This blank page was inserted to preserve pagination.

MAC TR-110

COMPLEXITY CLASSES OF RECURSIVE FUNCTIONS

Robert Moll

June 1973

This research was supported in part by the National Science Foundation under research grant GJ-34671, and in part by the Advanced Research Projects Agency of the Department of Defense under ARPA Order No. 433 which was monitored by ONR Contract No. N00014-70-A-0362-0001.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

CAMBRIDGE MASSACHUSETTS 02139

Abstract

An honest function is one whose size honestly reflects its computation time. In 1969 Meyer and McCreight proved the "honesty theorem", which says that for every t, the t-computable functions are the same as the t'-computable functions for some honest t'.

Ways of constructing honest functions are considered in detail. It is shown that for any t there is an honest t' such that the t-computable functions and the t'-computable functions are the same, and such that t' is arbitrarily large on a dense set of arguments. Moreover any construction method satisfying certain natural criteria will (almost) have this property.

On the other hand it is shown that by relaxing these criteria we can guarantee that $t' \leq t$ on a (weak) dense set. We can also guarantee that t' will be bounded above by a predetermined recursive function on all but finitely many arguments. Finally, we show that in the case where t is monotone, t' can also be made monotone.

We consider the t-computable functions, and order these classes under set inclusion as t varies over the recursive functions. We show that given any total effective operator $\underset{\sim}{F}$ and any recursive countable partial order R there is an r.e. sequence of machine running times $T_0, T_1, \cdots T_n, \cdots$ such that if iRj, then the T_j computable functions properly contain the $\underset{\sim}{F}(T_i)$ computable functions, and if i and j are incomparable, then $\underset{\sim}{F}(T_i) > T_j$ infinitely often and $\underset{\sim}{F}(T_j) > T_i$ infinitely often.

THESIS SUPERVISOR: Albert R. Meyer
TITLE: Associate Professor of Electrical Engineering

Acknowledgements

I would like to thank M.I.T. Project MAC for supporting me during the writing of this thesis.

I am grateful to the following people for their support during my formative years as a student of mathematics: R.A. Moore, Ed Fischer, Tom and Judith Wasow, Assaf Kfoury, and the Lee Street Commune.

I am especially grateful to Professor Albert R. Meyer for his professional and moral support, and above all his patience, during the preparation of this thesis.

Finally, I would like to thank Sharyn Cohn for typing the thesis.

TABLE OF CONTENTS

Preface

The three chapters of this thesis can be read independently. Chapters two and three are entirely self-contained; no attempt has been made to integrate them into a single document. Chapter two has been accepted for publication by the Journal of Symbolic Logic. It is co-authored by Albert R. Meyer. Chapter three has been submitted for publication to the Journal of Computer and System Sciences.

Chapter 1

A Survey of Work on Subrecursive Hierarchies and Subrecursive Degrees

The definition of the partial recursive functions is easily describable, involving merely the μ-operator in addition to the traditional initial functions and schemas for developing the primitive recursive functions. Moreover the Kleene normal form theorem gives an effective syntactic presentation of these functions. The recursive functions, those partial recursive functions which are total, has no such presentation. In general the demonstration that a partial recursive function is total involves a non-constructive existence proof.

To avoid this difficulty, subrecursive hierarchies have been constructed in an attempt to effectively approximate the class of recursive functions.

A subrecursive hierarchy is a sequence of classes of recursive functions $P_0, P_1, \cdots, P_\alpha, \cdots, P_\beta, \cdots$, where α and β may be finite or infinite ordinals. For $\alpha < \beta$, $P_\alpha \subsetneq P_\beta$, and the extension of a hierarchy from α to $\alpha+1$, or from $\{\alpha_n\}_{n\in N}$ to α (where $\lim_n \alpha_n = \alpha$ and α is limit ordinal) is usually carried out by some uniform effective principle.

The method of hierarchies has also been applied to certain rich and interesting subclasses of the recursive functions. The goal of such hierarchies is to approximate the given class from below with smaller, more comprehensible sets of functions. Hopefully such a construction will provide insight into the structure and complexity of the given class.

We begin by studying ω-length hierarchies of the primitive recursive functions. We show that these hierarchies are quite successful in that they give non-trivial alternative formulations of the primitive recursive functions. Moreover there is considerable agreement among the various hierarchies, and this agreement may be interpreted to mean that various notions of primitive recursive complexity coincide.

Similar results are obtained for ω-hierarchies of the elementary functions.

Next we consider various attempts to build hierarchies of transfinite length which exhaust the recursive functions. We discuss at length the issue of names for ordinals. Ordinal names must be used to index any transfinite hierarchy, and we show how problems with ordinal names has essentially ruled out any hope of building a meaningful exhaustive hierarchy of the recursive functions.

The difficulties with building exhaustive hierarchies has led investigators to construct and study "short" transfinite hierarchies which exhaust only a portion of the recursive functions. A key issue for such constructions is the selection of "nice" ordinal names to index such hierarchies, and this has been done with considerable success, at least for hierarchies of length less than or equal to ϵ_0.

Finally, we consider subrecursive degrees, corresponding to Turing degrees of full recursion theory. This recently revitalized area has begun to distinguish itself from the theory of Turing degrees, and has established some interesting structural results about subrecursive behavior.

Notations and Definitions

For basic notation from recursive function theory, we follow Rogers [2].

We denote by $< x,y >$ a 1-1 onto recursive map from $N \times N \to N$. Associated with $< >$ are decoding functions π_1, π_2, such that $z = < \pi_1(z), \pi_2(z) >$.

Let f be any function. Define $f^{(1)}(x) = f(x)$, $f^{n+1}(x) = f(f^n(x))$. $f^{(n)}$ is the nth-power of f.

If t is any total function, then the t-computable functions are the set of functions computable within t(x) Turing machine steps, for all but finitely many arguments. Our Turing machine conventions are those of Davis [ ].

If $f(x_0, \cdots, x_n) = h(g_0(x_0, \cdots, x_{k_0}), \cdots, g_m(x_0, \cdots, x_{k_n}))$ we say that f is defined from h, g_0, $\cdots$, g_m by composition.

If $f(0, x_1, \cdots, x_n) = g(x_1, \cdots, x_n)$,

$$f(n+1, x_1, \cdots, x_n) = h(f(n, x_1, \cdots, x_n), n, x_1, \cdots, x_n),$$

then we say that f is defined from g and h by primitive recursion.

The class of primitive recursive functions is the smallest class of functions containing the zero function, the successor function, and the projection functions $U_i^n(x_1, \cdots, x_n) = x_i$, which is closed under composition and primitive recursion.

If g, h, and j belong to some class of functions and f satisfies the equations

$$f(0,y) = g(y),$$

$$f(x+1, y) = h(x, y, f(x,y))$$

$$f(x,y) \leq j(x,y),$$

then we say that f is defined by limited recursion from g, h, and j.

If $f(n, x, \cdots x_n) = \prod_{z=0}^{n} g(z, x_1, \cdots, x_k)$, we say f is defined from g by limited multiplication. There is a similar scheme for limited summation.

If $f(x_1, \cdots, x_k) = g(y_1, \cdots, y_n)$, where each y_i equals some x_i, then we say that f is defined from g by explicit transformation.

The class of elementary functions E of Kalmar [77] can be defined as the smallest class containing x+y, x•y, and x^y which is closed under the operations of composition, explicit transformations, and limited recursion.

We use $\overline{x_k}$ as an abbreviation for the expression $x_1, x_2, \cdots, x_k$.

Section 1. ω-hierarchies of Primitive Recursive Functions

The primitive recursive functions have been the most widely studied subrecursive class, and so it is natural that much of the work on hierarchies of recursive functions has centered around classifying these functions. An ω-hierarchy of primitive recursive functions is an increasing sequence of classes of functions $P_0, P_1, \cdots, P_k, \cdots$, such that for each k, $P_k \subsetneq P_{k+1}$ and such that the union of these classes equals the primitive recursive functions. If f is primitive recursive, then the least k such that $f \in P_k$ in some sense measures the difficulty of f. As we shall see there are many different ways to formulate hierarchies of primitive recursive functions, each with its associated concept of difficulty; however, there is a high degree of invariance among these concepts, and this invariance makes the primitive recursive functions a well understood subrecursive class.

Primitive recursive hierarchies have been formulated in several different ways. One approach is to consider each class in the hierarchy as a closure class. Each P_k is formed by the application of certain sub-primitive recursive closure rules to certain initial functions, usually differing only in a single "key" function f_k. This is the approach of Gzregorzyk, and Axt. . Another formulation constructs each class using some external syntactic criterion; for example, one might assign f to P_k if f can be defined using at most n nested instances of primitive recursion. Axt did the initial work in this direction. Yet another approach, proposed by Robert Ritchie, Robbin, Cobham, and Meyer and Ritchie, is complexity-theoretic in nature. $f \in P_k$ in case f is (roughly) f_k computable.

The fundamental result of this section is that all these approaches yield essentially the same hierarchy.

Gzregorczyk in his 1953 paper [41] gives the first formulation of an ω-hierarchy of primitive recursive functions. He defines a sequence of rapidly increasing recursive functions f_n, and each f_n is used to define the n^{th} class in the sequence.

Definition: Define a sequence of functions $f_n \in \mathcal{R}_2$ as follows:

1. $f_0(x,y) = y+1$
2. $f_1(x,y) = x+y$
3. $f_2(x,y) = (x+1)\cdot(y+1)$
4. $f_{n+1}(0,y) = f_n(y+1, y+1)$
5. $f_{n+1}(x+1,y) = f_{n+1}(x, f_{n+1}(x,y))$

He defines his sequence of classes of primitive recursive functions $E^0, E^1, \cdots, E^k, \cdots$, as follows.

Definition: Let E^n be the smallest class of functions containing as initial functions the successor function, the projection functions, and f_n, and closed under the operations of composition, explicit transformation, and limited recursion.

Notice that f_3 is essentially exponentiation, and so E^3 is the elementary functions.

An essential feature of any proposed hierarchy is a hierarchy theorem, that is, a theorem which demonstrates that the classes of the hierarchy form a proper increasing chain.

Theorem: For all $n \geq 0$ $E^n \subsetneq E^{n+1}$.

Gzregorczyk's proof of this theorem is complicated by his choice of key functions f_k. The difficulty in the proof arises because f_{n+1} is not defined by a simple primitive recursive scheme, and so a bounded recursion argument by itself will not suffice to establish the result. Gzregorczyk uses a fairly intricate coding argument to show that for $i < n$, $f_i \in E^n$; this shows that $E^n \subset E^{n+1}$ for each n.

The proof that each containment is proper follows from the fact that for each n, $f_{n+1}(x,x)$ majorizes the one-variable functions of E^n.

By first observing that each f_n is primitive recursive, it is immediate that $\bigcup_n E^n \subseteq$ primitive recursive functions. The next result shows that this containment is actually an equality.

Theorem: $\bigcup_n E^n$ = primitive recursive functions.

To prove this Gzregorczyk uses a formulation of the primitive recursive functions due to R.M. Robinson [78]. The important feature of this formulation is that primitive recursion is eliminated and is replaced by a schema for iteration. It is then not difficult to show by induction on the order of a function (where the order of f counts the number of operations used in the definition of f) that if f is primitive recursive and has order k, then $f \in E^{k+3}$.

In an early paper Cobham [79], drawing on work of Ritchie [66], considers the Gzregorczyk hierarchy and observes that the classes E^k have interesting complexity-theoretic properties.

Theorem: For $k \geq 3$ $f \in E^k$ iff some program P computes f and $T_p \in E^k$, where T_p is the run-time of P iff there is a $g \in E^k$ such that f is g-computable.

Cobham states his result for $k \geq 3$ to achieve machine-independence; in this form the theorem is true for any device or programming language which can be arithmetized in an elementary-recursive way.

Meyer and Ritchie [72] exploit this result to give a complexity-theoretic formulation of the Gzregorczyk hierarchy. We develop the Meyer-Ritchie approach here because the ideas involved will be useful in proving the equivalence of various different hierarchies.

Definition: Given any f, let E(f), the functions elementary in f, be the

smallest set containing x^y, $x+y$, $x \cdot y$, and f, which is closed under composition, explicit transformation, and limited recursion.

Notice that for $k \geq 3$ $E(f_k) = E^k$, where f_k is the k^{th} Gzregorczyk function.

The following simple theorem proves one part of Cobham's result cited above.

Theorem: Let g be any function computable within t(x) steps for each argument x. If $t \in E(f)$, then $g \in E(f)$.

The proof of this theorem rests on the fact that in any reasonable machine model there exist elementary functions $0_m(e, \vec{x}_m, y)$ for $m \geq 1$ such that $0_m(e, \vec{x}_m, y)$ = the output of the e^{th} Turing machine on arguments $\vec{x}_m$, if the machine halts within y steps, and 0 otherwise. For every f, $0_m \in E(f)$, and so $0_m(e_g, \vec{x}_m, t(\vec{x}_m)) = g \in E(f)$, where e_g is a machine which computes g in time t.

Call a function f elementary-honest if f is h-honest for some elementary-recursive h.† The next result is a partial converse to the last theorem.

Theorem: If f is elementary-honest and if $g \in E(f)$, then there is a $t \in E(f)$ such that g is t-computable.

Summarizing these last two results we have that if f is elementary honest, then $g \in E(f)$ if and only if g is t-computable for some $t \in E(f)$. Classes with this property are called computation-time closed classes.

---

†See definition 1 of Chapter 2, Section 2.

The size of functions in E^k for each k plays an important role in Gzregorczyk's work. The following simple bounding lemma of Meyer and Ritchie will yield more precise information on function size for Gzregorczyk's classes.

Bounding lemma: If f is non-decreasing and $\geq 2^x$ and if $g \in E(f)$, then there is a constant c such that $g(\vec{x}_n) \leq f^{(c)}(\max[x_1, \cdots, x_n])$.

Meyer and Ritchie are now in a position to redefine the Gzregorczyk classes. First they note that $f \circ g$ is elementary honest if f and g are, and f is at least as large as the identity. Similarly if f is elementary honest and non-decreasing, then $f^{(c)}$ is elementary honest. Using these observations they construct a Gzregorczyk-like sequence of elementary honest functions g_n, (based on a modification of Gzregorczyk's functions due to Ritchie [67]), as follows:

$$g_3 = 2^x$$

$$g_{n+1}(x) = g_n^{(x)}(1).$$

These simple functions can be used instead of Gzregorczyk's functions f_k, so that for $n \geq 3$, $E^n = E(g_n)$. Thus for $n \geq 3$ the class E^n is precisely the set of functions which are computable within time bounded by some fixed iterate of g_n.

Gzregorczyk has one other result of interest, and this result leads very naturally to another formulation of a primitive recursive hierarchy.

Definition: Let E be a class of functions. A function F(x,y) is a universal function for E if for each x, $\lambda y\ F(x,y) \in E$, and for every $g \in E$ there is an x such that $g = \lambda y\ F(x,y)$.

Theorem: For $n \geq 2$ E^{n+1} contains a universal function for the one-variable functions in E^n.

An important feature of any hierarchy is the method used for class enlargement, the "jump operation" of the hierarchy. One of the weaknesses of Gzregorczyk's formulation is the obscurity of his jump operation, and the resulting relative difficulty of his hierarchy theorem.

Axt [13] proposes an ω-hierarchy where the jump operation is based explicitly on universal functions. To go from class to P_k to class P_{k+1} Axt adds to the initial functions of P_k an enumerating function for P_k.

Definition: Let θ be a set of total functions. We say a function ϕ is E^4 in θ if ϕ belongs to the smallest class containing θ and f_4(the Gzregorczyk function f_4), which is closed under composition, explicit transformation, and limited recursion.

Axt now chooses a particular E^4 function ef(x,y) such that $ef^{\theta}(x,y)$ is a universal function for the set of functions E^4 in θ.

Definition: Define e_n for $n \geq 0$ as follows:

$$e_0(x,y) \equiv 0$$

$$e_{n+1}(x,y) = ef^{e_n}(x,y)$$

Axt now defines classes E_n based on the enumerating functions e_n; E_n will be the set of functions E^4 in e_n.

Theorem (Axt Hierarchy Theorem): $E_0 \subsetneq E_1 \subsetneq \cdots \subsetneq E_n$.

Notice that the proof of this theorem is immediate, since E_{n+1} contains the initial functions for E_n, and by a trivial diagonization, $e_{n+1} \notin E_n$.

Axt is able to show his hierarchy is essentially the same as Gzregorczyk's.

Theorem: For all $n \geq 0$ $E^{n+4} = E_n$.

This is a pleasing result. It gives us a surprising alternative formulation of the Gzregorczyk hierarchy. However, Axt's result is less significant than one might suspect. The difficulty with his work lies with his choice of the universal function ef. It is not hard to show that there are a great many possible universal functions ef, each as natural as Axt's ef, and each yielding a different hierarchy when used as a jump procedure to construct an Axt-like hierarchy. Indeed, the significance of his technique is the highly non-invariant character of his jump operation. This phenomenon is in sharp contrast to the situation in full recursion theory, where the behavior of the jump operation on Turing degrees does not depend on the specific details of the jump definition.

Axt formulates another ω-hierarchy in [14] based on a natural syntactic criterion, depth of nesting of primitive recursion. He defines his classes K_n as follows.

Definition:

(i) if f is an initial function, that is if f is the zero function, the successor function, or a projection function, then $f \in K_0$.

(ii) if f is defined by composition from h, $g_1, \cdots, g_k$, and $h \in K_{i_0}$, $g_1 \in K_{i_1}, \cdots, g_k \in K_{i_k}$, then $f \in K_{\max[i_j \mid 0 \leq j \leq k]}$.

(iii) if f comes from g and h by primitive recursion and $g \in K_{n_1}$, and $h \in K_{n_2}$, then $f \in K_{1+\max[n_1, n_2]}$.

It is immediate that $K_0 \subsetneq K_1 \cdots \subsetneq K_n \subsetneq \cdots$, and the $\bigcup_n K_n$ = primitive recursive functions.

