INFORMATION AND CONTROL 11, 257–265 (1967)

On the Size of Machines*

MANUEL BLUM

MIT, Department of Mathematics and Research Laboratory of Electronics, Cambridge, Massachusetts

In this paper, the methods of recursive function theory are used to study the size (or cost or complexity) of machines. A positive result of this study shows that to a remarkable degree, the relative size of two machines is independent of the particular way in which machine size is measured. Another result suggests that in order for programs to be of economical size, the programming language must be powerful enough to compute arbitrary general recursive functions, rather than some restricted subset such as the primitive recursive functions. Finally, a kind of speedup theorem is proved which is curiously independent of whether the measure of complexity be the size or the number of steps taken by the machines that compute the functions.

INTRODUCTION

By machines we mean Turing machines, idealized computer programs, or any idealized devices for computing the recursive functions. The machines and their size must satisfy the axioms of section 1. Section 3 introduces a general set of axioms for step-counting. These determine what is acceptable as a definition of "a step" in a computation. These axioms are all so fantastically weak that any reasonable model of a computer and any reasonable definition of size and step satisfies them.

All our *examples* refer to a specific class of machines and a specific notion of size and step: The class of machines is Davis' (1958) 1-tape, 2-symbol unary radix Turing machines, as defined by sets of quadruples. The size of a Turing machine is defined to be the number of quadruples that define it. A single step in a computation is defined to be a shift of

* This work was supported in part by the National Institutes of Health (grant 5 R01 NB-04985-03), the U. S. Air Force (Aero Space Medical Division) under contract AF 33 (615)-3885, The Teagle Foundation, Inc., and Project MAC, an MIT research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research contract No. Nonr-4102(01).

the tape by 1 square (left or right) or a single print or erasure on a square.

Our theorems, if not our examples, refer to arbitrary devices and any measures of size and step-counting that satisfy the axioms.

1. The natural numbers are $N = 0, 1, 2, \cdots$. We let (φ_i) with i ranging over N denote a recursively enumerable (r.e.) sequence of *all* partial recursive (p.r.) functions of one variable. We let (φ_i^k), k being a fixed integer $\geqq 2$, denote an r.e. sequence of all p.r. functions of k variables.

DEFINITION (Rogers, 1958). The sequence (φ_i) has an *acceptable Godel numbering* if and only if it satisfies the universal Turing machine theorem and the iteration theorem, where

UNIVERSAL TURING MACHINE THEOREM. *There exists a partial recursive function f such that $f(i, x) = \varphi_i(x)$ for all i and x.*

ITERATION THEOREM. There exists a recursive function s such that $\varphi_i^2(x, y) = \varphi_{s(i,x)}(y)$ for all i, x, y.

THEOREM (*Rogers'* 1958). *If (φ_i) and (ψ_i) both have acceptable Godel numberings, then one of these sequences is just a recursive permutation of the other, i.e., there exists a* 1-1 *onto recursive function $f\colon N \to N$ such that $\varphi_{f(i)} = \psi_i$ for all i.*

This is a deep theorem with immediate consequences, as we shall see, for our study of machine size.

We assume that each φ_i is computed by some set of instructions M_i. M_i may be a Turing machine, a program for an idealized computer, or a set of equations. We say that the sequence (M_i) has an acceptable Godel numbering if (φ_i) has one, and in the remainder of this paper we always assume that (φ_i) has one. Roger's theorem allows us to assume, and so we do assume without loss of generality, that any two classes of machines (M_i) and (T_i) (both with acceptable Godel numberings) are so ordered that M_i and T_i compute the same function φ_i.

AXIOMS. *A recursive function Φ mapping N (viewed as the set of indices) $\to N$ (viewed as the set of sizes) is called a measure of the size of machines, Φ_i being called the size of M_i, if and only if*

(1) *there exist at most a finite number of machines of any given size and*

(2) *there exists an effective procedure for deciding, for any y, which machines are of size y.*

Suppose we compare the machines in one class (M_i) with those in another (T_i). For example, (M_i) might be the class of 1-tape 2-symbol Turing machines, while (T_i) might be the class of 10-tape 10-symbol

Turing machines. The following proposition shows that the size of M_i is approximately equal to the size of T_i. It is an immediate corollary of Roger's theorem:

PROPOSITION 1. *Let* $\|_M$ *and* $\|_T$ *be measures of the size of the machines* (M_i) *and* (T_i) *respectively. Then there exists a recursive function* g *such that for all* i:

1. $|\, i \,|_M \leqq g\,|\, i \,|_T$ (*i.e.,* M_i *cannot be much larger than* T_i)

2. $g\,|\, i \,|_M \geqq |\, i \,|_T$ (*i.e.,* T_i *cannot be much larger than* M_i).

