Phi_i(x) \leq t(x)] \Rightarrow (\overset{\infty}{\forall} x)[\Phi_i(x) \leq t(x)]$.

Proof. Let c be any recursive characteristic function and suppose that for some i, $n \in N$, if $x \geq n$ and $c(x)=1$, then $\Phi_i(x) \leq t(x)$. Design a new machine which computes c as follows: "Given input x, see if $x \geq n$. If not, compute c(x) (by simulating some given machine for c) and give output c(x). Otherwise, compute t(x), and then simulate machine i on input x for t(x) steps. If machine i does not halt in the allotted time, give output zero. Otherwise, compute c(x) and give output c(x)."

Whenever $x \geq n$ and $\Phi_i(x) > t(x)$, the machine operating according to the preceding instructions will halt with output zero in at most twice the number of steps required to compute t(x). Hence, suppose c is any recursive characteristic function such that every machine computing c takes more than twice as many steps at almost all arguments as some machine for t. Since the preceding machine computes c, it must be that

$\Phi_i(x) \leq t(x)$ for almost all x. □

Lemma 2. For every recursive function d, there are arbitrarily complicated recursive characteristic functions which are d-sparse.

Proof. The proof is a generalization of an argument of Rabin [7] which shows that there are arbitrarily complicated characteristic functions. Let g be any recursive function. We compute a d-sparse characteristic function c in stages, so that c differs from the function computed by any machine which halts in $\leq g(x)$ for infinitely many x. At any point in the computation c is defined on an initial segment of the integers, and execution of another stage increases the length of the segment by one or more arguments.

Initial stage. Set $c(y)=0$ for $0 \leq y \leq d(0)$. Set the list of cancelled indices to be empty.

Next stage. Let $x \in N$ be the least argument at which $c(x)$ is not yet defined. See if there is a $j \leq x$ such that j is not on the list of cancelled indices, and such that $\Phi_j(y) \leq g(y)$ for some y, $x \leq y \leq x + d(x)$. If no such j exists, define $c(x)=0$ and go to the next stage.

Otherwise, let i be the least j satisfying the above conditions. If $\Phi_i(x) > g(x)$, define $c(x)=0$ and go to the next stage. Otherwise, define $c(x) = 1 \dot{-} \varphi_i(x)$, define $c(y)=0$ for $x < y \leq x + d(x)$, add i to the list of cancelled indices, and go to the next stage. END

The only part of the computation of c which might appear non-effective is the computation of $1 \dot{-} \varphi_i(x)$, but $\varphi_i(x)$ is computed only when $\Phi_i(x) \leq g(x)$,

and so the computation of $1 \dot{-} \varphi_i(x)$ is guaranteed to converge in this case. Since each stage extends the initial segment in which c is defined, it follows that c is (total) recursive. Moreover, inspection of the procedure reveals that whenever c(x) is defined to be one, the next d(x) values of c are defined to be zero. Hence, c is d-sparse.

It remains to show that every machine computing c takes more than g steps at almost all arguments. The following assertion can be proved by induction on i; the proof is left to the reader.

Fact. If $i \in N$ is not on the list of cancelled indices at some stage in the computation of c, and if $\Phi_i(x) \leq g(x)$ for some x such that c(x) is not defined at that stage, then there is a $j \leq i$ which will be placed on the list of cancelled indices at some later stage.

Now suppose $\varphi_i = c$. Then i is never placed on the list of cancelled indices (since c differs from φ_j for every j which is cancelled). There will be some stage in the computation of c by which all of the finitely many integers $j \leq i$ which will ever be cancelled, have already been cancelled. Then for all values of c(x) defined at later stages, it follows from the fact above that $\Phi_i(x) > g(x)$. □

Proof of Theorem 2. Given recursive functions t, g, and d, let c be a d-sparse recursive characteristic function which is sufficiently complicated that Lemma 1 is satisfied. By Lemma 2 such a c exists. Let $t'(x) = t(x) + g(x) \cdot (1 \dot{-} c(x))$. Clearly t' satisfies parts (i) and (ii) of the theorem, and since $t' \geq t$, the left to right implication in (iii) is immediate. For the converse implication, suppose $(\overset{\infty}{\forall} x)[\Phi_i(x) \leq t'(x)]$.