This hierarchy uses no external machinery in its definition, and in this sense is perhaps the most naturally formulated hierarchy of any we have considered. It turns out that for $n \geq 3$ $E^{n+1} = K_n$[†], although Axt was unable to show this in his original study.

Meyer [80] was the first to show that Axt's depth of nesting hierarchy and the Gzregorczyk hierarchy eventually coincide. He shows that for $n \geq 9$, $E^{n+1} = K_n$. The best published result to date is due to Schwichtenberg [51], who proves that $E^{n+1} = K_n$ for $n \geq 3$. Meyer's proof rests on the complexity-theoretic properties of the Gzregorczyk hierarchy which we established earlier.

He begins by proving that for $n \geq 3$ $K_n \subset E^{n+1}$, using an inductive argument to show that every function in one class is majorized by some function in the other class. This yields the result, since if f appears in K_n by an instance of primitive recursion, then $g > f$ for some $g \in E^{n+1}$,

---

[†]Muller has announced the result for $n \geq 2$ in the Recursive Function Theory Newsletter, No. 5, April 1973.

and so $f \in E^{n+1}$ by an instance of limited recursion with g as bounding function.

The proof that $E^n \subset K_{n-1}$ for sufficiently large n is dealt with using explicit complexity-theoretic arguments. Since $\bigcup_{n\in N} K_n$ exhausts the primitive recursive functions, $O_m(e, \overrightarrow{x_m}, y)$ belongs to K_{n_0-1} for some n_0. Moreover, since the running time of f is bounded by some function in E^n, and hence, by the above, by some function $t \in K_{n_0-1}$, Meyer concludes that $f = O_m(e_f, \overrightarrow{x_m}, t(\overrightarrow{x_m})) \in K_{n_0-1}$, where e_f is a Godel number for f. He shows that $n_0 \leq 9$, and thus for sufficiently large n, $K_n = E^{n+1}$.

This is a rather striking result in that it relates the size f, the running time of f, and the syntactic form of f. The same general method yields Schwichtenberg's result, although the details of the construction of O_m in K_2 is much more difficult.

Several investigators have considered syntactically formulated hierarchies which are quite similar to the depth of nesting hierarchy. Parsons [63] observes that iteration is the feature of primitive recursion that increases functional complexity. Using this as a guide, he defines a hierarchy based on nested iteration rather than nested primitive recursion. With this phenomenon in mind, he builds his classes $\mathfrak{L}^p$ so that functions defined by primitive recursion are placed in this class only when p nested iterations takes place. He shows that for $p \geq 2$ $\mathfrak{L}^p = E^{p+1}$.

Schwichtenberg, [51] and Meyer and Ritchie [58] also build hierarchies similar to the depth of nesting hierarchy. They place $f \in K_n^{sim}$ in case f is defined from functions in K_{n-1}^{sim} by an instance of simultaneous recursion. They show that for $n \geq 2$ $K_n^{sim} = E^n$.

Meyer and Ritchie [59] propose yet another syntactically formulated hierarchy. They consider a simplified programming language, and they measure program difficulty by depth of nesting of LOOP-END pairs. Their language consists of five possible types of expressions, (1) Set X to X+1, (2) Set X to Y, (3) Set X to zero, (4) LOOP, and (5) END. A sequence of instructions is a Loop program when LOOP and END instructions are matched like left and right parentheses. LOOP-END pairs affect the normal sequential flow of the program. If P is a Loop program, and register X contains integer x, then "LOOP X, P, END" means that program P is to be executed x times before the next instruction (if any) after the END is executed.

A program hierarchy L_n is constructed by placing program P in L_n if P includes LOOP-END pairs nested to depth at most n. A hierarchy of functions $\mathcal{L}_n$ for $n \geq 0$ is now derived from the L_n hierarchy: $f \in \mathcal{L}_n$ in case some $P \in L_n$ computes f.

Since loop structure and the schema of primitive recursion are very similar, a routine inductive argument shows that $\bigcup_n \mathcal{L}_n$ = primitive recursive functions. Moreover, Meyer and Ritchie are able to relate their Loop hierarchy to Gzregorczyk's by linking the classes $\mathcal{L}_n$ directly to the modified Gzregorczyk functions g_n. They define a sequence of functions h_n as follows:

$$h_0 = \begin{cases} x+1 & \text{if } x = 0 \\ x+2 & \text{if } x \geq 2, \end{cases}$$

$$h_{n+1}(x) = h_n^{(x)}(1).$$

A routine calculation shows that for $k \geq 2$, $h_k = g_{k+1}$. Using the functions h_k they very elegantly prove the following bounding theorem.

Theorem: For $n \geq 2$ $f \in \mathcal{L}_n \Rightarrow$ f can be computed by some loop program whose running time is bounded by $h_n^{(k)}$ for some fixed k.

This result is almost the same as the bounding lemma which Meyer and Ritchie prove for the Gzregorczyk hierarchy. Indeed, it is not hard to show that the two results are essentially identical, establishing the following theorem:

Theorem: For $n \geq 2$, $\mathcal{L}_n = E^{n+1}$.

Thus, depth of nesting of loops gives yet another reformulation of the Gzregorczyk hierarchy. We now know that loop structure, function size, running time, and depth of nesting of primitive recursions all yield essentially equivalent notions of complexity for the primitive recursive functions.

Section 2. ω-hierarchies of Elementary Functions

Another important topic of hierarchy research has centered aroung ω-hierarchies of the elementary functions. Ritchie's work [66] was the first investigation in this direction. He develops a hierarchy of the so called "predictably computable" functions, which he demonstrates to be precisely the elementary functions. While complexity theory, and in particular space considerations, are intrinsic to his hierarchy, his approach is much like Gzregorczyk's in spirit.

We briefly discuss a variant of Ritchie's hierarchy, due to Herman, and we also consider a formulation based on register machines, due to Cleave.

As in the case of ω-hierarchies of primitive recursive functions, the hierarchies discussed here are closely related, although in this case they turn out to be distinct.

Ritchie's work yields the following characterization of the elementary functions: a function $f(\bar{x}_k)$ is elementary if and only if some Turing machine computes f and uses no more than space $2^{2^{\cdot^{\cdot^{\cdot^{2^{\max(\bar{x}_k)}}}}}\Big\}n}$ for all $\bar{x}_k$, where n is some fixed constant which depends on f and which is independent of $\bar{x}_k$. His work is specifically based on one-tape Turing machines with binary input and output. Such conventions are significant here since the elementary functions are intimately related to the arithmetization of various machine models.

Definition: Let T be any Turing machine. Let $\bar{\mathbb{C}}_T(x_1, \cdots, x_n)$ be the number of tape squares used by machine T on inputs $x_1, \cdots, x_n$, if T converges on these arguments, and undefined otherwise.

Ritchie begins by defining F_0, the base of his hierarchy, to be the class of finite automaton computable functions. He shows that this class contains the successor function, the constant functions, and the projection functions, and is closed under composition and addition of functions. He defines his hierarchy as follows:

Definition: The Ritchie hierarchy.

1. F_0 = the finite automaton computable functions.
2. $f \in F_{i+1}$ in case some machine T computes f and $\mathbb{C}_T < g$ for some $g \in F_i$.

Thus $f \in F_{i+1}$ is "predictable computable" in the sense that the space needed to compute f is bounded by some function in F_i. In a moment we will sharpen the notion of predictability by giving explicit upper bounds on the space needed to compute $f \in F_{i+1}$.

It is easy to prove that $F_0 \subset F_1 \subset \cdots \subset F_n \subset \cdots$. To show that all the containments are proper, Ritchie develops a sequence of functions f_0, f_1, $\cdots$, f_k, $\cdots$ which are similar to the Gzregorczyk functions, and which yield canonical estimates on the size of functions in F_k.

Theorem: Let $f_0(x) = x$, and let $f_{n+1}(x) = 2^{f_n(x)}$. Then for each n, $f_n \in F_n$, and if $g \in F_n$, then $g(x_1, \cdots, x_n) < f_n(K \cdot \max(x_1, \cdots, x_n))$.

Easy inductions prove both claims, and since for strictly increasing g 2^g majorizes $k \cdot g$, $f_{n+1} \notin F_n$. This establishes the Ritchie hierarchy theorem.

Theorem: For all $n \geq 0$, $F_n \subsetneq F_{n+1}$.

Ritchie next establishes that the elementary functions $E \subset \bigcup_n F_n$. He accomplishes this by showing that exponentiation is in F_2, and that explicit transformation, limited recursion, and composition do not lead out of the F-classes. Finally by carefully analyzing and reworking the Kleene normal form Theorem, Ritchie is able to show that every $f \in \bigcup_n F_i$ is elementary. Thus his hierarchy is precisely a hierarchy of the elementary functions.

In [44] Herman develops a variant of the Ritchie hierarchy, based on unary Turing machines. As in the Ritchie formulation, Herman places $f \in G_i$ in case some machine T computes f and $G_T(x_1, \cdots, x_n) < f_{i-1}(K \cdot \max(x_1 \ldots x_n))$.

By examining carefully the resource needed to convert between binary and unary notation, he proves that for all $i > 0$ $F_i \subsetneq G_{i+1} \subsetneq F_{i+1}$, and thus that $\bigcup_i G_i$ = the elementary functions.

Cleave [81] proposes another method of building hierarchies, based on register machines. He fixes a set of functions Σ, and defines a Σ-program to be a finite sequence of instructions I(1), I(2), ..., I(k). Instructions may be of two forms: I(j) may be arithmetic, that is, of the form $F(R_1, \cdots, R_m) \rightarrow R_p$ (For $F \in \Sigma$, apply F to the contents of registers $R_1 \cdots R_m$, and place the result in R_p); or jump, that is $J_i(\alpha, \beta)$ (if $R_i = 1$, go to instruction $I(\alpha)$, and otherwise, go to $I(\beta)$). He limits his machines by specifying a special register J which is decremented by one each time a jump instruction is executed. When J = 0, the program halts.

Definition: A function f is (h-Σ) computable (that is, $f \in (\Sigma)^h$) if some Σ-program P computes f at each argument x, with special register J initially set to h(x).

Using this notion of bounded computability, Cleave constructs his hierarchy.

Definition: The Cleave hierarchy.

1. $f \in \Sigma_0 \Leftrightarrow f \in (\Sigma)^h$ for some constant function h
2. $f \in \Sigma_{n+1} \Leftrightarrow f \in (\Sigma)^g$ for some $g \in \Sigma_n$.

For $\Sigma = \{+, x, =\}$ Cleave shows the following:

Theorem: $\Sigma_0 \subsetneq \Sigma_1 \subsetneq \cdots \subsetneq \Sigma_n \subsetneq \cdots$, and $\bigcup_n \Sigma_n$ = elementary functions.

In [45], Herman considers the equivalence of the Ritchie and Cleave hierarchies. He shows that for $\Sigma = \{+, =\}$, $\bigcup_n \Sigma_n$ = elementary functions, and using an induction argument based on Ritchies functions $\{f_n\}_{n\in N}$, he shows that for $i \geq 0$, $F_i \subsetneq G_{i+1} \subsetneq \Sigma_{i+2} \subsetneq F_{i+2} \subsetneq G_{i+3}$.

Section 3. Transfinite Hierarchies

In the first part of this chapter we discussed ω-hierarchies of two well understood, effectively presentable subclasses of the recursive functions, the primitive recursive functions, and the elementary functions. In this section we discuss various attempts to build natural, effectively constructed transfinite hierarchies which are designed to exhaust the class of recursive functions in a non-trivial way. The results we consider here are almost without exception, negative. The fundamental difficulty with building exhaustive hierarchies is the highly non-invariant character of the ordinal names used to index such hierarchies. These "naming" difficulties have led to the formulation of transfinite hierarchies with more modest goals, namely, the construction of hierarchies indexed by apparently "natural" names for a small subset of the constructive ordinals. We discuss non-exhaustive hierarchies of this type at the end of this section.

One natural and attractive approach to the problem of constructing exhaustive transfinite hierarchies is through ordinal recursion. One might formulate such a hierarchy informally as follows: place a function $f \in F_\alpha$ for $\alpha < \omega_1$ (the first non-constructive ordinal), if f can be

defined by ordinal recursion over some well ordering $\beta \leq \alpha$ involving functions in F_λ for $\lambda \leq \alpha$. By unnested recursion over a well-ordering R of N we mean the following:

Definition: Let R be a well-ordering. Define $\underset{\sim}{R}$ to be

$$x\underset{\sim}{R}a = \begin{cases} x \text{ if } xRa, \\ 0 \text{ otherwise;} \end{cases}$$

Then a function f is defined by ordinal recursion over R (or unnested R-recursion), from given functions $g_1, \cdots, g_k$ if

$$f(0) = n$$

$$f(a+1) = h(a),$$

where h(a) has the form $p(a, f(q(a) \underset{\sim}{R} a+1))$, and p, q are built up from $g_1, \cdots, g_k$ by composition.

Definition: Let U(R), the unnested R-recursive functions, be the smallest class containing + and closed under composition, explicit transformation, and ordinal recursion over R.

The next theorem shows that the proposed hierarchy outlined above collapses at the earliest possible stage. The character of the proof hints at the close link between the "strength" of a transfinite hierarchy and the ordinal names used to index the hierarchy.

Theorem: Myhill, Routledge [60], [50], [31] and [32]. Let f be any recursive function. Then there exists a recursive well-ordering R can be shown to be elementary) of order type ω such that $f \in U(R)$.

One proof of this theorem proceeds by constructing R from the running time function Φ for some Turing machine T which computes f. R is built with an encoding of $\Phi(0)$, $\Phi(1)$, $\cdots$ embedded in it in an R-ordinal recursive way. Φ can be extracted from R in an ordinal recursive way, and, using the Kleene T-predicate and Φ, one shows that $f \in U(R)$.

This is certainly a provacative result; it indicates that if there is to be any hope of a successful transfinite hierarchy of the recursive functions, then the issue of ordinal names must be treated with considerable care.

With this in mind Kleene [27], proposed a subrecursive hierarchy in which classes of functions are attached to the nodes of 0, the Church-Kleene system of ordinal notations. We assume the reader is familiar with 0; a readable account of 0 and its properties may be found in [2, pp. 205-213].

Hoping to avoid the difficulties which arise from the Myhill-Routledge result, Kleene restricts 0 by allowing only primitive recursive fundamental sequences. He shows in fact that under this restriction 0 still names all the ordinals $< \omega_1$. In what follows, we assume 0 is restricted in this way.

Loosely speaking, Kleene's hierarchy starts with the primitive recursive functions at the base level, and is built up at successor levels by taking an enumerating function for the previous class and forming its primitive recursive closure. At limit notations Kleene assigns the primitive recursive closure of a function which encodes the enumerating functions of the classes named by the fundamental sequence.

Definition: The Kleene subrecursive hierarchy. Let $pr^f(a,b)$ enumerate the functions primitive recursive in f. The enumeration procedure pr is uniform in f. Associate a function h_x with each $x \in 0$ as follows:

(i) if $x = 1$, let $h_x(b,a) \equiv 0$

(ii) if $x = 2^y$, let $h_x(b,a) = p_r^{h_y}(b,a)$

(iii) if $x = 3 \cdot 5^z$ let $h_x(b,a) = h_{\varphi_z(\pi_2(b))}^{}(pr(\pi_1(b), z))$

To each $x \in 0$ assign the class of functions P_x, where P_x = the primitive recursive closure of h_x.

Let us consider the issues Kleene's hierarchy raises. To be completely successful, his (or any similarly formulated) hierarchy should satisfy the following properties:

(i) (uniqueness) For each $\alpha < \omega_1$, if $x, y \in 0$ and $|x|_0 = |y|_0 = \alpha$ (i.e. if x and y are notations for α) then $P_x = P_y$;

(ii) (proper expansion) For each $\alpha < \omega_1$, $\bigcup_{|x|_0<\alpha} P_x \subsetneq \mathcal{R}$, the recursive functions;

(iii) (completeness) $\bigcup_{x\in 0} P_x = \mathcal{R}$; and

(iv) The mapping $x \to P_x$ should be reasonably constructive, e.g., P_x is uniformly r.e. in x.

Such a hierarchy would provide considerable information about the class of total recursive functions. It would imply (subject to the restriction to primitive recursive fundamental sequences) that subrecursive hierarchies are ordinal invariant: no matter what choice of

names we select, we always generate the same sequence of classes of recursive functions. Moreover a hierarchy satisfying the properties listed above would provide us with a useful classification technique for measuring the complexity of recursive functions. We could identify the complexity of a function f with the least ordinal α such that $|x|_0 = \alpha$ and $f \in P_x$. This would be a significant measure of function complexity, since uniqueness would guarantee that no function f could appear at an artificially early level.

Unfortunately the Kleene hierarchy, and indeed any reasonably constructive hierarchy built in 0 must fail to satisfy the first three criteria. This breakdown means that any transfinite hierarchy of recursive functions must depend critically on the choice of ordinal names used to index the hierarchy. These negative results have made the aims of subrecursive hierarchy theory much more modest, and as we shall see much of the recent work on hierarchies is concerned with finding "nice" names for sequences of ordinals, and building non-exhaustive hierarchies along these paths.

Axt [12] is the first to consider Kleene's hierarchy. He shows that indeed the Kleene hierarchy is unique for $\alpha < \omega^2$. However, he also shows non-uniqueness at ω^2: there exist x, y $\in$ 0 such that $|x|_0 = |y|_0 = \omega^2$ but $P_x \neq P_y$.

Feferman [38] considers Kleene's hierarchy in a more general setting, and his work reveals a great deal about difficulties involved in building successful hierarchies in 0. Feferman proves his results for any "primitive recursively expanding hierarchy", that is any hierarchy

satisfying five (rather complicated) abstract properties, the most restrictive of which specifies that classes at limit notations must contain a function which diagonalizes across the classes named by the fundamental sequence.