Proof. Let $g(x) = \max\{g_1(x), g_2(x)\}$ where g_1 is defined so that $|\, i \,|_M \leqq g_1\,|\, i \,|_T$ and g_2 is defined so that $g_2\,|\, i \,|_M \geqq |\, i \,|_T$: To compute $g_1(x)$, note that $\{i \mid |\, i \,|_T = x\}$ is finite and may be determined effectively. Let $g_1(x) = \max\{\, |\, i \,|_M \mid i$ satisfies $|\, i \,|_T = x\}$. Let $g_2(x) = \max\{\, |\, i \,|_T \mid i$ satisfies $|\, i \,|_M = x\}$. Q.E.D.

If M_i is much smaller than M_j, then one suspects that T_i is smaller than T_j. This is the content of the following:

PROPOSITION 2. *There exists a recursive* h *such that for all* i, j:

1. $|\, i \,|_M \leqq |\, j \,|_M \Rightarrow |\, i \,|_T \leqq h\,|\, j \,|_T$

2. $h\,|\, i \,|_M \leqq |\, j \,|_M \Rightarrow |\, i \,|_T \leqq |\, j \,|_T$

Proof. First note that the g in proposition 1 can be made monotonically increasing by setting

$$g(x) = 1 + \max\{g_1(x), g_2(x), g(x-1)\}.$$

We assume that g is monotonically increasing. Set $h(x) = gg(x)$:

1. If $|\, i \,|_M \leqq |\, j \,|_M$, then $|\, i \,|_T \leqq g\,|\, i \,|_M$ (by 2 of proposition 1) $\leqq g\,|\, j \,|_M$ (because g is monotonically increasing) $\leqq gg\,|\, j \,|_T$ (by 1 of proposition 1) $= h\,|\, j \,|_T$.

2. If $h\,|\, i \,|_M \leqq |\, j \,|_M$, then by a similar argument $g\,|\, i \,|_T \leqq gg\,|\, i \,|_M = h\,|\, i \,|_M \leqq |\, j \,|_M \leqq g\,|\, j \,|_T$, and since g is monotonically increasing, $|\, i \,|_T \leqq |\, j \,|_T$. Q.E.D.

2. A Turing machine may compute the constant function $c_n(x) = n$ by storing all n digits of the response inside its quadruples. In general such a machine will be overly large. For example, if $n = 10^{10}$, a smaller machine may output 10^{10} by multiplying 10 by itself 10 times, rather than by remembering all 10^{10} digits. We show that any infinite r.e. sequence of machines contains some machines whose size can be reduced in this way.

THEOREM 1. 1. *Let* g *by any recursive function with infinite range* (g *enumerates indices of an infinite sequence of machines*).

2. *Let* f *be any recursive function. Then there exist* $i, j \in N$, *both*

uniform in f, g such that

3. $\varphi_i = \varphi_{g(j)}$

4. $f \mid i \mid < \mid g(j) \mid$ (*i.e., M_i is considerably smaller than $M_{g(j)}$*).

EXAMPLE. If $f(x) = 100\,x$, then $M_{g(j)}$ is 100 times as large as M_i, though both $M_{g(j)}$ and M_i compute the same function.

Proof. We give a procedure for determining the two integers i and j uniformly in f, g. Consider the following set of instructions:

"With inputs x and y, first compute $f \mid y \mid$. Then compute $g(0)$, $g(1)$, $\cdots$ until the least j is found such that $f \mid y \mid < \mid g(j) \mid$. Then compute $\varphi_{g(j)}(x)$ and give $\varphi_{g(j)}(x)$ as output."

These instructions define a partial recursive function $\varphi_z^{\,2}(x, y)$ whose index z is uniform in f, g. The recursion theorem[1] then supplies an integer i which is uniform in f, g such that

$$\textit{Equation 1}\colon\ \varphi_i(x) = \varphi_z^{\,2}(x, i) \text{ for all } x.$$

We shall show that this is the desired i.

Conditions 1, 2 ensure that we can find a j uniformly in f, g that satisfies 4, $f \mid i \mid < \mid g(j) \mid$: First determine i, which we have shown to be uniform in f, g. Then compare $f \mid i \mid$ with $\mid g(0) \mid$, $\mid g(1) \mid$, $\cdots$ until a j is found that satisfies 4. For this j (cf. Eq. 1 and above instructions) condition 3 is satisfied, $\varphi_i(x) = \varphi_z^{\,2}(x, i) = \varphi_{g(j)}(x)$ for all x. Q.E.D.

g is an algorithmic function for enumerating an infinite set of machines. The constructive nature of this proof enables one to effectively replace g by a function g' which enumerates machines that are no larger, and sometimes are considerably smaller than those enumerated by g.