Then since $c(x)=1$ implies $t'(x) = t(x)$, it follows that $(\overset{\infty}{\forall}x)[c(x)=1 \Rightarrow \Phi_i(x) \leq t(x)]$. By Lemma 1 we conclude that $(\overset{\infty}{\forall}x)[\Phi_i(x) \leq t(x)]$. □

4. COMPLEXITY OF FINITE SEGMENTS

The times required by machines which compute a function f reflect one notion of the computational complexity of f. However, one can also take the view that since it is never possible to compute more than a finite portion of f, the complexity of f should be measured in terms of the complexity of its finite segments. Both notions of complexity are clearly of interest, but it is important to realize that they are quite different.

The complexity of a finite function has been measured by the size or depth of circuits which realize them (Winograd [9], Spira [8]). Let $c:\{0,1...,2^{n-1}\} \rightarrow \{0,1\}$ be a function. A logical circuit with n input lines and a single output line realizes c, if whenever the digits of the binary representation of $x \in \{0,1,...,2^{n-1}\}$ (with leading zeros if necessary) are applied on the n input lines of the circuit, the value on the output line is $c(x)$. The <u>circuit complexity</u> of c can be defined as the minimum value of the depth of circuits which realize c using only two-input logical gates. The circuit complexity of c is closely related to the minimum depth of parentheses in Boolean expressions for c using binary Boolean operations.

Alternatively, Kolmogorov, Chaitin and others have considered what might be called the <u>descriptive complexity</u> of c as the size of a minimal Turing machine or program for c. For example, we could define the descriptive complexity of c to be the least j such that φ_j restricted to

$\{0,1,\ldots,2^{n-1}\}$ equals c.

The circuit (or descriptive complexity) of a function $f:N \rightarrow \{0,1\}$ can be defined as a function of n equal to the circuit complexity of f restricted to $\{0,1,\ldots,2^{n-1}\}$. It now makes sense to consider the circuit complexity not only of recursive f, but any characteristic function f. This is the first indication of the difference between the running time notion of complexity, which is defined only for recursive functions, and the notions of complexity of finite segments. Informally, the former notion reflects the difficulty of, given n, finding a circuit or effective description of n values of f, whereas the latter notions reflect the difficulty of storing these values once they are found.

Both circuit and descriptive complexity of functions have the property that characteristic functions which equal one on sparse sets have low complexity. For example, if $c:N \rightarrow \{0,1\}$ is d-sparse for $d(x) > 2^x$, then the circuit complexity of c is bounded above by approximately $\log_2 n$. It can be shown that most characteristic functions on $\{0,1,\ldots,2^{n-1}\}$ have circuit complexity $n^{1-\epsilon}$. By Lemma 2 we conclude that there are arbitrarily complicated (in terms of running time) recursive characteristic functions whose circuit complexity grows quite slowly. We shall postpone a more thorough investigation of these observations for a later paper.

Another consequence of Lemma 2 is that there are arbitrarily complicated recursive functions which can be approximated with high precision by very easily computed functions. The constant function equal to zero, for example, is a reasonable approximation to an exponentially sparse characteristic function.

Finally, as a contrast to Lemma 2 we establish the existence of arbitrarily complicated characteristic functions with ones and zeroes evenly distributed among their values. The proof was suggested by R. W. Floyd.

Lemma 3. There are arbitrarily complicated recursive characteristic functions c such that $(\forall x)[c(2x) \neq c(2x+1)]$, and hence $|\{x \mid 0 \leq x \leq 2n \text{ and } c(x)=1\}| = n$.