His first result shows that in a primitively recursively expanding hierarchy, and in the Kleene hierarchy in particular, every recursive function occurs at a low level.

Theorem: Let $\{P_d\}_{d \in O}$ be the Kleene subrecursive hierarchy. For any $f \in \mathcal{R}$ there is a $d \in O$, $|d|_0 = \omega^2$, such that $f \in P_d$. Moreover for any $b \in O$ there is a $d \in O$, $b <_0 d$ and $|d|_0 = |b|_0 + \omega^2$ such that $f \in P_d$.

Feferman proves his theorem by showing how to encode any recursive function into a notation for ω^2. This result shows that for a large class of hierarchies, uniqueness must fail.

In [62], Parikh strengthens Feferman's non-uniqueness result.

Definition: (Parikh) A recursive transfinite progression of sets of functions over O (or any suitable subset of O, for example, O restricted to primitive recursive fundamental sequences) is an r.e. predicate $C(p, q, a, b)$ such that

(i) $x \in O$ implies that for any a, $\{< p,q > \mid C(p,q,a,x)\}$ is a function $f_{a,x}$: $\mathbb{N} \to \mathbb{N}$; and

(ii) If $x, y \in O$ and $x <_0 y$, then $C_x \subsetneq C_y$, where $C_x = \{f \mid \exists a\ (f = f_{a,x})\}$ and $C_y = \{f \mid \exists a (f = f_{a,y})\}$.

For such recursive transfinite progressions, of which Kleene's hierarchy is certainly an example, Parikh proves the following theorem.

Theorem: Every recursive transfinite progression of sets of functions is ω^2+1 non-unique; that is, there exist x, y $\in$ 0, $|x|_0 = \omega^2+1$ such that $C_x \neq C_y$.

Parikh's theorem is proved by methods similar to but simpler than those used to prove Feferman's result. The generality of his theorem is convincing evidence that transfinite subrecursive hierarchies are highly ordinal-name dependent.

Feferman has two other results, which, taken together, give concrete information on how dependent the strength of a hierarchy may be on the indexing ordinals for the hierarchy. By a path Z in 0 we mean a subset of 0 well-ordered by $<_0$ and containing, with any d $\in$ Z, all the predecessors of d. Let $|Z|$ denote the order type of Z.

Theorem: Let K be any ordinal $\leq \omega_1$. Then there exists paths Z, Z' $\subset$ 0, $|Z| = K + \omega^3$ for $K < \omega_1$, and $|Z'| = \omega_1$ for $K = \omega_1$, such that $\bigcup_{x\in Z} P_x = \bigcup_{x\in Z'} P_x = \mathfrak{R}$.

To prove the theorem with $K < \omega_1$, Feferman enumerates the recursive functions (a highly non-constructive procedure), and then, using $\dagger_0$ he iterates the techniques of his earlier theorem to obtain all the functions by $K + \omega^3$. For $K = \omega_1$, he enumerates 0 and the recursive functions, and he builds Z by alternately obtaining a new function, and then adding $(+_0)$ the next element in the 0-listing. This result establishes the existence of "complete" paths of length as short as ω^3, and as long as ω_1. This

is certainly a striking instance of ordinal non-invariance. It also shows that proper expansion is an impossiblity, at least for hierarchies of the Kleene type. The next result sharpens this phenomenon even further by showing that there are "incomplete" paths of length ω_1.

Theorem: These exist incomplete paths in 0 of length ω_1. That is, there exists a path $Z \in 0$, $|Z| = \omega_1$, and an $f \in \mathcal{R}$ such that for all $d \in Z$, $f \notin P_d$.

This is one of the deepest results in the theory of subrecursive hierarchies. The proof of the theorem builds on work done by Feferman and Spector in [39], in which a "non-standard" version of 0, 0^* is studied. 0^* is defined inductively as the intersection of all hyperarithmetic sets X satisfying

(i) $1 \in X$

(ii) if $d \in X$, then $2^d \in X$ and $d <_{0^*} 2^d$

(iii) if $\varphi_e(n) \in X$ for all n and $\varphi_e(n) <_{0^*} \varphi_e(n+1)$ for all n, then $3 \cdot 5^e \in 0^*$.

Interested readers unfamiliar with hyperarithmetic sets and their properties should consult [2, pp. 381-402].

Using this inductive definition, one can construct subrecursive hierarchies in 0* exactly as one constructs them in 0. Moreover, $0 \subsetneq 0^*$ and for $d \in 0$, the class of functions P_d attached to the d-node in 0 is exactly the same as the class P_d in 0^*. Feferman and Spector show that for any $d \in 0^* - 0$, $Z = C'(d) \cap 0$, (where $C'(d) = \{x \mid x <_{0^*} d\}$) is a π^1_1 path through 0 of order type ω_1. (For background material on π^1_1-sets, the interested reader should consult [2 , 397-403]).

Picking such a π^1_1-path Z in 0, we know there is a $d \in 0^*$ such that d "sits on top of" Z. Since the 0^*-hierarchy overlays the 0-hierarchy and agrees with the 0-hierarchy on 0, we know that P_d must properly contain $\bigcup_{x \in Z} P_x$. Hence $\bigcup_{x \in Z} P_x$ must omit some recursive function, and the following theorem, which applies to any subrecursive hierarchy in 0, is therefore established:

Theorem: Let Z be a π^1_1-path through 0 such that $Z = C'(d) \cap 0$ for $d \in 0^*$. Then there exists $f \in \mathcal{R}$ such that $f \notin \bigcup_{x \in Z} P_x$.

Combining the last two results, we see that the exhaustive power of a subrecursive hierarchy, at least of the Kleene-type, is intimately tied to the ordinal notations used in the hierarchy. In short, these results say that there are short (ω^3) complete hierarchies, and long (ω_1) incomplete hierarchies.

An unpublished result of Mochovakis [82] provides still more information on the behavior of hierarchies in 0.

Theorem: For $a \in 0$ (or any suitable version of 0, for example, Kleene's 0 restricted to primitive recursive fundamental sequences), let $A_a \subseteq N$. Then one of the following must fail:

1. $A = \bigcup_{a \in 0} A_a$ is hyperarithmetic; or
2. $P(x,a) \equiv [a \in 0 \text{ and } x \in A_a] \in \pi^1_1$; or
3. For each constructive ordinal α, $\bigcup_{|x|_0 < \alpha} A_x \subsetneq A$.

Properly interpreted, this theorem says that for any hierarchy on 0 built up in any manner which could possibly be considered constructive, if the recursive functions are exhausted at all, they are exhausted by some bounded level in 0.

Mochovakis proves his result by considering the π^1_1-predicate $Q(x,a) = [[x \notin A \text{ and } a = 1] \underline{\text{or}} [x \in A \text{ and } (P(x,a))]]$. The uniformization theorm [2, p.430] says that there must be a hyperarithmetic function g such that $\forall x\ Q(x, g(x))$; but then the range of g is an unbounded hyperarithmetic subset of 0, a contradiction.

In the case of the Kleene hierarchy, if we set $A_a = \{e \mid \varphi_e \in P_a\}$, then (1) and (2) are true, and so (3) must fail. Indeed, we saw for Kleene's hierarchy that this failure occurred at ω^2. Thus, even if one gives up the goal of uniqueness for hierarchies in 0, one must still contend with the problem that either the hierarchy will collapse by some bounded level, or it will omit some function.

By what we have just seen, hierarchies in 0 are extremely badly behaved. Such hierarchies can still be of use, however, for proving theorems about the various methods used in constructing hierarchies. As an example of this we consider the Bass-Young hierarchy [70]. This work has inspired many of the results in Chapter 2 of this thesis. In what follows, the reader is assumed to be familiar with Section 2 of Chapter 2.

Bass and Young build their hierarchy by starting with some complexity class $\mathcal{F}(t_1)$, where t_1 is some sufficiently large recursive function.

At successor stages they assign to notation 2^x the class $\mathcal{F}(t_{2^x})$, where t_{2^x} is obtained from t_x by an application of the honesty theorems followed by an application of the compression theorem. At limits they apply the union theorem of Meyer and McCreight [83], [84]. The resulting hierarchy is a recursive progression of sets of functions in the sense of Parikh, and so is non-unique at ω^2+1. However, Feferman's results do not apply: the union theorem insures that a limit class is precisely the union of the classes named by the fundamental sequence. In particular, the function which diagonalizes across the classes determined by the fundamental sequence does not appear in the limit class. Indeed, an appeal to the speed-up theorem of Blum [1] and the well-foundedness of 0 shows that no function with h-speed-up can appear anywhere in the hierarchy. Here $h \in \mathcal{R}_2$, the compression function used to build the hierarchy, is assumed to be monotone in its second argument. Using these techniques Bass and Young are able to construct a hierarchy on the full g in which every function is in the Gzregorczyk class E^4.

Bass and Young use Parikh's non-uniqueness result to establish several results about inherent irregularities of honesty procedures. For example, they prove the following theorem.

Theorem: For sufficiently large $h \in \mathcal{R}_2$ there exist honest functions t_1, t_2 such that $\mathcal{F}(t_1) = \mathcal{F}(t_2)$, but $\mathcal{F}(h(x, t_1(x))) \neq \mathcal{F}(h(x, t_2(x)))$.

This result and others like it in their paper led directly to our work in Chapter 2 on the honesty phenomenon.

By what we have just seen, the non-invariant character of ordinal notations makes the construction of a meaningful exhaustive hierarchy of the recursive functions extremely unlikely. The construction therefore of "short" hierarchies which classify only a portion of the recursive functions seems to be a more legitmate if more modest goal.

We survey several approaches to this problem. Hierarchies can be built up by unnested and nested ordinal recursion over particularly natural well orderings. By restricting attention to such well-orderings one can avoid the difficulties inherent in the Myhill-Routledge result. Another approach extends existing ω-length hierarchies into the transfinite. We discuss invariance between these hierarchies. A linearly-ordered Kleene hierarchy can be constructed by selecting a nice path in O and examining the Kleene hierarchy restricted to this set. The results of these investigations show that if one chooses ordinal names with care, then one can indeed build interesting and singificant hierarchies of portions of the recursive functions.

We begin by discussing work by Tait [52] relating unnested and nested ordinal recursion over a well ordering R of N. Recall that for R a well-ordering, the function $x\underset{\sim}{R}y$ is equal to x if xRy and 0 otherwise.

Definition: A function f is defined by nested R-recursion over R from functions $g_1, \cdots, g_k$ if f satisfies

$$f(0) = n,$$

$$f(a+1) = h(a),$$

where h(a) is built up from $g_1, \cdots, g_k$ and f by composition, but where every application of f has the form $f(x\underset{\sim}{R}\, a+1)$.

Definition: The R-nested (ordinal) recursive functions, N(R), is the smallest set containing + and closed under the operations of composition, explicit transformation, primitive recursion, and nested R-recursion.

Tait points out that, in the case of unnested R-recursion, computation of f(a+1) proceeds in a linear way down a well-ordering until f(0) is reached and evaluated. For nested R-recursion, the computation of f(a+1) may lead to a computation tree, and the value of f(a+1) cannot be determined until the computations on each path of the tree have been reduced to known functions or constants. The comparison of these two types of recursion lies in the analysis of these two forms of computation.

Definition: Let R be a well-ordering. Define R^* to be the limit of all polynomials in ω of the form $\omega^{\alpha_i}\cdot a_1 + \omega^{\alpha_2}\cdot a_2 + \cdots + \omega^{\alpha_n}\cdot a_n$, for $\alpha_n \leq \alpha_{n-1} \leq \cdots \leq \alpha_1 < R$, and $a_1, \cdots, a_n$ integers. R^* has order type $\omega^{|R|}$.

If R is a recursive well-ordering of N we can assign integers to polynomials in ω of the above form. This assignment induces an R^* ordering of N, and it is not hard to show that this ordering is primitive recursive in R.

Definition: Define $<_1 = \omega$, $<_{n+1} = <_{n+1} \times <_n$; define $Q_0 = <_1$, $Q_{n+1} = <_n^*$.

Thus, $|<_n| = \omega^n$, and $|Q_{n+1}| = \omega^{\omega^n}$. Moreover, for each n $<_n$ and Q_n are recursive well-orderings on N.

Using tree analysis of nested computaions as a guide, Tait shows that for the well-orderings $<_n$, nested recursion on $<_{n+1}$ is reducible to unnested recursion on Q_n.

Theorem: For $n \geq 0$, if $f \in N(<_{n+1})$ then $f \in U(Q_n)$.

Robbin [68] proves the converse of Tait's theorem and puts these results in a more hierarchy theoretic framework. He obtains significant results about various short hierarchies and their relationship to one another. In particular he relates these results to the multiply-recursive functions of Peter [10].

Peter invented the multiply recursive functions after Ackermann had shown that nested double recursion (Ackermann's function) leads out of the class of nested single recursion definable functions, the primitive recursive functions. The function ψ defined by the equations

$$\psi(o,n) = n+1$$

$$\psi(m+1, 0) = \psi(m, 1)$$

$$\psi(m+1, n+1) = \psi(m, \psi(m+1, n))$$

is an example of a "2-recursive" function: the inductive definition is done over two arguments, and the computation of ψ is nested in the sense that to compute $\psi(m+1, n+1)$, one must first evaluate ψ at other arguments. Peter generalizes this to k variables for $k \geq 2$ and obtains the "k-recursive" functions for each $k > 0$. She considers the k-recursive functions with k as a parameter, the so called multiply-recursive functions, and shows by a diagonal argument that for each k, the k+1-recursive functions properly contain the k-recursive functions. We denote the k-recursive functions by N_k.

Robbin's first main theorem relates nested and unnested ordinal recursion to the Peter hierarchy.

Theorem: $N_{n+1} = N(<_{n+1}) = U(Q_n)$.

Robbin relates these results to an extended version of the Gzregorczyk hierarchy and a linearly ordered portion of the Kleene hierarchy. He deals with the problem of ordinal notations by specifying very carefully how limit ordinals are to be approached.

Definition: For α a limit ordinal $<\omega^\omega$, let $\alpha = \omega^{k+1}(\beta+1)$. Define $\lambda n\alpha(n)$ such that $\lim_n \alpha(n) = \alpha$ to be $\alpha(n) = \omega^{k+1}\cdot\beta+\omega^k\cdot n$.

Using this definition Robbin defines a sequence of Gzregorczyk-like functions $\mathbf{W}_\alpha$ which are quite similar to the modified Gzregorczyk functions g_n later introduced by Meyer and Ritchie.

Definition: For $\alpha < \omega^\omega$, define W_α as follows:

1. $W_0(x) = 2^x$
2. $W_{\alpha+1}(x) = W_\alpha^{(x)}(1)$
3. $W_\alpha(x) = W_{\alpha(x)}(x)$ for α a limit ordinal.

The W_α's provide a natural way to extend the Gzregorczyk hierarchy.

Definition: For each $\alpha < \omega^\omega$, define E^α to be $E(W_\alpha)$, that is E^α = the functions elementary in W_α.

It is easy to see that a proper hierarchy is established. Robbin is able to show that his extended Gzregorczyk hierarchy refines the hierarchy of multiply recursive functions, and hence also the nested and unnested ordinal recursion hierarchies.

Theorem: For $\alpha < \beta$, $E^{\alpha} \subsetneq E^{\beta}$; moreover, for each k $N_k = \bigcup_{\alpha<\omega^k} E_{\alpha}$.

Robbin's proof uses ideas which were employed later in the Meyer-Ritchie account of the Gzregorczyk hierarchy. He proves a bounding lemma relating the size of the W_{α}'s to the multiply-recursive functions, and a key step in his proof is an appeal to the honesty of the functions W_{α}.

We remarked earlier that the 1-recursive functions of Peter are the primitive recursive functions. If $f(x) = g^{(x)}(1)$, we say that f is obtained from g by 1-fold iteration, and we can generate the primitive recursive functions by using this iteration scheme instead of the schema for primitive recursion. Robbin extends this equivalence, showing that the k-recursive functions can be obtained by replacing the schema for k-recursion with a schema for k-fold iteration, a generalization of 1-fold iteration.

Using k-fold iteration, Robbin gives an analysis of a Kleene-type ω^{ω}-hierarchy in terms of the multiply-recursive functions. He defines his hierarchy as Kleene does, but he chooses a single path through 0 out to ω^{ω}, the path determined by his $\alpha(n)$ fundamental sequences.

Theorem: Let P_{α}, $\alpha < \omega^{\omega}$ be the Kleene subrecursive hierarchy restricted to the 0-path determined by the $\alpha(n)$ fundamental sequences. Then for $n \geq 1$, $N_n = \bigcup_{\alpha<\omega^{n-1}} P_{\alpha}$.

Robbin's work is an excellent example of how short hierarchies can yield information about various notions of difficulty for subclasses of the recursive functions. His results relate nested and unnested ordinal recursion to the multiply-recursive functions, and through the extended Gzregorczyk hierarchy, to the actual size of functions.

Earlier we discussed a construction of Cleave's which yielded an ω-hierarchy of the elementary functions. In the same paper Cleave extends his hierarchy to ω^2, and shows that the resulting hierarchy exhausts the primitive recursive functions.

Definition: The ω^2-Cleave hierarchy.

(i) $f \in \Sigma_0 \Leftrightarrow f \in (\Sigma)^h$ for some constant function h

(ii) for $k > 0$, $f \in \Sigma_{\omega \cdot r+k} \Leftrightarrow f \in (\Sigma)^h$ for $h \in \Sigma_{\omega \cdot r+(k-1)}$

(iii) for $r > 0$, $f \in \Sigma_{\omega \cdot r} \Leftrightarrow f \in \bigcup_{k=0}^{\infty} \Sigma_{\omega \cdot (r-1)+k}$

Cleave's work is of interest for several reasons. First, the construction of a proper ω^2 length hierarchy of the primitive recursive functions indicates that ordinal length, even for hierarchies which only exhaust a portion of the recursive functions, can be a misleading measure of hierarchy strength. (Of course, the subsequent construction by Bass and Young of a proper hierarchy in the full 0 which fails to exhaust E^4 is a more spectacular example of this phenomenon.) Second, Cleave's construction brings out some of the difficulties involved in the construction of hierarchies by machine theoretic means. Indeed, Cleave points out that his hierarchy must die out at ω^2. He argues as follows: since each program is of fixed length, P = I(1), I(2), $\cdots$, I(k), if $f \in \Sigma_{\omega^2}$ then all the functions used to define f must appear in $\Sigma_{\omega \cdot k}$ for some k. Hence, extension of the chain beyond ω^2 yields nothing new, since any $f \in \Sigma_{\omega^2+1}$, say, must already appear in some $\Sigma_{m \cdot k}$.