It has been said that since practically all computable functions are primitive recursive, one does not need general recursion for any *practical* purposes. Theorem 1, though, gives practical reasons for favoring general recursion: It implies that there exists a primitive recursive function whose smallest derivation (defining equations) in the primitive recursive format is considerably larger than its smallest derivation in the general recursive format. More precisely, suppose primitive and general recursion are defined by derivations as in Davis (1958). Take the size of a derivation to be the number of letters in it. Then each primitive recursive function has at least one *smallest* primitive recursive derivation. The set of all such smallest derivations is r.e.; let g enumerate them. Upon setting $f(x) = n \cdot x$, the theorem supplies a primitive recursive

[1] The recursion theorem asserts that for every partial recursive function h there exists an integer i which is uniform in h such that $\varphi_i(x) = h(x, i)$ for all x.

function whose smallest primitive recursive derivation is n times as large as its smallest general recursive derivation. The method of theorem 2 can be used to show that the primitive recursive derivation and the much smaller general recursive derivation take approximately the same number of steps to compute the same function.

3. When a machine is reduced in size, it frequently happens that the smaller machine takes more steps than the original larger one to do its computations. To prepare the way for a study of this phenomenon, we now introduce the axioms of Blum (1967) for step-counting. This is done by associating with each partial recursive function φ_i another partial recursive function Φ_i called its step-counting function. Intuitively, $\Phi_i(x)$ represents the number of steps taken by M_i with input x to compute $\varphi_i(x)$.

AXIOMS. *Suppose the sequence of all partial recursive functions* (φ_i) *has an acceptable Godel numbering. Let* (Φ_i) *denote any r.e. sequence of (some but not necessarily all) partial recursive functions. We say that* (Φ_i) *is a sequence of step-counting functions for* (φ_i) *if and only if*

1. $\varphi_i(x)$ *is defined* $\Leftrightarrow \Phi_i(x)$ *is defined.*
2. *There exists a recursive function* R *such that*

$$R(i, x, y) = \begin{cases} 1 & \text{if } \Phi_i(x) = y \\ 0 & \text{if not.} \end{cases}$$

The first axiom asserts that if $\varphi_i(x)$ is defined, then the number of steps required to compute it is finite, and vice versa, if the number of steps required to compute $\varphi_i(x)$ is finite, then $\varphi_i(x)$ is defined. We shall write "$\Phi_i(x) < \infty$" instead of "$\Phi_i(x)$ is defined," and "$\Phi_i(x) = \infty$" instead of "$\Phi_i(x)$ is undefined." The second axiom asserts that there is an algorithm for deciding whether or not M_i with input x halts in y steps. The predicate R of this axiom is similar to Kleene's T-predicate.

We note that axioms 1 and 2 are independent: A sequence of step-counting functions that satisfies axiom 1 but not 2 is obtained by setting $\Phi_i = \varphi_i$ for all i. On the other hand, $\Phi_i(x) = 0$ for all i, x satisfies 2 but not 1.

In the following sections, we shall always assume that (Φ_i) is a sequence of step-counting functions for (φ_i). The function R will never be mentioned, though it is implicit in every statement of the form "Determine whether $\Phi_i(x) = y$". This is determined by computing $R(i, x, y)$.

4. The following theorem, which extends theorem 1, shows that

the increase in number of steps that occurs when a machine is reduced in size is negligible for all functions that are sufficiently difficult to compute.

THEOREM 2. *There exists a recursive function h such that if*

1. *g is any recursive function with infinite range*
2. *f is any recursive function*

then there exist $i, j \in N$, both uniform in f, g, such that

3. $\varphi_i = \varphi_{g(j)}$
4. $f \mid i \mid < \mid g(j) \mid$
5. *for all but a finite number of integers x, $[\varphi_{g(j)}(x)$ defined $\Rightarrow \Phi_i(x) \leqq h(x, \Phi_{g(j)}(x))]$ (i.e., M_i does not take too many more steps than $M_{g(j)}$).*

Remark. The function $h(x, y) = c(x + y)$, $c =$ constant, serves for the class of Turing machines defined in the introduction. If these Turing machines can have an unbounded number of tape symbols, then one may choose $c = 1$.

