A NEW HIERARCHY OF ELEMENTARY FUNCTIONS

G. T. HERMAN[1]

0. **Introduction.** Let C be the class of all functions on nonnegative integers into nonnegative integers for which there is a mechanical procedure for obtaining the values from the arguments. Such functions are called computable. C will contain functions whose computations are of very different complexity. The search for a good measure of the computational complexity of a function has been studied recently by several logicians, computer scientists and even statisticians.

One method of obtaining such a measure is the method of hierarchies. This method consists of giving a sequence $H_0, H_1, H_2, \cdots$, of classes of computable functions, such that, for all $i \geqq 0$, $H_i \subset H_{i+1}$ and we have some reason to believe that a function which belongs to H_{i+1} but not to H_i is of a greater computational complexity than any function in H_i. Then a measure of the computational complexity of a function f can be taken to be the least i such that $f \in H_i$. (Such hierarchies have been studied by Axt [1], Cleave [2], Grzegorczyk [4] and Ritchie [8].)

Ritchie [8] provides a hierarchy $F_0, F_1, F_2, \cdots$. F_0 is taken to be the class of all functions computable by a finite automaton and, for all $i \geqq 0$, F_{i+1} is defined to be the class of functions computable by Turing machines which use in their computations an amount of tape bounded by a function in F_i. It is shown by Ritchie [8, Theorem 3, p. 148] that $\bigcup_{i=0}^{\infty} F_i$ is exactly the class of elementary functions in the sense of Kalmar (i.e. the class $\mathcal{E}^3$ in Grzegorczyk [4]). Since this class contains most simple arithmetic functions the hierarchy is of considerable interest. (See §4 for an application.)

This paper contains two main results.

The first (Theorem 1) is a characterization of Ritchie's classes in a new and neat way. This usually allows us to locate an elementary function of interest in the hierarchy more simply and efficiently than it could be done using the results of Ritchie [8].

Secondly, for various reasons (some of which are explained on p. 143 especially in footnote 8 in [8]), Ritchie encodes his functions in binary notation on the Turing machines. However, encoding in the literature is usually unary. In order to make use of this available information, it is desirable to have a hierarchy of elementary func-

Received by the editors June 29, 1967 and, in revised form, December 10, 1967.

[1] Now at IBM (UK) Ltd., London.

tions based on unary encoding. We define such a hierarchy G_1, G_2, G_3, $\cdots$. We show that, for all $i \geqq 0$, $F_i \subset G_{i+1} \subset F_{i+1}$, where all the inclusions are proper. Hence this new hierarchy, which uses unary encoding, is as good in reflecting computational complexity as Ritchie's hierarchy which is based on binary encoding.

1. **The Ritchie hierarchy.** We assume familiarity with Appendix 2 of Ritchie [8]. We shall use the words Turing machine and finite automaton as they are defined there, except that we extend Definition 7 [8, p. 162]. When we speak of a Turing machine T computing a function f we do not assume (unless otherwise stated) that the alphabet A of T is (*, 0, 1), but only that (*, 0, 1) $\subset A$. (We use * to denote the blank symbol.)

DEFINITION 1. Let f be an n-ary computable (total) function and let T be a Turing machine which computes it. We define the n-ary function $a_{f,T}$ by $a_{f,T}(\mathbf{x})$ = the maximum amount of tape used by T while computing $f(\mathbf{x})$ (notation $\mathbf{x} = (x_1, \cdots, x_n)$).

DEFINITION 2. F_0 is the set of functions computable by a finite automaton. For any $i \geqq 0$, F_{i+1} is the set of functions f for which there exists a Turing machine T over the alphabet (*, 0, 1) which computes f and is such that $a_{f,T}(\mathbf{x}) \leqq g(\mathbf{x})$ for some functions g in F_i.

These are the classes F_i as defined in Ritchie [8].

LEMMA 1. *If a function f is computable by a Turing machine T, then there exists an integer K and a Turing machine Z over the alphabet* (*, 0, 1) *such that Z computes f and for all* $\mathbf{x}$, $a_{f,Z}(\mathbf{x}) \leqq K \cdot a_{f,T}(\mathbf{x})$.

PROOF. This is immediate from the construction by Shannon [9, pp. 163–165].

Using this lemma we can now easily show that for all $i \geqq 0$, $F_i \subset F_{i+1}$. This is not proved by Ritchie [8], but it is implicitly assumed in the proofs of some of the results we shall quote from that paper.

Another consequence of the lemma is that if we forego in Definition 2 the requirement that T is over the alphabet (*, 0, 1) and allow T to be any Turing machine whatsoever, the classes F_i defined by the definition will not change. (The proof of this is by induction on i, using the fact that if g and h are functions such that $g \in F_i$ and $h(\mathbf{x}) = K \cdot g(\mathbf{x})$ for a fixed constant K and all $\mathbf{x}$, then $h \in F_i$ also. See proof of Lemma 2 in Ritchie [8, pp. 144–145].)