This inherent limitation of Cleave's approach is by-passed by Constable

[18], who uses RASP machines to extend the Cleave hierarchy to ϵ_0, the limit of the sequence ω, ω^{ω}, $\omega^{\omega^{\omega}}$, $\cdots$. A RASP machine is perhaps the closest to real computers of all theoretically proposed machines. Its fundamental characteristic for our purposes is its ability to monitor and modify itself in the course of a computation. This is a fundamental difference between RASPs and register machines, and this difference accounts for Constable's successful extension to ϵ_0.

For ordinals $\alpha < \epsilon_0$ Constable carefully handles the problem of finding nice fundamental sequences. He puts α in (unique) Cantor normal form,

$$\alpha = \omega^{\alpha_1}\cdot a_1 + \cdots + \omega^{\alpha_n}\cdot a_n \quad \text{for } \alpha_1 \geq \cdots \geq \alpha_n, \quad \text{and}$$

$$a_1, \cdots, a_n \text{ integers,}$$

and then he defines his fundamental sequences:

Definition: Let $\alpha < \epsilon_0$ be a limit ordinal in Cantor normal form as above. If α_n is successor ordinal, define

$$\alpha(x) = \omega^{\alpha_1}\cdot a_1 + \cdots + \omega^{\alpha_n - 1}\cdot x;$$

if α_n is a limit ordinal, define

$$\alpha(x) = \omega^{\alpha_1}\cdot a_1 + \cdots + \omega^{\alpha_0(x)}.$$

Using this formulation of fundamental sequences, Constable extends the Cleave hierarchy using RASP machines, and he also extends the Gzregorczyk classes (already extended to ω^{ω} by Robbin) to ϵ_0. His Gzregorczyk extension is a direct generalization of Robbin's extension: for $\alpha < \epsilon_0$, $E^{\alpha} = E(\omega_{\alpha})$, where $W_{\alpha}(x) = W_{\alpha-1}(x)$ if α is a limit ordinal, and $W_{\alpha}(x) = W_{\alpha(x)}(x)$ iff α is a limit ordinal. His RASP hierarchy of length

ε_0 is proper because his RASP programs modify themselves in the course of their execution, thus avoiding the problems of the register machine approach. Constable establishes the following result:

Theorem: For $\alpha < \varepsilon_0$ $E^{\alpha+1} = RASP_{(1+\alpha)+1}$, where $RASP_\beta$ is the β^{th} RASP hierarchy class.

Thus Constable is able to extend to ε_0 the growing body of results relating various generation methods for short hierarchies.

In [71], Schwichtenberg also considers the equivalence problem for various ε_0-length hierarchies. He shows that the modified Kleene hierarchy, the generalized Gzregorczyk hierarchy, and a standardized unnested recursion hierarchy all coincide up to ε_0. He defines standard fundamental sequences exactly as Constable does, and his version of the extended Gzregorczyk hierarchy is the same as Constable's. Moreover, he extends, with minor modifications, Robbins version of the Kleene hierarchy to ε_0. His unnested ordinal recursion classes, R_α, are defined in a rather unusual way, and the analysis of these classes is the most original part of the paper.

Definition: Define well-ordering S_n of N as follows: $S_1 = \omega$, $S_{n+1} = \omega^{S_n}$. A standard well-ordering of type $\alpha < \varepsilon_0$ is a well-ordering of the natural numbers which is elementary-recursive isomorphic to an initial segment of S_{n+1} for $S_n < \alpha \leq S_{n+1}$.

Schwichtenberg considers only standard well-orderings $< \varepsilon_0$; functions defined by instances of unnested λ-recursion for standard well-orderings $\lambda < \varepsilon_0$ are said to be defined by elementary λ-recursion. The ε_0-recursive functions, then, are the set of functions which can be defined by elementary-

λ-recursion, $\lambda < \varepsilon_0$, from given ε_0-recursive functions and elementary functions in an elementary way.

Schwichtenberg assigns ordinals $< \varepsilon_0$ to ε_0-recursive functions, and he uses this assignment to define his ordinal recursion classes. If f is defined explicitly from $g_1, \cdots, g_k$ in an elementary way, then f is assigned the ordinal $\underline{\max}(\alpha_1, \cdots, \alpha_k)$, where the α_i's are the ordinals assigned to the g_i's. If f is defined using an $\omega \cdot \alpha$-elementary recursion from $g_1, \cdots, g_k$, then f is assigned the ordinal $\underline{\max}(\alpha_1, \cdots, \alpha_k) + \alpha$.

Definition: R_α is the set of recursive functions which are assigned ordinals $\leq \alpha$.

This rather curious definition is the key to Schwictenberg's results: by allowing R_α to contain functions defined by $\omega \cdot \alpha$ recursions, he gives himself enough slack to prove his main result.

Theorem: For all $\alpha < \varepsilon_0$ the extended Gzregorczyk hierarchy class $E_\alpha = R_\alpha = P_\alpha$, the modified Kleene class.

The critical part of the theorem is the proof that $R_\alpha \subseteq E_\alpha$. Here Schwichtenberg introduces a formal reduction system for the ε_0-recursive functions, and he develops a step-counting function s_f for each $f \in R_\alpha$ which keeps track of the reductions necessary to evaluate f. He shows that for $f \in R_\alpha$, $s_f \in R_\alpha$. Moreover, he shows that each function in R_α can be defined from elementary functions alone by a single $\omega \cdot \alpha$ recursion. Using this he establishes his claim by proving that each function in R_α is majorized by $W_\alpha(g(x))$, where g is some elementary function.

Schwichtenberg also notes that the ε_0-recursive functions are equal to the so-called "provable recursive functions". A recursive function f is provably recursive if for some index e for f $\forall x \exists y\ T(e,x,y)$ is provable in elementary number theory, where T is the Kleene T-predicate. For a thorough account of the provably recursive functions, see Fischer [40].

Schwichtenberg's very elegant paper is one of the best examples of a successful hierarchy construction of constructive ordinal length. His work is a natural extension of Robbin's work from ω^ω to ε_0.

In a sense the Schwichtenberg result may be one of the last investigations in short hierarchy theory. While work in the Schwichtenberg framework obviously could be extended beyond ε_0, it is not clear what sort of insight such an investigation would provide.

We turn therefore to a different method of classifying the recursive functions, the method of subrecursive degrees.

Section 4. Subrecursive Degrees

As we have seen, subrecursive hierarchies constitute an important and extensively studied approach to the problem of classifying the recursive functions. A fundamental problem with the hierarchy approach is the difficulties inherent in attempts to exhaust the recursive functions in any meaningful way. An immediate attraction of the degree approach, which we turn to now, is inclusiveness: every total recursive function belongs to some primitive recursive (or elementary recursive) degree.

The degree approach was initiated by Kleene [27]. He directly applied the concepts and notations of Turing degrees of unsolvability to the subrecursive case to obtain primitive recursion degrees.

Definition: Let f and g be total functions. We say f is primitive recursive in g, $f \leq_p g$, if f is definable in a primitive recursive way using g as an additional initial function. The degree of f, $d(f) = \{g \mid f \leq_p g \text{ and } g \leq_p f\}$.

Following the development of Turing degrees closely, he defines $d(f) \cup d(g)$ (the join of f and g), and $d(f)'$ (the jump of f). $d(f) \cup d(g) = d(2^f \cdot 3^g)$, and $d(f)'$ equals d(h), where h is an enumerating function for the functions primitive recursive in f which is generated in a uniform, primitive recursive way.

Kleene ends his work here, and Axt [12] continues Kleenes investigation of the basic properties of primitive recursive degrees. His main result is the analogue of the celebrated Friedberg-Muchnik Theorem,

Theorem: For each n there exists n pairwise incomparable primitive recursive degrees contained in the recursive Turing degree.

We emphasize that primitive recursiveness is not the only notion which can be analyzed by a degree approach. Indeed, we could just as easily study elementary degrees or multiply-recursive degrees and achieve basically the same results. In fact, with few exceptions, theorems proved for one such concept carry over to the others with little effort.

We can also consider studying subrecursive classes of functions, rather than degrees.

Definition: $Pr(f)$, the primitive recursive class of f, is the set of functions primitive recursive in f.

It is not hard to show that there is an order preserving isomorphism between the primitive recursive degrees and the primitive recursive classes (or, for that matter, between elementary degrees and elementary classes). Indeed, the map which sends $d(f) \rightarrow Pr(f)$ is the desired isomorphism. Much of the work to date on the structure of subrecursive degrees has actually centered around subrecursive classes rather than degrees, and we consider these investigations now.

Early work on the structure of subrecursive classes was done by Meyer and Ritchie [72]. They consider elementary honest classes, as outlined in Section 1 of this chapter, and they show that between any two Grzegorczyk classes E^n and E^{n+1} for $n \geq 3$, there are dense chains of elementary honest classes. They prove their result by interpolating between the iterates of g_n, where $E(g_n) = E^n$ and $E(g_{n+1}) = E(\lambda x\, g_n^{(x)}(1)) = E^{n+1}$.

They also prove the existence of denumerable incomparable families of elementary honest classes between E^3 and E^4.

Feferman [38] also has a density result: he shows the existence of dense chains in 0^*, and hence that there are dense chains of primitive recursive degrees.

Similar results by other investigators are discussed at the end of Chapter 3.

In a series of three papers [8], [85], and [86], Machtey develops an extremely elegant structure theory for elementary and primitive recursive classes.

Definition: Let $\underset{\sim}{C}(f)$ denote the subrecursive class generated by the recursive function f. If the class under consideration is the set of functions elementary in f, then $\underset{\sim}{C}(f) = \{\underset{\sim}{C}_i(f) \mid i \in N\}$, where $\underset{\sim}{C}_i(f)$ is the i^{th} function elementary in f.

Central to Machtey's approach is his complexity-theoretic point of view. He picks as a measure of computation Turing machine space (see Section 2 of Chapter 2 for definitions). He then makes a fundamental distinction: a class $\underset{\sim}{C}(f)$ is an honest class if $\underset{\sim}{C}(f) = \underset{\sim}{C}(S_i)$ for some space function (measure function) S_i; otherwise $\underset{\sim}{C}(f)$ is said to be a dishonest class. The fundamental property of honest subrecursive classes is that they are complexity classes, that is, they equal the t-computable functions for some recursive function t. Machtey establishes a great many structure results in these papers, and we consider some of them.

Theorem: Every countable partial order can be embedded in the dishonest subrecursive classes.

Machtey proves this result using techniques developed by Sacks to analyze the structure of the r.e. Turing degrees.

Definition: Two sequences of honest functions $f_0, f_1, \cdots$ and $g_0, g_1, \cdots$ determine a gap if, for all i, $\underset{\sim}{C}(f_i) \subsetneq \underset{\sim}{C}(f_{i+1})$, $\underset{\sim}{C}(g_{i+1}) \subsetneq \underset{\sim}{C}(g_i)$, and $\underset{\sim}{C}(f_i) \not\subseteq \underset{\sim}{C}(g_i)$. An effective gap is a gap for which there is a set $\{i_0, i_1, \cdots,\}$ which is recursive in 0' (the complete r.e. Turing degree) such that for all i $f_j = \varphi_{i_{2j}}$ and $g_j = \varphi_{i_{2j+1}}$.

Theorem: Any countable partial order can be embedded in the honest subrecursive classes between any effective gap.

This rather complicated result has two important corollaries.

Corollary: The honest subrecursive classes are dense; that is, if f and g determine honest classes $\underset{\sim}{C}(f) \subsetneq \underset{\sim}{C}(f)$, then there exists an h such that $\underset{\sim}{C}(h)$ is honest, and $\underset{\sim}{C}(f) \subsetneq \underset{\sim}{C}(h) \subsetneq \underset{\sim}{C}(g)$.

Corollary: No r.e. properly increasing sequence of honest subrecursive classes has a least upper bound in the honest subrecursive classes.

Machtey also proves the following result, which is rather unexpected given that the corresponding result fails for the r.e. Turing degrees.

Theorem: The partial ordering of the honest subrecursive classes is a distributive lattice.

The novel element of Machtey's work is his distinction between honest and dishonest subrecursive classes. This is a distinction which allows the elegant methods of complexity theory to play a role, and leads to his more interesting results, for example, his lattice result for honest degrees.

In [92], Ladner examines the structure of subrecursive classes and obtains results similar to Machtey's.

Theorem: The subrecursive degrees are dense, and are not a lattice.

He also considers the problem of minimal degrees.

Theorem: There exist minimal pairs of elementary degrees. That is, there exist recursive functions f and g such that if $h \leq_e f$ and $h \leq_e g$, then h is elementary (here $h \leq_e f$ means h is elementary in f).

Ladner is particularly interested in considering the range of his (or Machtey's) results. His methods certainly apply to primitive recursive or multiply-recursive degrees, etc., as do Machtey's. However, he also discusses abstract notations of reducibilities which, hopefully, will shed some light on concrete problems in theoretical computer science. We discuss one such notion here.

Definition: A set S of unary functions is a space class if it is r.e., contains the identity, and for all f and g in S and constants c_1 and c_2 there exists an h $\in$ S such that

(i) h is increasing

(ii) $h(n) \geq c_1 \cdot f(n) + c_2$

(iii) $h(n) \geq f(x(n))$,

(iv) $h(n) \geq \max[f(n), g(n)]$.

The class of linear functions, and the class of polynomial functions are examples of space classes.

Ladner considers 0-1 valued functions, that is "decision problems", for his notion of reducibility. If p(x) and g(x) are 0-1 valued, he defines p to be S-space reducible to g if some oracle Turing machine with oracle g computes p in space bounded by some function in S.

He then concludes that for the degree structure induced by S-space reducibility, the two theorems of his paper quoted above are true.

References

1. Blum, M., A machine-independent theory of the complexity of recursive functions, JACM 14, 1967, 322-336.

2. Rogers, H. Jr., Theory of recursive functions and effective computability, Mcgraw-Hill, 1967.

3. Sacks, G., Degrees of Unsolvability, Annals of Math. Studies, No. 55, Princeton, N.J., 1963.

4. Enderton, H., Degrees of computational complexity, JCSS 6 , 1972, 389-396.

5. Meyer, A., and Ritchie, D., Classification of functions by computational complexity, Proc. of the Hawaii International Conf. on Sys.Sciences, 1968, 17-19.

6. Basu, S.K., On classes of computable functions, ACM Symp. on Theory of Computing, 1969, 55-61.

7. Alton, D., Operator embeddability in computational complexity, Notices of the AMS, 1972, A-763.

8. Machtey, M., Augmented loop languages and classes of computable functions, JCSS 6, 1972, 603-624.

9. Feferman, S., Classification of recursive functions by means of hierarchies, Trans. of the AMS 104, 1962, 101-122.

10. Peter, R., Recursive functions, Academic Press, New York, 1967.

11. Anderson, D., Nested ordinal recursive functions and a subrecursive hierarchy, doctoral thesis, Duke University, Durham N.C., 1961.

12. Axt, P., On a subrecursive hierarchy and primitive recursive degrees, Trans. of the AMS 92, 1959, 85-105.

13. Axt, P., Enumeration and the Gzregorczyk hierarchy, Zeitschrift fur mathematische Logik und Grundlagen der Mathematik 9, 53-65.

14. Axt, P., Iteration on primitive recursion, Zeitschrift fur mathematische Logik unf Grundlagen der Mathematik 11, 1965, 253-55.

15. Bennett, J.H., On spectra, doctoral thesis, Princeton University, Princeton, N.J., 1962.

16. Church, A., and Kleene, S.C., Formal definitions in the theory of ordinal numbers, Fundamenta Mathematicae 28, 1937, 11-21.

17. Cleave, J.P., A hierarchy of primitive recursive functions, Zeitschrift fur mathematische Logik und Grundlagen der Mathematik 9, 1963, 331-346.

18. Constable, R.L., Extending and refining hierarchies of computable functions, Tech. Report 25, Comp. Sci. Dept., University of Wisconsin, 1968.

19. Constable, R.L., Subrecursive programming languages for R^n, Tech. Report 70-53, Comp. Sci. Dept., Cornell University, 1970.

20. Jones, N.D., Classes of Automata and transitive closure, Information and Control 13, 1968, 207-229.

21. Kazanovich, IA. B., A classification of the primitive recursive functions with the help of Turing machines, Problemy Kibernetiki 22, 1970, 95-106, in Russian.

22. Kent, C.F., Reducing ordinal recursion, Proceedings of the AMS 22, 1969, 690-696.

23. Kleene, S.C., On notations for ordinal numbers, JSL 3, 1938, 150-55.

24. Kleene, S.C., Arithmetical predicates and function quantifiers, Trans. of the AMS 79, 1955, 312-40.

25. Kleene, S.C., Hierarchies of number theoretic predicates, Bulletin of the AMS 61, 1955, 193-213.

26. Kleene, S.C., On the forms of the predicates in the theory of constructive ordinals, II, American Journal of Mathematics 77, 1955, 405-28.

27. Kleene, S.C., Extension of an effectively generated class of functions by enumeration, Colloquium Mathmaticum 6, 1958, 67-78.

28. Kreider, D.L., and Ritchie, R., Predictably computable functionals and definition by recursion, Zeitschrift fur mathematische Logik und Grundlagen der Mathematik 10, 1964, 65-80.