Proof. We define a recursive function r. This r has the property that for any f, g satisfying 1, 2 there exist integers i, j, *in fact, the ones supplied by the proof of theorem 1*, and there exists an integer w such that

(a) $\varphi_{g(j)}(x)$ defined $\Rightarrow r(w, x, \Phi_{g(j)}(x)) = \Phi_i(x)$ for all x.

Out of r we define h to be the recursive function

$$h(x, y) = \max \{r(w, x, y) \mid w \leqq x\}$$

whence it follows that for all sufficiently large x

(b) $\varphi_{g(j)}(x)$ defined $\Rightarrow h(x, \Phi_{g(j)}(x)) \geqq r(w, x, \Phi_{g(j)}(x))$.

Together, (a) and (b) prove 5.

DEFINITION OF $r(w, x, y)$. We choose some fixed effective 1-1 onto map $N \rightarrow \underbrace{N \times \cdots \times N}_{5}$. The number w is a code word for a 5-tuple $(w_1, \cdots, w_5)$. Set $f = \varphi_{w_1}$, $g = \varphi_{w_2}$. Determine i in terms of these f, g, using the algorithm in the proof of theorem 1. This is possible since that algorithm defines an index i for any partial recursive functions f, g, not necessarily satisfying 1, 2 (though j is not necessarily defined unless 1, 2 are satisfied). Now set $r(w, x, y) = \Phi_i(x)$ if all the following hold:

(i) $\Phi_{w_1} \mid i \mid = w_3$ (so $f \mid i \mid$ is defined)

(ii) $\sum_{k=0}^{w_4} \Phi_{w_2}(k) = w_5$ (so $g(0), \cdots, g(w_4)$ are defined)

(iii) There exists a j such that $j \leqq w_4$ and $f \mid i \mid < \mid g(j) \mid$

(iv) The least j satisfying (iii) satisfies $\Phi_{g(j)}(x) = y$

Set $r(w, x, y) = 0$ otherwise.

If (i)–(iv) are satisfied, then Φ_i and $\Phi_{g(j)}$ have the same domain, and

$\Phi_{g(j)}(x)$ is defined by (iv). We note that if f, g satisfy 1, 2, then there exists a j such that $|\, g(j)\,| > f(|\, i\,|)$. Hence there exists a $w = \langle w_1, \cdots, w_5\rangle$, w_1, w_2 being indices of f, g, which satisfies (i)–(iii). For this w, equation (a) is satisfied. Q.E.D.

One can show that theorem 2 is made false by replacing the "for all but a finite number of integers x" in 5 by "for all x". This is interesting because theorem 3, which looks so much like theorem 2, *does* have "for all x" in its version of 5.

THEOREM 3. 1. *Let g be a recursive function satisfying $\varphi_{g(j)} = \varphi_j$ for all j.*

2. *Let f, h be recursive functions.*

Then for every $i \in N$ there exists a $j \in N$ which is uniform in f, h, g, i, such that

3. $\varphi_i = \varphi_{g(j)}$

4. $f(|\, i\,|) < |\, g(j)\,|$ (*i.e., M_i is considerably smaller than $M_{g(j)}$*)

5. *For all $x[\varphi_i(x)$ defined $\Rightarrow h(x, \Phi_i(x)) < \Phi_{g(j)}(x)]$ (i.e., M_i takes considerably fewer steps than $M_{g(j)}$)*

Proof. Select any integer i. We seek a j for which 3, 4 and 5 are true. This j is gotten as follows: We define a partial recursive function $r(n, x, y)$ uniformly in f, h, g, i (Eq. 1). By an application of the recursion theorem we obtain from it a recursive function q such that $\varphi_{q(n)}(x) = r(n, x, q(n))$ (Eq. 2). This q is uniform in f, h, g, i. Finally, we show how to effectively select n, so that $j = q(n)$ satisfies 3, 4, 5.

Equation 1: $r(n, x, y) = n$ if $f(|\, i\,|) \geqq |\, g(y)\,|$ (i.e., if 4 is *not* satisfied with $y = j$)

$\varphi_i(x)$ if (i) $f(|\, i\,|) < |\, g(y)\,|$ (4 is satisfied with $y = j$)
and (ii) $\varphi_i(x)$ is defined
and (iii) $h(x, \Phi_i(x)) < \Phi_{g(y)}(x)$ (5 is satisfied with $y = j$)
undefined otherwise.

This r is computable. The recursion theorem[2] supplies a recursive function q which is uniform in f, h, g, i, such that $\varphi_{q(n)}(x) = r(n, x, q(n))$.