2. **A simple characterization of the F_i.**

DEFINITION 3. For all $i \geqq 0$, we define a unary function f_i by

$$f_0(x) = x, \qquad f_{i+1}(x) = 2^{f_i(x)}.$$

LEMMA 2. *For any $i \geqq 1$ and for any integer K,*

$$f_i(K \cdot \max(\mathbf{x}, 1)) \in F_i.$$

PROOF. By induction on i.

The case $i=1$ can easily be shown, since $2^{K \cdot \max(\mathbf{x},1)}$ can be computed by a Turing machine using binary notation without using more tape than some fixed multiple L of $\sum_{i=1}^{\infty} x_i + 1$.

Suppose $f_i(K \cdot \max(\mathbf{x}, 1)) \in F_i$. Then

$$f_{i+1}(K \cdot \max(\mathbf{x}, 1)) = 2^{f_i(K \cdot \max(\mathbf{x},1))} \in F_{i+1}.$$

This follows from Lemma 2 of Ritchie [8, p. 144], with $g(\mathbf{x}) = f_i(K \cdot \max(\mathbf{x}, 1)) \in F_i$ and $h(x) = 2^x \in F_1$.

THEOREM 1. *Let $i \geqq 1$ and f be any numerical function. Then the following two conditions are equivalent.*

(i) $f \in F_i$.

(ii) *There exists a Turing machine $\mathbf{T}$ which computes f and an integer K such that*

$$a_{f,\mathbf{T}}(\mathbf{x}) \in f_{i-1}(K \cdot \max(\mathbf{x}, 1)).$$

PROOF. (i)$\rightarrow$(ii). Proved by Ritchie [8, p. 145, Lemma 3]. (Due to a misprint, the lemma is stated for $i>1$ only, but the proof deals with $i=1$ as well.)

(ii)$\rightarrow$(i). For $i>1$, this is immediate from Lemma 2 and the remark at the end of §1. For $i=1$ we note that $\max(\mathbf{x}, 1) \leqq \sum_{i=1}^{n} x_i + 1$, and so

$$a_{f,\mathbf{T}}(\mathbf{x}) \leqq f_0(K \cdot \max(\mathbf{x}, 1)) \leqq K \cdot \left(\sum_{i=1}^{n} x_i + 1 \right),$$

which is a function in F_0. Hence, by the remark at the end of §1, $f \in F_1$.

It is a consequence of this theorem that the function of f_i is not an element of F_{i-1} (for $i \geqq 2$), because the amount of tape required for expressing the value of the function increases too fast to be expressible on a tape of length $f_{i-2}(K \cdot \max(\mathbf{x}, 1))$. The question arises whether it is always the value of the function f which determines the least i such that $f \in F_i$. In other words, can we replace in Theorem 1

$$a_{f,\mathbf{T}}(\mathbf{x}) \leqq f_{i-1}(K \cdot \max(\mathbf{x}, 1))$$

by

$$f(\mathbf{x}) \leqq f_i(K \cdot \max(\mathbf{x}, 1)).$$

The answer is that we cannot. Ritchie [8, pp. 159–160] produces, for each $i>0$, a function Φ_i, such that $\Phi_i \in F_{i+1}$, $\Phi_i \notin F_i$ and the value of Φ_i for any $\mathbf{x}$ is either 0 or 1.

3. A new hierarchy using unary encoding.

DEFINITION 4. A Turing machine T will be said to *unary compute* the function f from all n-tuples of nonnegative integers to nonnegative integers if it computes the function f': $D \rightarrow T_{(1)}$, where

(i) D is the set of all strings

$$1^{a_1+1} * 1^{a_2+1} * \cdots * 1^{a_n+1}$$

for $(a_1, a_2, \cdots, a_n)$ an n-tuple; and

(ii) $f'(1^{a_1+1} * 1^{a_2+1} * \cdots * 1^{a_n+1})$ is defined to be

$$1^{f(a_1, a_2, \cdots, a_n)+1}.$$

DEFINITION 5. Let $b_{f,T}$ be defined for unary computations in the same way as $a_{f,T}$ was defined for computations (Definition 1).

DEFINITION 6. For each $i \geqq 1$, G_i is the set of functions f for which there exists an integer K and a Turing machine T which unary computes f, such that

$$b_{f,T}(\mathbf{x}) \leqq f_{i-1}(K \cdot \max(\mathbf{x}, 1)).$$

THEOREM 2. *For each* $i>0$, $F_i \subset G_{i+1} \subset F_{i+1}$.