29. Kreisel, G., Non uniqueness results for transfinite progressions, Bulletin de l'Academie Polonaise des Science, Serie des Science mathematiques, astronomiques et physiques 8, 1960, 287-290.

30. Kreisel, G., Shoenfield, J., and Wang, H., Number theoretic concepts and recursive well-orderings, Archiv fur mathemathische Logik und Grundlagenforschung 5, 1959, 42-64.

31. Liu, S., A theorem on general recursive functions, Proceedings of the AMS 11, 1960, 184-187.

32. Liu, S., A generalized concept of primitive recursion and its application to deriving general recursive functions, Hung-Ching Chow 60th Anniversity volume, Institute of Mathematics, Academia Sinica, Taipei, 1962, 93-98.

33. Constable, R.L., On the size of programs in subrecursive formalisms, Tech. Report 70-58, Dept. of Comp. Sci., Cornell University, 1970.

34. Cook, S.A., A survey of classes of primitive recursive functions, notes for Mathematics 290, University of California, Berkeley, 1967.

35. Fabian, R.J., Hierarchies of general recursive functions and ordinal recursion, doctoral thesis, Case Institute of Tech., Cleveland Ohio, 1964.

36. Constable, R.L., and Borodin, A.B., On the efficiency of programs in the subrecursive formalisms, Tech. Report 70-54, Dept. of Comp. Sci., Cornell University, 1970.

37. Fabian, R.J., and Kent, C.F., Recursive functions defined by ordinal recursions, Proceedings of the AMS 23, 1969, 206-210.

38. Weihrauch, K., Hierarchien primitive-rekursiver Wortfunktionen I, Bericht 50, Instituts fur Theorie der Automaten Schalnetzwerke, Gesellschaft fur Mathematik und Datenverarbeitung, Bonn Germany, 1972.

39. Feferman, S., and Spector, C., Incompleteness along paths in progressions of theories, JSL 27, 1962, 383-390.

40. Fischer, P., Theory of provable recursive functions, Trans. of the AMS 117, 1965, 494-529.

41. Grzegorczyk, A., Some classes of recursive functions, Rozprawy Matematyczne 4, 1953, 1-45.

42. Hart, J., $\forall E'0$-Arithmetic, Zeitschrift fur mathematische Logik und Grundlagen der Mathematik 15, 1969, 273.

43. Heinermann, W., Untersuchungen uber die Recursionszahlen recursiver Funktionen, Dissertation, Munster, Germany, 1961.

44. Herman, G.T., A new hierarchy of elementary functions, Proceedings of the AMS 20, 1969, 557-62.

45. Herman, G.T., The equivelence of various hierarchies of elementary functions, Zeitschrift fur mathematische Logik und Grundlagen der Mathematik 17, 1971, 115-131.

46. Rodding, D., Uber die Eliminierbarkeit von Definitionsschemata in der Theorie der recursiven Funktionen, Zeitschrift fur mathemathische Logik und Grundlagen der Mathematik 10, 1964, 315-30.

47. Rodding, D., Darstellungen der elemetaren Funktionen, Archiv fur mathematische Logik und Grundlagenforschungen 7, 1965, 139-158.

48. Rodding, D., Darstellung der elementaren Funktionen II, Archiv fur mathematische Logik und Grundlagenfurschung 9, 1966, 36-48.

49. Rodding, D., Klassen rekusiver Funktionen, in Proceedings of the Summer School in Logic, Leeds, 1967, ed. M.H. Lob, Lecture Notes in Mathematics 70, 1967, Springer-Verlag, Berlin.

50. Routledge, N.A., Ordinal recursion, Proceedings of the Cambridge Philosophical Society 49, 1953, 175-182.

51. Schwichtenberg, H., Rekursionzahlen und die Grzegorczyk-Hierarchie, Archiv fur mathematische Logik und Grundlagenforschung 9, 1966, 36-48.

52. Tait, W., Nested recursion, Mathematische Annalen 143, 1961, 236-50.

53. Thompson, D.B, Subrecursiveness and finite computers, doctoral thesis, Stanford University, 1968.

54. Tsichritzis, D., and Weiner, P., Some unsolvable problems partial solutions, Tech. Report 69, Princeton University, 1968.

55. Cleave, J.P., and Rose, H.E., E'n Arithmetic, in Sets, Models and Recursion Theory, ed. J.N. Crossley, North Holland, Amsterdam, 1967, 297-308.

56. Cobham, A., The intrinsic computational difficulty of functions, in Proceedings of the 1964 International Congress for Logic, Methodology, and Philosophy of Science, ed. Y. Bar-Hillel, North Holland, Amsterdam, 1964, 24-30.

57. Meyer, A., Depth of nesting of primitive recursion: another formulation of the Grzegorczyk hierarchy, term paper for Applied Mathematics 230, Harvard University, 1965.

58. Meyer, A., and Ritchie, D., Computational complexity and program structure, IBM Tech. Report 1917, 1967.

59. Meyer, A., and Ritchie, D., The complexity of loop programs, Proceedings of 22nd National Conference of the ACM, 1967, 465-69.

60. Myhill, J., A stumbling block in constructive mathematics, abstract in the JSL 18, 1953, 190-191.

61. Nepomnyaschy, V.A., The rudimentary interpretation of two-tape Turing's calculations, Kibernetika 6, 1970, 29-35.

62. Parikh, R., On non uniqueness in transfinit progressions, Journal of the Indian Mathematical Society 31, 1967, 23-32.

63. Parsons, C., Hierarchies of primitive recursive functions, Zeitschrift fur mathematische Logik und Grundlagen der Mathematik 14, 1968, 357-76.

64. Ritchie, D., Complexity classification of primitive recursive functions by their machine programs, A term paper for Applied Mathematics 230, Harvard University, 1965.

65. Ritchie, D., Program structure and computational complexity, Harvard University, doctoral thesis, 1968.

66. Ritchie, R., Classes of predictably computable functions, Trans. of the AMS 106, 1963, 139-72.

67. Ritchie, R., Classes of recursive functions based on Ackermann's function, Pacific Journal of Mathematics 15, 1965, 1027-44.

68. Robbin, J., Subrecursive hierarchies, doctoral thesis, Princeton University, 1965.

69. Bass, L., Hierarchies based on computational complexity and irregularities of class determining measured sets, doctoral thesis, Purdue University, 1970.

70. Bass, L., and Young, P., Ordinal hierarchis and naming classes, JACM, to appear.

71. Schwichtenberg, H., Eine Klassifikation der ϵ_0-rekusiven funktionen, Zeitschrift fur mathematische Logik und Grundlagen der Mathematik 17, 1971, 61-74

72. Meyer, A., and Ritchie, D., A classification of the recursive fu functions, Zeitschrift fur mathematische Logik und Grundlagen der Mathematik 18, 1972, 71-82.

73. Basu, S.K., On the structure of the subrecursive degrees, JCSS 4, 1970, 452-64.

74. Crossley, J.N., and Schutte, K., Non uniqueness at ω^2 in Kleene's O, Archiv fur mathematische Logik und Grundlagenforschung 9, 1966, 95-101.

75. Warkentin, J.C., Small classes of recursive functions and relations, doctoral thesis, Research Report CSRR 2052, Dept. of Applied Analysis and Comp. Sci., University of Waterloo, Waterloo, Ontario, 1971.

76. Davis, M., Computability and Unsolvability, Mcgraw-Hill, New York, 1958.

77. Kozhminykh, V.V., On primitive recursive functions of one argument, Algebra i Logika,7,1, 75-90; translation: Algebra and Logic 7, 44-53.

78. Robinson, R.M., Primitive recursive functions, Bull. of the AMS 53, 1947, 925-942.

79. Thompson,D.B., Subrecursiveness: machine-independent notions of computability in restricted time and storage, Math. Systems Theory 6, 1972, 3-15.

80. Meyer, A., Depth of nesting of primitive recursion: another formulation of the Grzegorczyk hierarchy, Notices of the AMS 13, 1965, 342.

81. Schwichtenberg, H., Eine Klassifikation der E-0 rekursiver Funktionen, Zeitsch. f. Wahrscheinlichskeittheorie und verwandte Gebiete 17, 1971, 61-74.

82. Mochovakis, Y.N., A remark on subrecursive hierarchies, unpublished memoir, 1964.

83. McCreight, E.M., and Meyer, A., Classes of computable functions defined by bounds on computation: preliminary report, Conf. Record of ACM Conf. on Theory of Computing, 1969, 79-88.

84. McCreight, E.M., Classes of computable functions defined by bounds on computation, doctoral dissertation, Carnegie-Mellon University, Pittsburgh, Pa., Dept. of Comp. Sci., 1969.

85. Machtey, M., The honest subrecursive classes are a lattice, Purdue University, Comp. Sci. Dept. Tech. Report 82, 1972.

86. Machtey, M., On the density of subrecursive classes, Purdue University, Comp. Sci. Dept. Tech. Report 92, 1973.

87. Lob, M.H., and Wainer, S.S., Hierarchies of number-theoretic functions I, Archiv fur mathematische Logik und Grundlagenforschung 13, 1970, 39-51.

88. Lob, M.H., and Wainer, S.S., Hierarchies of number-theoretic functions II, Archiv fur matmematische Logik und Grundlagenforschung 13, 1970, 97-113.

89. Lob, M.H., and Wainer, S.S., Hierarchies of number-theoretic functions I,II,: a correction, Archiv fur mathematische Logik und Grundlagenforschung 14, 1971, 198-199.

90. Marchenkov, S.S., Multiple recursion bounded in the class of primitive recursive functions, Kibernetika 6, 6, 1970, 53-59.

91. Kozmidiadi, V.A., and Muchnik, A.A., Problems of mathematical logic: complexity of algorithms and classes of computable functions, Collection of translations, MIR(Publ.), Moscow.

92. Ladner, R., Polynomial time reducibility, in Proceedings of Fifth Annual ACM Symposium on Theory of Computing, Austin, Texas, 1973, 122-130.

Chapter 2

Honest Bounds for Complexity Classes of Recursive Functions

1. Introduction

Let $\mathcal{F}(t)$ be the set of recursive functions computable by machines using $t(x)$ computation steps on argument x, for all but finitely many inputs x. We call t a name for the complexity class $\mathcal{F}(t)$. Suppose we allow our machines to run longer, say $h(x,t(x))$ steps on argument x, where h is some fixed recursive function. One might expect that for large enough h, permitting our machines to run longer by an amount h will always allow us to compute new functions, i.e. $\mathcal{F}(t)$ is a proper subset of $\mathcal{F}(h(x,t(x))$. This turns out not to be the case: the "gap theorem" [2], [3] implies that for every recursive h there exists a recursive t such that $\mathcal{F}(t) = \mathcal{F}(h(x,t(x)))$. However, if we restrict our attention to names from a certain subclass of the recursive functions, then we can indeed uniformly increase the size of our $\mathcal{F}$-classes. Informally, we call a recursive function t "honest" if some machine computes $t(x)$ in roughly $t(x)$ steps for each argument x. (A precise definition is given in Definition 1 below.) Then according to the

"compression theorem " [3], there exists a single recursive function h such that for every honest t, $\mathcal{F}(t)$ is a proper subset of $\mathcal{F}(h(x,t(x)))$. Thus the phenomenon of the gap theorem is avoided by restricting attention to honest functions. It is a surprising consequence of the "honesty theorem" of McCreight and Meyer [4], [5] that there is no loss of generality in this restriction. Namely, for any recursive function t there is an honest recursive function t' such that $\mathcal{F}(t) = \mathcal{F}(t')$.

In this paper we present a new simplified proof of the honesty theorem, and then we analyze the possible behaviors of precedures for constructing honest names equivalent to arbitrarily given names. Part of the motivation for this analysis springs from the construction of hierarchies of recursive functions based on computational complexity. Bass and Young [7] have observed that application of the honesty theorem followed by the compression theorem to a function t yields a reasonable natural "jump" to a larger complexity class. The behavior of this jump operation and the resulting hierarchy of course depend critically on the honesty procedure being used.

Section 2 describes our notation and the axioms of Blum [1] which provide a machine-independent characterization of running time; Blums measured sets [1] and classes of honest functions are shown to be essentially equivalent. Section 3 consists of our new proof of the honesty theorem. In section 4 we consider honesty procedures which work on partial functions

as well as total functions, and we show that such procedures must generate arbitrarily large names for any complexity class. As a corollary we obtain another "gap"-like theorem which shows that every complexity class has honest names which are arbitrarily large on all but a vanishing fraction of arguments, thereby strengthening a result of [8]. In section 5 we show that honesty procedures restricted to total functions need not yield arbitrarily large names for classes, and can preserve monotonicity, thereby settling questions raised in [7], [4].

2. Preliminaries

For notation from recursive function theory we follow Rogers [9].

For each $n \in \mathbb{N}$, $\mathcal{P}_n$ stands for the partial recursive functions of n variables. $\mathcal{R}_n$ stands for the total recursive functions of n variables.

We use "(a.e.)" to denote "almost everywhere", which for our purposes stands for "all but finitely many". Similarly "(i.o.)" stands for "infinitely often".

If ψ and φ are partial functions and φ is undefined at argument x we adopt the convention that $\psi(x) \leq \varphi(x)$.

Suppose $\{\varphi_0, \varphi_1, \ldots\}$ is a Gödel numbering of $\mathcal{P}_1$. A <u>measure on</u> computation [1] $\Phi = \{\Phi_0, \Phi_1, \ldots\}$ is a sequence of functions in $\mathcal{P}_1$ satisfying

1) $\forall i \in \mathbb{N}[\text{dom}(\varphi_i) = \text{dom}(\Phi_i)]$

2) $\lambda i\ x\ y[\Phi_i(x) = y]$ is a recursive predicate.

If we think of our Gödel numbering in the usual one-tape Turing machine formalism, then

$\Phi_i(x)$ = "the number of steps in the computation of the i^{th} Turing machine on argument x" is a measure on computation.

Henceforth let Φ be some fixed measure on computation. Then we define for any total function t

$$F(t) = \{i \in \mathbb{N} \mid \varphi_i \in \mathcal{R}_1 \text{ and } \Phi_i \leq t \text{ (a.e.)}\},$$

and

$$\mathcal{F}(t) = \{\varphi_i \mid i \in F(t)\}.$$

That is, F(t) is the set of (indices of) total machines or programs which run in time t, and $\mathcal{F}(t)$ is the set of total functions computable within time t. Similarly we define for any partial function ψ

$$F_p(\psi) = \{i \in \mathbb{N} \mid \Phi_i \leq \psi \text{ (a.e.)}\}$$

and

$$\mathcal{F}_p(\psi) = \{\varphi_i \mid i \in F_p(\psi)\}.$$

A sequence of partial functions $\Psi = \{\psi_0, \psi_1, \ldots\}$ is said to be an r.e. sequence of partial functions if $\lambda i\ x[\psi_i(x)] \in \mathcal{P}_2$.

Definition 1. (McCreight-Meyer [4]) A function $\psi \in \mathcal{P}_1$ is g-honest for $g \in \mathcal{R}_2$ if there is an i such that $\varphi_i = \psi$ and $\Phi_i \leq \lambda x g(x,\psi(x))$ (a.e.).

Definition 2. (Blum [1]) An r.e. sequence of partial functions $\Psi = \{\psi_0, \psi_1, \ldots\}$ is said to be a measured set* if

$$\lambda ixy[\psi_i(x) = y] \text{ is a recursive predicate.}$$

The relationship between honest functions and measured sets is explained by the following theorem of Meyer-McCreight [4]. Since the proof appears only in McCreight's unpublished thesis [5], we reproduce it here.

Theorem 1. [4], [5]. Every measured set Ψ is made up of g-honest functions for some $g \in \mathcal{R}_2$; furthermore the set of g-honest functions form a measured set.

Proof. Let $\Psi = \{\psi_0, \psi_1, \ldots\}$ be a measured set. By Definition 2 and elementary recursion theory there is an $s \in \mathcal{R}_1$ such that $\psi_i = \varphi_{s(i)}$. Define

$$g(x,y) = \max\{\Phi_{s(i)}(j) \mid i, j \leq x \text{ and } \psi_i(j) \leq y\}.$$

Then for $x > i$ we have $\Phi_{s(i)}(x) \leq g(x, \varphi_{s(i)}(x))$, and so for each i $\varphi_{s(i)} = \psi_i$ is g-honest.

---

*Measured sequence would be more accurate, but we conform to the terminology of Blum [1].

To prove the second statement consider the partial recursive function $\varphi_{\sigma(i,j,k)}$ for $\sigma \in \mathcal{R}_3$, which, roughly, imitates $\varphi_i(x)$ when $\varphi_i(x)$ appears to be g-honest from arguments j to x. More precisely

$$\varphi_{\sigma(i,j,k)}(x) = \begin{cases} \varphi_i(x) & \underline{\text{if}}[(x \le j \ \underline{\text{and}} \ \Phi_i(x) \le k \ \underline{\text{or}} \ (x > j \ \underline{\text{and}} \\ & \quad \Phi_i(x) \le g(x, \varphi_i(x)))] \\ & \underline{\text{and}}[(\forall y \le j)[\Phi_i(y) > k \Rightarrow \Phi_i(y) > x]] \\ & \underline{\text{and}}[(\forall y)(j < y \le x)[\Phi_i(y) < x \Rightarrow \Phi_i(y) \\ & \quad \le g(y, \varphi_i(y))]] \\ \infty & \text{otherwise.} \end{cases}$$

It follows from the definition of measure on computation that $\lambda i, j, k, x, z[\varphi_{\sigma(i,j,k)}(x) = z]$ is a recursive predicate. Hence

$$S = \{\varphi_{\sigma(i,j,k)} \mid i,j,k \ge 0\} \text{ is a measured set.}$$

We claim S equals the g-honest functions. Indeed if for fixed i, j, k $\Phi_i(x) \le g(x, \varphi_i(x))$ for all $x > j$ and

$$k \ge \max\{\Phi_i(y) \mid y \le j \text{ and } \Phi_i(y) \text{ convergent}\},$$

then $\varphi_{\sigma(i,j,k)} = \varphi_i$ and $\varphi_{\sigma(i,j,k)}$ is g-honest.