[2] The extended recursion theorem states that for every partial recursive function $r(x_1, \cdots, x_m, y_1, \cdots, y_n, z)$ there exists a recursive function q which is uniform in r such that $\varphi_{q(x_1, \cdots, x_m)}(y_1, \cdots, y_n) = r(x_1, \cdots, x_m, y_1, \cdots, y_n, q(x_1, \cdots, x_m))$.

Plugging into the definition of r, we see that

Equation 2: $\varphi_{g(n)}(x) = n$ if $f \mid i \mid \geqq \mid gq(n) \mid$
$\varphi_i(x)$ if (i) $f \mid i \mid < \mid gq(n) \mid$
and (ii) $\varphi_i(x)$ is defined
and (iii) $h(x, \Phi_i, (x)) < \Phi_{gq(n)}(x)$
undefined otherwise

How to Select n. First note that $f \mid i \mid \geqq \mid gq(n) \mid$ means that there exists a machine which computes the constant function $\varphi_{gq(n)}(x) = \varphi_{q(n)}(x) = n$, whose size is less than a certain fixed integer $f \mid i \mid$. This is impossible for all n, so there exists an n such that $f \mid i \mid < \mid gq(n) \mid$. To find this value of n, simply test $q(0)$, $q(1)$, $q(2)$, $\cdots$ until an n appears that satisfies $f \mid i \mid < \mid gq(n) \mid$. For this particular choice of n, set $j = q(n)$. So 4 is satisfied. It follows that

Equation 3: $\varphi_j(x) = \varphi_i(x)$ if (ii) $\varphi_i(x)$ is defined and
(iii) $h(x, \Phi_i(x)) < \Phi_{g(j)}(x)$
undefined otherwise

5 *is satisfied.* Suppose $\varphi_i(x)$ is defined, but that (to the contrary) $h(x, \Phi_i(x)) \geqq \Phi_{g(j)}(x)$. Then $r(n, x, j)$ is undefined. Hence $\varphi_j(x) \neq \varphi_{g(j)}(x)$ since $\Phi_{g(j)}(x) \leqq h(x, \Phi_i(x)) < \infty$. This contradicts 2: $\varphi_{g(j)} = \varphi_j$.

3 *is satisfied.* If $\varphi_i(x)$ is undefined, then $\varphi_j(x)$ is undefined (Eq. 3), so $\varphi_{g(j)}(x) = \varphi_j(x) = \varphi_i(x)$. If $\varphi_i(x)$ is defined, then the proof of 5 shows that $h(x, \Phi_i(x)) < \Phi_{g(j)}(x)$, so (ii), (iii) in Eq. 3 are satisfied and therefore $\varphi_{g(j)}(x) = \varphi_i(x)$. Q.E.D.

Any algorithmic function g for reducing the size of machines can thus be effectively replaced by another function g' that further reduces the size and number of steps taken by infinitely many machines.

We note that theorem 3 is curiously symmetric with respect to size 4 and steps 5.

Acknowledgments

I am especially grateful to my wife, Lenore, for reading the manuscript and shortening the proofs, and to the referee for his most helpful suggestions.

Received: November 29, 1966

References

Arbib, M. (1966), Speed-up theorems and incompleteness theorems in automata theory. *In* "Automata Theory" (E. R. Caianiello, ed.) pp. 6–24. Academic Press, New York.

ARBIB, M., AND BLUM, M. (1965), Machine dependence of degrees of difficulty, *Proc. Am. Math. Soc. XVI*, No. 3 (June), 442-447.

BLUM, M. (1967), A machine-independent theory of the complexity of recursive functions, *J. Assoc. Comp. Mach., XIV*, No. 2 (April 1967) 322-336.

CHAITIN, G. (1966), On the length of programs for computing finite binary sequences, *J. Assoc. Comp. Mach., XIII*, No. 4 (Oct.), 547-569.

CHAITIN, G., On the simplicity and speed of programs for computing infinite sets of natural numbers, *J. Assoc. Comp. Mach.*, in press.

DAVIS, M. (1958), "Computability and Unsolvability," McGraw-Hill, New York.

HARTMANIS, J., AND STEARNS, R. (1965), On the computational complexity of algorithms, *Trans. Amer. Math. Soc., CXVII*, No. 5 (May), 285-306.

MINSKY, M. (1962), Size and structure of universal turing machines using tag systems *In* "Recursive Function Theory, Proceedings of the Symposia in Pure Mathematics" (Amer. Math. Soc. Pub.). vol. V. 229-237.

ROGERS, H., JR. (1958), Gödel numberings of partial recursive functions, *J. Symbol. Logic, XXIII*, No. 3 (Sept.), 331-341.

ROGERS, H., JR. (1968), Recursive functions and effective computability, McGraw-Hill, New York, in press.