PROOF. Suppose $f \in F_i$. We consider the Turing machine T which unary computes f in the following way: First it changes the initial tape into one which represents the same n-tuple in binary notation. Then it operates the same way as the finite automaton (if $i=0$) or Turing machine (if $i>0$) which computes f showing that $f \in F_i$. Finally it retranslates the result into unary notation. Since the length of the binary encoding of $f(\mathbf{x})$ is $\leqq k+l(t)$ (if $i=0$, by Theorem 1 of Ritchie [8, p. 164]) where $l(t)$ is the length of the binary encoding of $\max(\mathbf{x}, 1)$ and k is a fixed constant, or it is $\leqq f_{i-1}(L \cdot \max(\mathbf{x}, 1))$ (if $i>0$, by Theorem 1 above) where L is a fixed constant, the length of the unary decoded answer is $\leqq f_i(K \cdot \max(\mathbf{x}, 1))$. By choosing K suitably large we can carry out the whole computation within this length of tape and so, by Definition 6, $f \in G_{i+1}$.

Now suppose $f \in G_{i+1}$. The method or reencoding from binary to unary, computing in unary and decoding into binary again, shows that $f \in F_{i+1}$. (Definition 6 and Theorem 1 have to be used.)

COROLLARY 1. *Let* $G = \bigcup_{i=1}^{\infty} G_i$. *$G$ is precisely the class of elementary functions.*

THEOREM 3. *Each inclusion in Theorem 2 is proper.*

PROOF. $f_{i+1} \in F_{i+1}$ but it is not in G_{i+1}. (See remarks at the end of §2.)

$\max(x, 1) \in G_1$, but it is not computable by a finite automaton.

For $i > 0$, the function Φ_i mentioned at the end of §2 is such that $\Phi_i \in G_{i+1}$, but $\Phi_i \notin F_i$. ($\Phi_i \in G_{i+1}$ because the reencoding of the argument into binary notation, the carrying out of binary computation of Φ_i and decoding of the answer, which is 0 or 1, will not need essentially more tape than just the binary computation.)

THEOREM 4. *For any class H of functions let $(H)_*$ be the class of relations whose characteristic function is in H. Then, for all $i \geqq 1$, $(F_i)_* = (G_i)_*$.*

PROOF. This follows from Theorem 2 and the final remarks in the proof of Theorem 3.

4. Further remarks. Hierarchies of elementary functions which reflect the complexity of their computation can be used in a variety of situations. One such comes about if we adopt a definition like that in Fischer [3, Appendix] for the simulation of one Turing machine by another using the elementary functions as the basic class. In this case the index of the smallest class F_i (or G_i) to which the functions of that definition belong can be considered to be the measure of the "difficulty of encoding" of the simulation. (Such investigations are carried out in Herman [6].)

Using the computations described in detail in Herman [5], one can immediately locate the G_i's in which the functions discussed below belong. Note that in each case the G_i given is the smallest possible, since the value of the function for a given argument already needs enough tape to force the function to be in the given class. $x+y$, $x-y$, (x/y), $\mathrm{rem}(x, y)$, $(x)^{1/n}$ (n fixed) are all in G_1, $x \cdot y$, x^n (n fixed) p_x are in G_2, $x!$ is in G_3.

Cleave [2] also considers a hierarchy of elementary functions. This is defined in terms of the number of jumps needed to perform the computation of a function on a J-limited machine (a version of the unlimited register machine described by Shepherdson & Sturgis [10]). In spite of this very different approach, it can be shown (see Herman [7]) that both Cleave's hierarchy and another closely related hierarchy gives essentially the same classification of elementary functions as the hierarchies F_i or G_i. This indicates a certain stability of these classifications.

Acknowledgement. I wish to thank Professor W. W. Boone and the referee for their helpful criticisms of an earlier version of this paper.

References

1. P. Axt, *Enumeration and the Grzegorczyk hierarchy*, Z. Math. Logik Grundlagen Math. 9 (1963), 53–65.

2. J. P. Cleave, *A hierarchy of primitive recursive functions*, Z. Math. Logik Grundlagen Math. 9 (1963), 331–345.

3. P. C. Fischer, *On formalisms for Turing machines*, J. Assoc. Comput. Mach. 12 (1965), 570–580.

4. A. Grzegorczyk, *Some classes of recursive functions*, Rozprowy Matematyczne 4 (1953), 1–44.

5. G. T. Herman, *Turing machines and their applications to Hilbert's tenth problem*, M.Sc. Thesis, University of London, 1964.

6. ———, *A definition of simulation for discrete computing systems*, Third Internat. Congr. for Logic, Methodology and Philosophy of Science, Amsterdam, 1967.

7. ———, *On hierarchies of elementary functions*, Proc. Internat. Colloq. Recursive Functions and their Applications (Tihany, 1967), Institut Blaise Pascal, Paris and Bolyai Janos Matematikai Tarsulat, Budapest, (in press).

8. R. W. Ritchie, *Classes of predictably computable functions*, Trans. Amer. Math. Soc. 106 (1963), 139–173.

9. C. E. Shannon, *A universal Turing machine with two internal states*, Automata Studies, Ann. of Math Studies No. 34, Princeton Univ. Press, Princeton, N. J., 1956, pp. 157–165.

10. J. C. Shepherdson and H. E. Sturgis, *Computability of recursive functions*, J. Assoc. Comput. Mach. 10 (1963), 217–255.

Brighton College of Technology, England