If however the preceeding condition is not met, then $\varphi_{\sigma(i,j,k)}$ diverges (a.e.), but such functions are also (by convention) g-honest. So S is a subset of the g-honest functions.

Furthermore, if γ is any g-honest function, then $\gamma = \varphi_i$ for some i such that $\Phi_i(x) \leq g(x, \varphi_i(x))$ for all $x > j$ for some j. Let $k = \max[\Phi_i(y) \mid y \leq j \text{ and } \Phi_i(y) \text{ convergent}]$. Then $\gamma = \varphi_{\sigma(i,j,k)}$ and we conclude that S equals the g-honest functions. □

We state, for completeness, the following generalized compression theorem of Blum [1].* The compression theorem says that an r.e. sequence of partial recursive functions is a measured set precisely when a uniform procedure exists for constructing, for each function in the sequence, a 0-1 valued partial recursive function whose complexity is only a little bit above the designated function.

Proposition. Let $\Psi = \{\psi_0, \psi_1, \ldots\}$ be an r.e. sequence of partial recursive functions. Then Ψ is a measured set if and only if there is a $p \in \mathcal{R}_3$ and an $r \in \mathcal{R}_1$ such that (1) $\varphi_{r(i)}$ is 0-1 valued, (2) domain $(\varphi_{r(i)})$ = domain (ψ_i), (3) $\Phi_{r(i)} \leq \lambda x[p(i,x, \psi_i(x)]$, and (4) if $\varphi_e = \varphi_{r(i)}$, then $\Phi_e > \psi_i$ (a.e.).

It is an immediate corollary of the compression theorem that if we restrict attention to recursive functions t from a measured set Ψ, then we can uniformly enlarge $\mathcal{F}(t)$ by composing t with a fixed recursive function h independent of t.

Corollary. Let Ψ be a measured set. Then there exists an $h \in \mathcal{R}_2$ such that if $t \in \Psi$, $t \in \mathcal{R}_1$, then $\mathcal{F}(t) \subsetneq^{+} \mathcal{F}(\lambda x[h(x, t(x))])$.

---

*We remark that Blum's theorem in [1] p. 333 is incorrectly stated; the correct statement is given above.

Proof. Let $h(x,z) = \max_{e \leq x} [p(e,x,z)]$, where p is the recursive function of the proposition.

Definition 3. Let $s \in \mathfrak{R}_1$. s is an honesty procedure on $\mathfrak{P}_1$ if $\lambda e,x,y[\varphi_{s(e)}(x) = y]$ is a recursive predicate, and if for every e $\mathcal{F}_p(\varphi_e) = \mathcal{F}_p(\varphi_{s(e)})$.

Definition 4. Let $s \in \mathfrak{R}_1$. s is an honesty procedure on $\mathfrak{R}_1$ if $\lambda e,x,x[\varphi_{s(e)}(x) = y]$ is a recursive predicate, and if for every total φ_e, $\varphi_{s(e)}$ is total and $\mathcal{F}(\varphi_e) = \mathcal{F}(\varphi_{s(e)})$.

Notice that not every honesty procedure on $\mathfrak{P}_1$ need be an honesty procedure on $\mathfrak{R}_1$: an honesty procedure on $\mathfrak{P}_1$ need not map total functions to total functions. However, suppose s is an honesty procedure on $\mathfrak{P}_1$ which also preserves F-classes. That is, suppose that for every e, $F_p(\varphi_e) = F_p(\varphi_{s(e)})$. Then a minor modification of s yields an honesty procedure on $\mathfrak{P}_1$ and on $\mathfrak{R}_1$. Indeed, it is easy to show that $s' \in \mathfrak{R}_1$ defined by

$$\varphi_{s'(e)}(x) = \min[\varphi_{s(e)}(x),\ (\varphi_e(x) + \Phi_e(x))]$$

in such a procedure.

Constable has observed that no honesty procedure on $\mathfrak{R}_1$ can be a total effective operator. We prove a corresponding result for honesty procedures on $\mathfrak{P}_1$ and effective operators (see [9] for definitions).

Proposition. No honesty procedure on $\mathfrak{P}_1$ can be an effective operator.

Proof. Let s be any honesty procedure on $\mathfrak{P}_1$, and let $t = \varphi_j$ be any recursive function. Define using the recursion theorem:

$$\varphi_e(x) = \begin{cases} \varphi_j(x) & \text{if } \exists z[\varphi_{s(j)}(z) \neq \varphi_{s(e)}(z)] \\ \infty & \text{otherwise.} \end{cases}$$

The computation of $\varphi_e(x)$ is effective since $\varphi_{s(e)}$ and $\varphi_{s(j)}$ are in a measured set. Clearly φ_e is either total or empty. If φ_e is empty, it follows that s cannot be an honesty procedure on $\mathcal{P}_1$, for then $\mathcal{F}_p(\varphi_{s(e)}) = \mathcal{F}_p(\varphi_{s(j)}) = \mathcal{F}_p(\varphi_j) \subsetneq \mathcal{P}_1 = \mathcal{F}_p(\varphi_e)$. **So φ_e must be total.** Then $\varphi_e = \varphi_j$ and $\varphi_{s(e)} \neq \varphi_{s(j)}$. □

3. The Honesty Theorem

The honesty theorem says that given any function we can effectively find an honest, function which names the same class. Our proof explicity exhibits an honesty procedure on $\mathcal{P}_1$. Recall from section two, however, that with a minor modification we can obtain a procedure on $\mathcal{R}_1$ as well.

Theorem 2. There exists an honesty procedure on $\mathcal{P}_1$. Moreover, s preserves F_p classes, namely for every e, $F_p(\varphi_e) = F_p(\varphi_{s(e)})$.

Proof. Let e be an index for ψ. A function ψ' such that $F_p(\psi) = F_p(\psi')$ is defined in stages beginning with stage 0. At stage n the integers from 0 to n will be ordered in a sequence or queue = $q_0, q_1, \ldots, q_n$, which is updated from stage to stage. Also a zero-one valued function "pop" on the integers from 0 to n is defined and updated from stage to stage. Let $< x,y >$ be a one-one onto pairing function with projection functions π_1 and π_2. As a technical convenience we use the fact that the pairing function $< x,y >$ is strictly increasing in its second argument, so that stage $< x,y >$ always precedes stage $< x, y+1 >$.

We outline the idea of the construction. Dovetail the computations of ψ, Φ_0, Φ_1, ..., Φ_n ... at all arguments. Whenever it is discovered that $\psi(x) < \Phi_i(x)$ set pop(i) = 1, and try to define $\psi'(z) < \Phi_i(z)$ for some argument z. When pop(j) = 0, try to keep $\psi'(z) \geq \Phi_j(z)$. The pop conditions on i and j may be inconsistent, and the queue assigns priorities to the integers (programs) to resolve the conflict. The dovetail nature of the construction guarantees that ψ' will be honest.

Stage n.

A) Put n on the bottom of the queue (i.e. set $q_n = n$). Set pop(n) = 1. Let $\pi_1(n) = x$, $\pi_2(n) = y$.

B) If $\Phi_e(x) = y$, then for $0 \leq i \leq n$, if $\Phi_i(x) > \psi(x)$ set pop(i) = 1.

C) If $\psi'(x)$ has already been defined at some previous stage, go to stage n+1.

D) Find the least $i \leq n$ (if any) such that

1) $pop(q_i) = 1$

2) $\Phi_{q_i}(x) > y$

3) $(\forall j < i)[pop(q_j) = 0 \rightarrow \Phi_{q_j}(x) \leq y]$

If i exists, define $\psi'(x) = y$, set $pop(q_i) = 0$, and put q_i on the bottom of the queue. Go to stage n+1. If no such i exists, go to stage n+1. □

For any $\varphi_e = \psi$ and any $n \geq 0$, stage n in the computation of ψ' is effective and will terminate. Condition (C) guarantees that if $\psi'(x)$ is defined, it is defined at only one stage, and so ψ' is well defined. Furthermore since our procedure is uniform in e, $\psi' = \varphi_{s(e)}$ for some $s \in \mathcal{R}_1$. Condition (D) guarantees that if $\psi'(x)$ is defined, it is defined at stage $n = \langle x, \psi'(x) \rangle$; hence the predicate $\lambda e\ x\ y[\varphi_{s(e)}(x) = y]$ is recursive (we need only run our procedure until stage $\langle x,y \rangle$), and so $\{\varphi_{s(e)}\}_{e=0}^{\infty}$ is a measured set. This implies by Theorem 1 that ψ' will be g-honest for some $g \in \mathcal{R}_2$ independent of ψ.

We now show that for each i, $\Phi_i \leq \psi$ (a.e.) $\Leftrightarrow \Phi_i \leq \psi'$(a.e.). This immediately implies $F_p(\psi) = F_p(\psi')$.

The proof divides into cases depending on the final positions of the integer i on the queue. If i reaches a final location on the queue we shall say that i is stable; otherwise we say i is unstable.

Case 1: i is unstable.

If i does not stabilize it must be moved to the bottom of the queue by step (D) at stage $\langle x,y \rangle$ for infinitely many x. Step (D) defines $\psi'(x) = y < \Phi_i(x)$, and hence $\Phi_i > \psi'$(i.o.). Moreover step (D) moves i to the bottom of the queue only if pop(i) = 1, at which time pop(i) is reset to 0. In order for step (D) to apply again to i, pop(i) must be reset to 1 by step (B) at some later stage. But condition (B) sets pop(i) to 1 only at stages $\langle x, \Phi_e(x) \rangle$ such that $\Phi_i(x) > \psi(x)$. Thus $\Phi_i > \psi$ (i.o.).

Case 2: i is stable.

If i reaches a stable position on the queue, then pop(i) must also stabilize since it is set to 0 only by a step (D) execution, at which time i is moved to the bottom of the queue.

Case 2a: pop(i) stable at 0.

Pop(i) can be set to 1 by step (B) at only finitely many arguments, hence $\Phi_i \leq \psi$ (a.e.). Elements above i on the queue can only be moved finitely often by step (D), for otherwise i would be unstable. So for almost all arguments x in the domain of ψ', $\psi'(x)$ is defined via step (D) for some j below i on the queue with pop(j) = 1. But then condition (2) of step (D) guarantees that $\Phi_i(x) \leq \psi'(x)$. Hence $\Phi_i \leq \psi'$(a.e.).

Case 2b: pop(i) stable at 1.

Consider any x such that i, the elements above i on the queue, and their pops have stabilized at stage $< x,0 >$ and all later stages. By case 2a we may assume x is sufficiently large that $\Phi_j(x) \leq \underline{\min}(\psi(x), \psi'(x))$ for those (finitely many) j which are above i on the queue with pop(j) = 0. Let

$$m = \underline{\max}\{\Phi_j(x) \mid j \text{ is above i on queue and pop(j) = 0}\}.$$

We observe that $m \leq \underline{\min}[\psi(x), \psi'(x)]$, and thus if m is infinite, both $\psi(x)$, $\psi'(x)$ are undefined, implying by convention that $\Phi_i(x) \leq \psi(x)$, $\Phi_i(x) \leq \psi'(x)$. So suppose m is finite. Since the pairing function is monotone in its second argument, $< x,m >$ is the earliest stage at which $\psi'(x)$ could be defined without violating condition (3) of step (D) and our assumption that the queue above i has stabilized. But i has stabilized as well, and so i must fail to satisfy condition (2) of step (D) at stage $< x,m >$. That is, $\Phi_i(x) \leq m$, and we therefore have $\Phi_i(x) \leq m \leq \underline{\min}(\psi(x), \psi'(x))$.

Combining cases we have $\Phi_i(x) \leq \psi(x)$ (a.e.) $\Leftrightarrow$ i is stable $\Leftrightarrow$ $\Phi_i(x) \leq \psi'(x)$ (a.e.). □

Corollary. There exists an honesty procedure on $\mathcal{R}_1$.

Proof. Immediate from section two and the fact that the procedure of Theorem 2 preserves F-classes as well as $\mathcal{F}$-classes.

4. Large Honest Bounds on Computation

Given a recursive function t we can think of t as a name for the class of functions $\mathcal{F}(t)$. Now in a sense we have understood a complexity class if we know how to compute its name, t. It follows that more easily computed functions (i.e. functions which can be computed rapidly) are more satisfactory names for a given class than long-running functions. Honest functions seem to be good candidates for names, then, because they are only as hard to compute as they are large. We now show that in general honest functions are not necessarily satisfactory names in the sense described above. Indeed we exhibit an honesty procedure on $\mathcal{R}_1$ which takes any recursive class name to an honest recursive name for the same class which is arbitrarily large (and therefore arbitrarily long-running) on all but a rapidly vanishing percentage of arguments. Furthermore we prove that any honesty procedure on $\mathcal{P}_1$ must (almost) have this property. We remark that this phenomenon is closely related to the gap theorem mentioned in the introduction: in both cases we pass from a recursive function t to a much larger recursive function t' while preserving class size.

Theorem 3. There is an honesty procedure s, on $\mathcal{P}_1$, such that for every e

$$\lim_{x\to\infty} \frac{|\{y \le x \mid y \in \text{domain } (\varphi_{s(e)})\}|}{x} \to 0.$$

Proof of the Theorem. The procedure of the theorem is only a slight variant of the procedure of Theorem 2. As before ψ' is defined in stages beginning with stage 0. A function "pop" from integers to integers is defined and updated during successive stages. Clause (D) has the added restriction that when pop(i) is larger than x, i is excluded from the priority scheme of the queue at arguments $\le x$. The pop function is sufficiently fast-growing to insure that only a small fraction of the entries on the queue can be used to define ψ' at arguments $\le x$. Hence at "most" arguments $\le x$, ψ' will be undefined.

A) Put n on the bottom of the queue, (i.e. set $q_n = n$); set $pop(n) = 2^n$. Set $x = \pi_1(n)$, $y = \pi_2(n)$.

B) If $\Phi_e(x) = y$, then for each i, $0 \le i \le n$, if [pop(i) = 0 and $\Phi_i(x) > \psi(x)$] set $pop(i) = 2^n$.

C) If $\psi'(x)$ has been defined previously go to stage n+1.

D) Find the least $i \le n$ (if any) such that

1) $0 < pop(i) < x$

2) $\Phi_{q_i}(x) > y$, and

3) $(\forall j < i)(pop(j) = 0 \to \Phi_{q_j}(x) \le y)$

If such an i exists, set $pop(q_i) = 0$ and move q_i to the bottom of the queue. Go to stage n+1. If no such i exists, go to stage n+1. □

We omit the proof that our procedure is indeed an honesty procedure on Φ_1; the proof here is virtually identical with that given in Theorem 2. We prove the limit condition of the lemma. Given any x, step (A) guarantees that at any stage $n = \langle y,z \rangle$ where $y \leq x$, at most $\log_2(x)$ indices on the queue can have pops which might be used in step (D) condition (1) to define $\psi'(y)$. Furthermore, if i is such an index and if i is used again at stage $n = \langle y,z \rangle$, $y \leq x$, to define $\psi'(y)$, then if it is to be used again at some later stage to define $\psi'(w)$ for some other $w \leq x$, its pop will be at least 2^{n+1}. Hence i can be used to define at most $\lceil \log_2(x)-i \rceil$(the greatest integer in $(\log_2(x)-i)$) arguments $y \leq x$. Thus $\psi'(x)$ can be defined on at most

$$\sum_{j=0}^{\lceil \log_2 x \rceil} \lceil \log_2(x)-j \rceil$$

arguments $\leq x$. So

$$\frac{\left|\{y \leq x \mid y \in \text{domain}(\psi)\}\right|}{x} \leq \frac{\sum_{j=0}^{\lceil \log_2(x) \rceil} \lceil \log_2(x)-j \rceil}{x};$$

but the right hand expression goes to 0 in the limit, proving the theorem. □

Theorem 3 leads naturally to the following result about honesty procedures on $\mathcal{R}_1$.

Theorem 4.* There is an honesty procedure on $\mathcal{R}_1$ such that given any $t \in \mathcal{R}_1$ and any $b \in \mathcal{R}_1$, there exists an e, $\varphi_e = t$, such that

$$\lim_{x\to\infty} \frac{|\{y \le x \mid \varphi_{s(e)}(y) < b(y)\}|}{x} \to 0.$$

Proof. Let s be the honesty procedure on $\mathcal{P}_1$ described in Theorem 3. Recall that we can make s into an honesty procedure on $\mathcal{R}_1$ by defining $\varphi_{s'(e)}(x) = \underline{\min}[\varphi_{s(e)}(x), (\varphi_e(x) + \Phi_e(x))]$. Let t be any recursive function. Blum [1] shows that every recursive function has arbitrarily bad (i.e. arbitrarily long running) programs. That is, we can choose $\varphi_e = t$ such that $\Phi_e(x) > b(x)$ for all x. Hence given t and b, choose such an e, $\varphi_e = t$, and then $\varphi_{s'(e)}$ satisfied the theorem. □

The following theorem describes the behavior of any honesty procedure on $\mathcal{P}_1$.

Theorem 5. Let s be any honesty procedure on $\mathcal{P}_1$ and let t and b be any recursive functions. Then there is a $\varphi_e = t$ such that

$$\liminf_{x\to\infty} \frac{|\{y \le x \mid \varphi_{s(e)}(y) < b(y)\}|}{x} \to 0.$$

*Bass and Young [7] prove a somewhat weaker form of this theorem: they show that an e can be found such that $\varphi_{s(e)}$ will be larger than b with recursive frequency.