Proof. Let g be a strictly increasing recursive function, and $h(x) = g(2x+1) + x + 1$. Let c' be a recursive characteristic function such that $\varphi_i=c'$ implies $(\overset{\infty}{\forall} x)[\Phi_i(x) \geq h(x)]$. Such a c' exists by Lemma 2. Let $c(2x) = c'(x)$ and $c(2x+1) = 1 \dot{-} c'(x)$. If $\varphi_j=c$, then a simple modification of machine j yields a new machine i which computes c'. Given input x the machine i adds another digit zero at the low order end of the representation of x, and then simulates machine j on the altered input.

Thus $\varphi_i=c'$, and $\Phi_i(x)$ equals $\Phi_j(2x)$ plus the time to add a digit at the end of the input - which certainly requires fewer than x+1 steps. Hence, $(\forall x)[\Phi_j(2x) + x + 1 \geq \Phi_i(x)]$. But $(\overset{\infty}{\forall} x)[\Phi_i(x) \geq h(x)]$ by choice of c', and so $(\overset{\infty}{\forall} x)[\Phi_j(2x) \geq g(2x)]$. A similar argument implies that $(\overset{\infty}{\forall} x)[\Phi_j(2x+1) \geq g(2x+1)]$, and therefore $(\overset{\infty}{\forall} x)[\Phi_j(x) \geq g(x)]$.

By choosing g to be rapidly increasing, c can be made arbitrarily complicated, and by definition $c(2x) \neq c(2x+1)$. □

References

1. Blum, M., A machine independence theory of computation complexity, JACM, v.14 (1967), 322-336.

2. Borodin, A., Complexity classes of recursive functions and the existence of complexity gaps, Proc. of ACM Symposium on Theory of Computing (1969), to appear.

3. Chaitin, C., On the length of programs for computing finite binary sequences, JACM, v. 13, No. 4 (1966), 547-569.

4. Fischer, P. C., A. R. Meyer, and A. L. Rosenberg, Real-time counter machines, Eighth Annual IEEE Symp. on Switching and Automata Theory, Austin, Texas (1967), 148-154.

5. Kolmogorov, A. N., Three approaches to the definition of the concept "quantity of information", Problemy Peredachi Informatsii, v. 1 (1965), 3-11. (in Russian)

6. McCreight, E. M., and A. R. Meyer, Classes of computable functions defined by bounds on computation, Proc. of ACM Symposium on Theory of Computing (1969), to appear.

7. Rabin, M. O., Degree of difficulty of computing a function and a partial ordering of recursive sets, Tech. Report No. 2, Hebrew University, Israel (1960).

8. Spira, P. M., The time required for group multiplication, JACM, v. 14 (1969).

9. Winograd, S., On the time required to perform multiplication, JACM, v. 14, No. 4, (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, Penn sylvania 15213

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE
PROPERTIES OF BOUNDS ON COMPUTATION

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

5. AUTHOR(S) *(First name, middle initial, last name)*
A. R. Meyer
E. M. McCreight

6. REPORT DATE
April 1969

7a. TOTAL NO. OF PAGES
12

7b. NO. OF REFS
9

8a. CONTRACT OR GRANT NO.
F44620-67-C-0058

b. PROJECT NO.
9718

c.
61545O1R

d.
681304

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT
1. This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES
TECH, OTHER

12. SPONSORING MILITARY ACTIVITY
Air Force Office of Scientific Research (SRMA) 1400 Wilson Boulevard Arlington, Virginia 22209

13. ABSTRACT

Partial recursive functions which equal the amount of time or space required by computations have special properties which distinguish them from arbitrary partial recursive functions. Our main result illustrates a property of running times similar in interpretation to Borodin's gap theorem. The proof is based on the construction of difficult to compute characteristic functions which take the value one very infrequently.

DD FORM 1 NOV 65 1473

Security Classification

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