Proof. Define using the recursion theorem a program φ_e such that

$$\varphi_e(x) = \begin{cases} t(x) & \text{if } [(x = 0) \text{ or } (x > 0 \text{ and } \varphi_e(x-1) \text{ convergent})] \text{ and } \exists z > x \text{ such that } \dfrac{|\{y \le z \mid \varphi_{s(e)}(y) < b(y)\}|}{z} < \dfrac{1}{x+1} \\ \infty & \text{otherwise.} \end{cases}$$

Clearly, if φ_e is total, then $\varphi_e = t$. Suppose φ_e is not total, and let x be the least y such that $\varphi_e(y)$ diverges. Then $\varphi_e(z)$ diverges for all $z \ge x$, but since $\varphi_e(x-1)$ converges, the first clause in the definition of $\varphi_e(x)$ implies that for all $z > x$

$$\frac{|\{y \le z \mid \varphi_{s(e)}(y) < b(y)\}|}{z} > \frac{1}{x+1}.$$

In particular, domain $(\varphi_{s(e)})$ must be infinite. However, it is easy to show that if $\psi \in \mathcal{P}_1$ has infinite domain, then $\mathcal{F}_p(\psi) \neq \mathcal{P}_1$. Hence $\mathcal{F}_p(\varphi_{s(e)}) \neq \mathcal{P}_1 = \mathcal{F}_p(\varphi_e)$, contradicting the fact that s is an honesty procedure on $\mathcal{P}_1$. Therefore, $\varphi_e = t$.

Now for each x let z_x be the least $z > x$ for which the second conjunct in the definition of φ_e holds. Then $\{z_x\}_{i=0}^{\infty}$ is a subsequence of the integers for which

$$\lim_{x \to \infty} \frac{|\{y \le z_x \mid \varphi_{s(e)}(y) < b(y)\}|}{z_x} \to 0$$

and the theorem is proved. □

We remark that the "lim inf" appearing in Theorem 5 cannot be replaced by "lim". We sketch breifly why this is so. Let s be the honesty procedure on $\mathcal{P}_1$ of Theorem 2, and let t be any g-honest recursive function. We construct an honesty procedure s_t on $\mathcal{P}_1$ in the following manner. Given index e, begin constructing $\varphi_{s(e)}$ as prescribed in the theorem. If at some stage n it is discovered that φ_e has converged on a new initial segment, see if $\varphi_e = t$ on that initial segment. If so find the least x such that stage $< x,0 >$ has not yet been reached and define $\varphi_{s_t(e)}(z) = t(z)$ for $x \leq z \leq 2x$. It is not hard to show that s_t is a legitimate honesty procedure on $\mathcal{P}_1$, and furthermore for any $\varphi_e = t$

$$\limsup_{x \to \infty} \frac{\left|\{y \leq x \mid \varphi_{s_t(e)}(y) < t(y)\}\right|}{x} \geq \frac{1}{2}.$$

In particular not all honesty procedures on $\mathcal{P}_1$ satisfy Theorem 3.

5. Good Honest Names for Complexity Classes

In this section we consider honesty procedures that work for total functions only. We show that by relaxing the requirements on honesty procedures in this way, we can indeed build well-behaved honest bounds for complexity classes which often significantly improve on the original bound for the class. We first build an honesty procedure yielding honest bounds which are no larger than the original class bound on a significant percentage of arguments. Next we show how to keep honest

bounds for $\mathcal{F}(t)$ bounded (a.e.) in a manner independent of the program we choose for t. Lastly we exhibit an honesty procedure on $\mathcal{R}_1$ which preserves monotonicity.

Implicit in the work of Constable [11] is the observation that there are complexity classes all of whose honest names are much larger (i.o.) than some dishonest name. Indeed any class $\mathcal{F}(t)$ where t is obtained via the gap theorem has this property. Theorem 6 shows that this result is false if we replace "(i.o.)" with "(a.e.)".

Theorem 6. There is an honesty procedure on $\mathcal{R}_1$, s , such that if φ_e is total, then

$$\liminf_{n\to\infty} \frac{|\{x \le n \mid \varphi_{s(e)}(x) > \varphi_e(x)\}|}{n} = 0.$$

Proof of the Theorem. The proof follows the general outline of Theorem 2. In the course of the procedure we define a "percentage" function $\mathcal{P}(n)$ which monitors the frequency with which $\varphi_{s(e)}$ is small. In addition the pop function in this proof is 0-1-2 valued. Here pop(i) = 2 means that $\varphi_{s(e)}$ has been defined to be less than Φ_i, but movement of i to the bottom of the qneue has been delayed.

Stage n:

Let $\pi_1(n) = x$, $\pi_2(n) = y$.

A) Set $q_n = n$, and set pop(n) = 1.

B) If $\Phi_e(x) = y$, then for $0 \le i \le n$, if $\Phi_i(x) > \varphi_e(x)$ and $pop(i) = 0$, set $pop(i) = 1$.

C) If $\psi'(x)$ has been defined at some previous stage, go to stage n+1.

D) Find least $i \le n$, if any, such that

1) $pop(q_i) = 1$ or 2

2) $\Phi_{q_i}(x) > y$

3) $(\forall j < i)[pop(q_i) = 0 \rightarrow \Phi_{q_j}(x) \le y]$

If such an i exists, set $\psi'(x) = y$, and set $pop(q_i) = 2$.

E*) See if $\exists w \le n$ such that $w > P(n)$ and

$$\frac{|\{z \le w \mid \varphi_{s(e)}(z) \le \varphi_e(z) \text{ and } \Phi_{s(e)}(z),\ \Phi_e(z) \le n\}|}{w} \ge 1 - \frac{1}{P(n)}$$

If such a w exists, set all 2's on queue to 0, and move them to the bottom of the queue. Set $P(n+1) = P(n) + 1$. Go to next stage.

If no such w exists, set $P(n+1) = P(n)$, and go to next stage. □

As in the proof of theorem 2, there is an $r \in \mathcal{R}_1$ such that the procedure yields, for ever e, a function $\varphi_{r(e)} = \psi'$. Furthermore, $\{\varphi_{r(e)}\}_{e \in \omega}$ is a measured set. Define

$$t(x) = \varphi_{s(e)}(x) = \underline{\min}[\varphi_{r(e)}(x), \varphi_e(x) + \Phi_e(x)];$$

then $\{\varphi_{s(e)}\}_{e \in \mathcal{R}}$ is a measured set, and s is the desired procedure.

---

*Clause (E) involves an implicit use of the recursion theorem.

To prove the limit condition of the theorem, we need to show that Clause (E) is executed infinitely often for recursive $t = \varphi_e$. Suppose therefore that (E) is executed only finitely often, and let $< x,0 >$ be a stage after which there are no Clause (E) executions. We can assume without loss of generality that by stage $< x,0 >$ all pop 0 entries on the queue which are ever set to 1 have been set to 1, and furthermore $< x,0 >$ is large enough that an index for the empty function appears on the queue (its pop at stage $< x,0 >$ must be 1 or 2). Then for all $z \geq x$ and all i such that pop(i) = 0, $\Phi_i(z) \leq t(z)$, and so clause (D) and the presence of an index for the empty function on the queue guarantees that $\psi'(z)$ will be defined and $\psi'(z) \leq t(z)$. Therefore the percentage of arguments where $\psi'(z) \leq t(z)$ will eventually move above $1 - \frac{1}{P(< x,0 >)}$, and at that time, clause (E) will get executed.

To show that $\mathcal{F}(\varphi_e) = \mathcal{F}(\varphi_{s(e)})$ in the case where $\varphi_e \in \mathcal{R}_1$, notice that if i stabilizes on the queue, its pop cannot be 2. Using this observation it is easy to show that the classes are the same by using the techniques developed in theorem 2, and we omit the proof.

Our next theorem illustrates the striking difference between honesty procedures on $\mathcal{R}_1$ and honesty procedures on $\mathcal{P}_1$. Theorem 5 says that given any $t \in \mathcal{R}_1$, every procedure on $\mathcal{P}_1$ must map some program for t to an arbitrarily large honest name for $\mathcal{F}(t)$. We now construct a procedure on $\mathcal{R}_1$ which produces uniformly bounded honest names for $\mathcal{F}(t)$ independent of the program chosen for t. The procedure of Theorem 7 is an example of an honesty procedure on $\mathcal{R}_1$ which cannot be extended to an honesty procedure on $\mathcal{P}_1$.

Theorem 7. There is an honesty procedure on $\mathcal{R}_1$, s, with the property that for every recursive t, there is a $b \in \mathcal{R}_1$ such that if $\varphi_e = t$, then $\varphi_{s(e)} \leq b$ (a.e.).

Proof. Let s' be any honesty procedure on $\mathcal{R}_1$ with the property that if φ_e is total, then $F(\varphi_e) = F(\varphi_{s(e)})$. Say that $\varphi_e \overset{n}{=} \varphi_{e'}$ if after n steps of the dovetailed computation of φ_e and $\varphi_{e'}$, φ_e and $\varphi_{e'}$ have not differed on any argument. Define

$$\varphi_{s(e)}(x) = \underline{\min}[\varphi_{s'(e)}(x), \varphi_{e'}(x) + \Phi_{e'}(x) \mid e' \leq e, \varphi_{e'} \overset{x}{=} \varphi_e]. \quad (*)$$

$\{\varphi_{s(e)}\}_{e\in\mathbb{N}}$ is a measured set since both $\{\varphi_{s'(e)}\}_{e\in\mathbb{N}}$, and $\{\varphi_e + \Phi_e\}_{e\in\mathbb{N}}$ are measured. Furthermore $F(\varphi_{s(e)}) = F(\varphi_e)$. Suppose $\Phi_i \leq \varphi_e$(a.e.). Then for sufficiently large x those $e' \leq e$ which compute functions which differ from e will be omitted from the expression (*) for $\varphi_{s(e)}$. From then on if $\varphi_{e'}(y)$ converges for any $e' \leq e$, $\varphi_{e'}(y)$ will equal $\varphi_e(y)$. Therefore since $\Phi_i \leq \varphi_{s'(e)}$(a.e.), $\Phi_i \leq \varphi_{s(e)}$(a.e.). If on the other hand $\Phi_i > \varphi_e$ (i.o.), then $\Phi_i > \varphi_{s'(e)}$(i.o.), and since $\varphi_{s'(e)} \geq \varphi_{s(e)}$ everywhere, $\Phi_i > \varphi_{s(e)}$(i.o.).

Let e' be the least index for $\varphi_e \in \mathcal{R}_1$. Then for every $\varphi_i = \varphi_e$, $\varphi_{s(i)} \leq \varphi_{e'} + \Phi_{e'}$. (a.e.). □

Our last theorem shows that every class with a monotone name has a monotone honest name, settling a question raised in [4].

Theorem 8. There is a $g \in \mathcal{R}_2$ such that for every monotone recursive $t \geq \lambda x[x]$ there is a g-honest monotone recursive t' such that $\mathcal{F}(t) = \mathcal{F}(t')$.

Proof. Our construction will again follow the lines of Theorem 2. However, t' will be total and monotone whenever t is total, and so $\mathcal{F}(t)$ may differ from $\mathcal{F}(t')$ in the case where t is not monotone. Define $\varphi_{\sigma(i)}(x) = \max_{z\leq x}[\Phi_i(z),x]$. $\{\varphi_{\sigma(i)}\}_{i\in\mathbb{N}}$ is a measured set. Moreover for monotone $t \geq \lambda x[x]$ we have that $\Phi_i \leq t$ (a.e.) $\Leftrightarrow \varphi_{\sigma(i)} \leq t$ (a.e.). Let $t = \varphi_e$.

Stage n. Let $\pi_1(n) = x$, $\pi_2(n) = y$

A) Set $q_n = n$; set $pop(q_n) = 1$

B) If $\Phi_e(x) = y$, then for $0 \leq i \leq n$ if $\varphi_{\sigma(i)}(x) > t(x)$, set $pop(i) = 1$.

C) If $t'(x)$ has already been defined at some previous stage, go to stage n+1; if $\exists z < x$ such that $t'(z) > y$, go to stage n+1; if $x \leq y$, go to stage n+1.

D) Find least $i \leq n$ (if any) such that

1) $pop(q_i) = 1$

2) $\varphi_{\sigma(q_i)}(x) > y$

3) $(\forall j < i)[pop(q_j) = 0 \rightarrow \varphi_{\sigma(q_j)}(x) \leq y]$

If i exists, set $t'(x) = y$, set $pop(q_i) = 0$, and move q_i to the bottom of the queue.

E) If (D), find greatest $z < x$ such that $t'(z)$ has already been defined; set $t'(w) = y$ for $z < w < x$. Go to stage n+1. □

The procedure yields, for each e, a partial function $\varphi_{s(e)}$. Moreover $\{\varphi_{s(e)}\}_{e\in\mathbb{N}}$ is a measured set: to test $\varphi_{s(e)}(x) = y$, merely run the procedure through the first $< x,y >$ stages and check to see if $\varphi_{s(e)}(x)$ is defined to be y at one of these stages. Clause (C) guarantees that $\varphi_{s(e)}(x)$ can never be set equal to y after stage $< y,y >$.

If t is total, then t' will be total and monotone by clause (C) and the fact that Clause (D) must be executed infinitely often. If $t \geq \lambda x[x]$, then pop stability analysis and the fact that each $\varphi_{\sigma(i)}$ is monotone shows that for every i, $\varphi_{\sigma(i)} \leq t$ (a.e.) $\Leftrightarrow \varphi_{\sigma(i)} \leq t'$ (a.e.). But then for monotone t we have

$$\Phi_i \leq t \text{ (a.e.)} \leq \varphi_{\sigma(i)} \leq t \text{ a.e.} \Leftrightarrow \varphi_{\sigma(i)} \leq t' \text{ (a.e.)}$$
$$\Leftrightarrow \Phi_i \leq t' \text{ (a.e.)}. \qquad \square$$

Corollary. There is an honesty procedure on $\mathcal{R}_1$, s^*, such that for every recursive monotone $t \geq \lambda x[x]$, if $\varphi_e = t$ then $\varphi_{s^*(e)}$ is monotone.

Proof. Let s be the procedure of Theorem 7, and let s' be any honesty procedure on $\mathcal{R}_1$. Define s^* as follows:

$$\text{Let } \varphi_{s^*(e)}(x) = \begin{cases} \varphi_{s'(e)}(x) & \text{if, within x steps it is discovered that } \varphi_e \text{ is not monotone, or within x steps it is discovered that } \varphi_e \leq \lambda x[x]; \\ \varphi_{s(e)}(x) & \text{otherwise.} \end{cases}$$

The first clause on the right is obviously recursive, and so $s^* \in \mathcal{R}_1$. If $\varphi_e \geq \lambda x[x]$ and is monotone, then $\varphi_{s^*(e)} = \varphi_{s(e)}$. Otherwise $\varphi_{s^*(e)} = \varphi_{s'(e)}$ (a.e.). Hence s^* is the desired honesty procedure on $\mathcal{R}_1$.

We remark that the lower bound $\lambda x[x]$ of the theorem and the corollary may be replaced by any slow-growing unbounded function. Borodin [2] shows that some lower bound is necessary, and thus our result is best possible.

Bibliography

1. Blum, M. "A machine-independent theory of the complexity of recursive functions", JACM 14 (1967), pp. 322-336.

2. Borodin, A. "Complexity classes of recursive functions and the existence of complexity gaps", JACM 19 (1972), pp. 158-174.

3. Trachtenbrot, B.A., "Complexity of algorithms and computations", Course notes, Novosibirsk U., Novosibirsk, Russia (1967).

4. McCreight, E.M. and A.R. Meyer, "Classes of computable functions defined by bounds on computation: Preliminary Report", Conf. Rec. ACM Symp. on Th. of Computing (1969), pp. 79-88.

5. McCreight, E.M. "Classes of computable functions defined by bounds on computation", Doctoral Thesis, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pa. (1969).

6. Bass, L. "Hierarchies based on computational complexity and irregularities of class determining measured sets" Doctoral Thesis, Purdue University (1970).

7. Bass, L. and P. Young, "Ordinal hierarchies and naming classes", JACM, to appear.

8. Meyer, A.R. and E.M. McCreight, "Properties of bounds on computation", Proc. of Third Ann. Princeton Conf. on Info. Sci. and Systems (1969), pp. 154-156.

9. Rogers, H. The Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York (1967).

10. Robertson, E.L. "Complexity classes of partial recursive functions", Proc. of the Third Ann. Symp. on Th. of Computing (1971), pp. 258-265.

11. Constable, R. "Upward and downward diagonalizations over axiomatic complexity classes", Cornell University, Department of Computer Science, Tech. Report (1969).

12. Hartmanis, J. and J. Hopcroft, "An overview of the theory of computational complexity", JACM 18 (1971), pp. 444-475.

13. Landweber, L. and E. Robertson, "Recursive properties of abstract complexity classes", JACM 19 (1972), pp. 296-308.

14. Constable, R., "The operator gap" JACM 19 (1972), pp. 175-183.

15. Meyer, A.R. and R. Moll, "Honest bounds for complexity classes of recursive functions", Proc. of the Third Ann. Switching and Automata Theory Symp., College Park, Md. (1972), pp. 61-66.

16. Young, P., "Easy constructions in complexity theory: speed-up and gap theorems", Purdue University, Computer Science Department, Tech. Report No. 57 (1971).

Chapter 3

An Operator Embedding Theorem for Complexity Classes of Recursive Functions

1. INTRODUCTION

Let $\mathcal{F}(t)$ be the set of functions computable by some machine using no more than $t(x)$ machine steps on all but finitely many arguments x. If we order the $\mathcal{F}$-classes under set inclusion as t varies over the recursive functions, then it is natural to ask how rich a structure is obtained. We show that this structure is very rich indeed. If R is any countable partial order and $\underset{\sim}{F}$ is any total effective operator, then we show that there is a recursively enumerable sequence of recursive machine running times $\{\Phi_{s(k)}\}_{k\in N}$ such that if jRk, then $\mathcal{F}(\underset{\sim}{F}(\Phi_{s(j)})) \subsetneq \mathcal{F}(\Phi_{s(k)})$, and if j and k are incomparable, then $\underset{\sim}{F}(\Phi_{s(j)}) < \Phi_{s(k)}$ on infinitely many arguments, and $\underset{\sim}{F}(\Phi_{s(k)}) < \Phi_{s(j)}$ on infinitely many arguments.

An interesting feature of our proof is that we avoid appealing explicitly to the continuity of total effective operators; indeed our proof follows directly from a single appeal to the recursion theorem.

Several investigators have considered this and related problems, and in Section 4 we briefly summarize these investigations and compare them to our own.

2. PRELIMINARIES

For notation from recursive function theory we follow Rogers [2].

For each $n \in N$, $\mathcal{P}_n$ stands for the partial recursive functions of n-variables, and $\mathcal{R}_n$ stands for the total recursive functions of n variables.

We use (a.e.) to denote "almost everywhere", which for our purposes stands for "all but finitely many". Similarly (i.o.) stands for "infinitely often".

Suppose $\{\varphi_0,\varphi_1,\ldots\}$ is a Godel numbering of $\mathcal{P}_1$. A <u>measure on Computation</u> [1] $\Phi = \{\Phi_0,\Phi_1,\ldots\}$ is a sequence of functions in $\mathcal{P}_1$ satisfying

1. $\forall i \in N\ [\mathrm{dom}(\varphi_i) = \mathrm{dom}(\Phi_i)]$
2. $\lambda ixy[\Phi_i(x) = y]$ is a recursive predicate.

If we think of our Gödel numbering in the usual one-tape Turing machine formalism, then

$\Phi_i(x)$ = "the number of steps in the computation of the i^{th} Turing machine on argument x" is a measure on computation.

Henceforth let Φ be some fixed measure on computation. Then we define for any total function t

$$F(t) = \{i \in N \mid \varphi_i \in \mathcal{R}, \text{ and } \Phi_i \leq t \text{ (a.e.)}\},$$

and

$$\mathcal{F}(t) = \{\varphi_i \mid i \in F(t)\}.$$

That is, F(t) is the set of (indices of) total machines which run in time t, and $\mathcal{F}(t)$ is the set of total functions computable within time t. $\mathcal{F}(t)$ is called a complexity class.

A sequence of partial functions $\Psi = \{\psi_0, \psi_1, \ldots\}$ is said to be an r.e. sequence of partial functions if $\lambda i x[\psi_i(x)] \in \mathcal{P}_2$.

The following theorem of Blum [1] shows that we can uniformly enlarge complexity classes $\mathcal{F}(t)$ if t is a sufficiently well-behaved function.

Theorem. (Compression Theorem) There is a $g \in \mathcal{R}_2$ such that for every $\Phi_i \in \mathcal{R}_1$, $\mathcal{F}(\Phi_i) \subsetneq \mathcal{F}(\lambda x g(x, \Phi_i(x))$. g is called a compression function for Φ.

An operator is a map which takes functions to functions; we write $\underset{\sim}{F}(f)(x)$ to mean the value of the operator $\underset{\sim}{F}$ applied to the function f, evaluated at x. An operator $\underset{\sim}{F}: D \subseteq \mathcal{P}_1 \to \mathcal{P}_1$ is called an effective operator if there is an $s \in \mathcal{R}_1$ such that $\underset{\sim}{F}(\varphi_e)(x) = \varphi_{s(e)}(x)$.

An effective operator $\underset{\sim}{F}$ is total effective if for every $f \in \mathcal{R}_1$, $\underset{\sim}{F}(f)$ is defined and $\underset{\sim}{F}(f) \in \mathcal{R}_1$.

3. THE EMBEDDING THEOREM

Theorem. Let $\underset{\sim}{F}$ be any total effective operator, and let R be any recursive countable partial order on N. Then there exists an r.e. sequence of recursive functions $p_0, p_1, \ldots p_n \ldots$ such that if jRk, then $\underset{\sim}{F}(p_j) < p_k$ (a.e.), and if j and k are incomparable, then $\underset{\sim}{F}(p_j) < p_k$(i.o.), and $p_k < \underset{\sim}{F}(p_j)$ (i.o.).

Proof. We assume without loss of generality that R orders N-{0} rather than N, and in addition that R contains kRO for each $k > 0$. Let $a_0 = < i_0, k_0 >$, $a_1 = < i_1, k_1 >$, ... $a_n = < i_n, k_n >$, ... be a recursive listing of all incomparable pairs in R such that if x and y are incomparable, then $< x, y >$ and $< y, x >$ both appear infinitely often in the list. As a technical convience we define $\underline{\max}[\phi] = 0$. Let $s \in \mathcal{R}_2$ be the s^1_1 function of the s-m-n theorem defined by the equation

$$\varphi_e(< x, y >) = \varphi_{s(e,x)}(y).$$

Define $\psi \in \mathcal{P}_2$ as follows:

$$\psi(e, <k, x>) = \begin{cases} 0 \text{ if } x < k \text{ or } \exists n < k \text{ such that } \Phi_e(< 0, n >) > x, & (1) \\ \underset{\substack{j \le x \\ jRk}}{\max} [\varphi_{s(e,j)}(x) + \underset{\sim}{F}(\varphi_{s(e,j)})(x)]) + & (2)(i) \\ [\varphi_{s(e,i_n)}(x) + \underset{\sim}{F}(\varphi_{s(e,i_n)})(x)], & (2)(ii) \\ \text{where } n = \mu m \le x[((m = 0) \underline{\text{and}} (x = k_0)) \underline{\text{or}} \\ [(m > 0) \underline{\text{and}} (k = k_m) \underline{\text{and}} [(\forall i \ (0 \le i \le m)) \\ (\exists z_i \le x) \text{ such that } (z_0 = k_0) \underline{\text{and}} \\ (z_{i+1} = z_i + \Phi_{s(e,k_i)}(z_i)) \underline{\text{and}} (z_m = x)]]], \text{ if} \\ \text{such an n exists and (1) is not true, and} \\ \underset{\substack{j \le x \\ jRk}}{\max} [\varphi_{s(e,j)}(x) + \underset{\sim}{F}(\varphi_{s(e,j)})(x)] \text{ otherwise.} & (3) \end{cases}$$

$\psi \in \mathcal{P}_2$ since all the test computations in clauses (1) and (2) are recursive by the second measure on computation axiom. By the recursion theorem there is an e such that $\psi(e, \langle k,x \rangle) = \varphi_e(\langle k,x\rangle)$; we apply the s-1-1 version of the s-m-n theorem to obtain $\psi(e, \langle k,x\rangle) = \varphi_{s(e,k)}(x)$. To simplify our notation we now suppress mention of e and write $p_k(x) = \varphi_{s(e,k)}(x)$. Similarly we write $\Phi_{p_k}(x)$ for $\Phi_{s(e,k)}(x)$. Our definition now becomes

$$p_k(x) = \begin{cases} 0 \quad \text{if } x < k \text{ or } \exists n < k \text{ such that } \Phi_{p_0}(n) < x, & (1) \\ \max\limits_{\substack{j \le x \\ jRk}} [p_j(x) + \underset{\sim}{F}(p_j)(x)]) + & (2)(i) \\ \qquad [p_{i_n}(x) + \underset{\sim}{F}(p_{i_n})(x)], & (2)(ii) \\ \quad \text{where } n = \mu m \le x[((m = 0) \text{ and } (x = k_0)) \text{ or} \\ \quad [(m > 0) \text{ and } (k = k_m) \text{ and } [(\forall i(0 \le i \le m))(\exists z_i \le x) \\ \quad \text{such that } (z_0 = k_0) \text{ and } (z_{i+1} = z_i + \Phi_{p_{k_i}}(z_i)) \text{ and} \\ \quad (z_m = x)]]], \text{ if such an n exists and (1) is not} \\ \quad \text{true, and} \\ \max\limits_{\substack{j \le x \\ jRk}} [p_j(x) + \underset{\sim}{F}(p_j)(x)] \quad \text{otherwise.} & (3) \end{cases}$$

We first establish that at most finitely many of the functions $\{p_k\}_{k \in N}$ can be non-total. Suppose $p_k(x)$ diverges. Since p_0 is defined by (3) at all arguments, $p_0(x)$ must diverge, and so by (1) $p_j(x) \equiv 0$ for all $j > x$.

We now prove that for all k p_k is total.

Say that a_n is serviced at x if $p_{k_n}(x)$ is defined by (2), and if n is the least $m \leq x$ satisfying the body of (2) in the definition of $p_{k_n}(x)$. We allow the possibility that $P_{k_n}(x)$ may diverge. If a_n is serviced at x, (2) guarantees that $x = z_n = \sum_{i=1}^{n-1} z_i + \Phi_{p_{k_i}}(z_i)$, and so a_n is serviced at no other argument. Moreover, if a_n is serviced at x and $p_{k_n}(x)$ diverges, then for $n' > n$ $a_{n'}$ will never be serviced, since $a_{n'}$ is serviced at y only when y bounds the computation of $\Phi_{p_{k_n}}(x)$.

Let k be an R-minimal element in the finite set $\{k' \mid p_{k'}$ non-total$\}$. Then if $p_k(x)$ diverges, it must do so because of (2)(ii). That is, a_n is serviced at x for some n, and p_{i_n} must be non-total.

But suppose $p_{i_n}(y)$ diverges by an instance of (2)(ii) for some y. This means that $i_n = k_j$ for some j and a_j is serviced at y. If $j < n$, then y must equal z_j, but since a_n is serviced x, $\Phi_{p_{k_j}}(z_j) < x$ and hence $p_{k_j}(z_j)$ must converge. If $j > n$, then since a_n is serviced at x and $p_k(x)$ is assumed to diverge, a_j is never serviced. Moreover j cannot equal n, for then i_n would equal k_n. Hence p_{i_n} must be non-total because of (2)(i) or (3), and so some function $p_{i'}$ such that $i' R i_n$ is non-total.

Let i be R minimal among $\{i' \mid i'R\, i_n$ and i' non-total$\}$. Then p_i must be non-total by an instance of (2)(ii), say at argument y. Hence $i = k_j$ for some j, and a_j must be serviced at $y = \sum_{m=0}^{j-1} z_m + \Phi_{p_k}(z_m)$. If $j < n$, $p_{k_j}(y)$ must converge since a_n is serviced at x by assumption; and if $j = n$, then i_n and k_n are comparable, a contradiction. Furthermore if $j > n$, then a_j will never be serviced. Hence p_i is total, and we conclude that for every k $p_k \in \mathcal{R}_1$.

If jRk, then $\underset{\sim}{F}(p_j)(z) \leq p_k(z)$ for all $z \geq m_0 = \underline{\max}[k,j,\Phi_{p_0}(0), \Phi_{p_0}(1), \ldots \Phi_{p_0}(k-1)]$.

If j and k are incomparable, then $< j,k > = a_{n_0}, a_{n_1}, \cdots a_{n_q}, \cdots$ for some infinite sequence $n_0 < n_1 < n_2 \cdots n_q \cdots$.

For arguments $z \geq m_0$ $p_k(z)$ is defined by (2) or (3). Since the sequence of z_i's is strictly increasing, there is an i_0 such that for $i > i_0, z_i \geq m_0$. At those arguments z_i for $i > i_0$, $i = n_q$, $p_k(z_i)$ will be defined by clause (2) and $p_k(z_i) > F(p_j)(z_i)$. A symmetric argument shows that $p_j > F(p_k)$(i.o.), and the theorem is proved.

Corollary. Let $\underset{\sim}{F}$ be any total effective operator, and let R be any countable partial order on N. Then there exists an r.e. sequence of recursive measure functions $\Phi_{r(0)}, \Phi_{r(1)}, \ldots$ such that if jRk, then $\underset{\sim}{F}(\Phi_{r(j)}) < \Phi_{r(k)}$ (a.e.) and $\mathcal{F}(\underset{\sim}{F}(\Phi_{r(j)})) \subsetneq \mathcal{F}(\Phi_{r(k)})$, and if j and k are incomparable, then $F(\Phi_{r(j)}) < \Phi_{r(k)}$(i.o.), and $\underset{\sim}{F}(\Phi_{r(k)}) < \Phi_{r(j)}$(i.o.).

Proof. Mostowski [3] has shown that there is a countable partial order R^* into which any countable partial order may be embedded. Moreover, Sacks [4] has shown that R^* is recursive.

We assume without loss of generality that $\underset{\sim}{F}$ is at least as large as the identity operator, and that the compression function for Φ, g, is strictly increasing in its second argument. Blum [1] has shown that there is an $h \in \mathcal{R}_2$ such that for all i $\varphi_i(x) \leq h(x, \Phi_i(x))$(a.e.). We assume that h is strictly increasing in its second argument. To prove the corollary, apply the theorem to R^*, rewrite clause (2) as

$$\max_{\substack{j \leq x \\ jRk}} [p_j(x) + h(x, g(x, \underset{\sim}{F}(\Phi_{p_j})(x)))] + [p_{i_n}(x) + h(x, g(x, \underset{\sim}{F}(\Phi_{p_{i_n}})(x)))],$$

and we rewrite clause (3) as

$$\max_{\substack{j \leq x \\ jRk}} [p_j(x) + h(x, g(x, \underset{\sim}{F}(\Phi_{p_j})(x)))].$$

It is easy to see that the theorem goes through as before, and the monotonicity restrictions on g and h guarantee that the functions $\{\Phi_{p_k}\}_{k \in N}$ satisfy the corollary.

4. RELATION TO OTHER WORK, AND OPEN PROBLEMS

McCreight [5] is the first investigator to prove an embedding theorem for subrecursive classes. He shows that any countable partial order can be embedded in the complexity classes ordered under set inclusion. However, his theorem is weaker than our results in that the functions of his partial order are "separated" by composition with a fixed recursive function, whereas our functions are separated by a total effective operator. In [6] Enderton also proves a universal embedding theorem for subrecursive classes. His notion of a subrecursive class is quite weak, however, and his result is an immediate corollary of McCreight's theorem.

Early work on the structure of subrecursive classes was done by Feferman [12], Meyer and Ritchie [7], and Basu [8]. Feferman shows that dense chains exist for various notions of subrecursive classes. Meyer and Ritchie define what they call elementary honest classes, and they show the existence of dense chains and infinite anti-chains for such classes. Moreover, they are able to exhibit certain functions f such that dense chains of classes will exist between f and the iterate of f, $\lambda x[f^{(x)}(x)]$. Basu builds dense chains of subrecursive classes, where these classes are closed under the application of a fixed recursive operator.

Machtey [11] has announced universal embedding theorems for both the "honest" primitive recursive degrees and the "dishonest" primitive recursive degrees. Both of these theorems follow immediately from our results.

We also note that Alton [9] has independently announced our embedding theorem.

We leave open the question of the size of the functions in our embedding theorem. That is, given $\underset{\sim}{F}$, what is a reasonable upper bound on the size of p_0 in terms of $\underset{\sim}{F}$(recall that p_0 bounds all the functions $\{p_k\}_{k\in N}$ on all arguments).

The author wishes to acknowledge the generous assistance of Professor Albert R. Meyer in the conception and preparation of this paper.

REFERENCES

1. M. Blum, A machine-independent theory of the complexity of recursive functions, JACM 14, 1967, 322-336.

2. H. Rogers, Jr., Theory of recursive functions and effective computability, McGraw-Hill, 1967.

3. A. Mostowski, Über gewisse universelle relationen, Ann. Soc. Polon. Math. 17, 1938, 117-118.

4. G. Sacks, Degrees of unsolvability, Annuals of Math. Studies, No. 55, Princeton, N.J. 1963.

5. E. McCreight, Classes of computable functions defined by bounds on computation, Doctoral Dissertation, Carnegie-Mellon University, Department of Computer Science, 1969.

6. H. Enderton, Degrees of computational complexity, JCSS No. 6, 1972, 389-396.

7. A. Meyer and D. Ritchie, Classification of functions by computational complexity, Proc. of the Hawaii Internat'l Conf. on Sys. Sciences, 1968, 17-19.

8. S.K. Basu, On classes of computable functions, ACM Symp. on Theory of Computing, 1969, 55-61.

9. D. Alton, Operator embeddability in computational complexity, Notices of the AMS, 1972, A-763.

10. A. Meyer and P. Fischer, Computational speed-up by effective operators, JSL, No. 37, 1972, 55-68.

11. M. Machtey, Augmented loop languages and classes of computable functions, JCSS, to appear.

12. S. Feferman, Classifications of recursive functions by means of hierarchies, Trans. of the AMS, No. 104, 1962, 101-122.

Biographical Note

Robert Moll was born December 17, 1945 in Pittsburgh, Penna. He attended public school in Pittsburgh. As an undergraduate he attended Carnegie Institute of Technology, spending his junior year at the University of Vienna. After two years of graduate school at Carnegie, he transferred to MIT. He now lives at the Lee street commune in Cambridge, Massachusetts. In a few weeks he will be married to Rachel Folsom, the famous artist.

He has recently accepted a position as an assistant professor at the University of Massachusetts at Amherst in the Department of Computer and Information Science.

This empty page was substituted for a blank page in the original document.

CS-TR Scanning Project
Document Control Form

Date : 2/15/96

Report # LCS-TR-110

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 96 (102-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:
☒ DOD Form ☐ Funding Agent Form ☒ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number): FOLLOWS LAST PAGE (95)

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:
IMAGE MAP: (1-96) UNNUMBERED TITLE PAGE, 2-95 UNNUMBERED BLANK
(97-102) SCANCONTROL COVER, DOD, TRGT'S (3)

Scanning Agent Signoff:
Date Received: 2/15/96 Date Scanned: 2/28/96 Date Returned: 2/29/96

Scanning Agent Signature: Michael W. Cook

BIBLIOGRAPHIC DATA SHEET	1. Report No. NSF-OCA-GJ34671 - TR-110	2.	3. Recipient's Accession No.

4. Title and Subtitle

Complexity Classes of Recursive Functions

5. Report Date **Issued June 1973**

6.

7. Author(s) **Robert Moll**

8. Performing Organization Rept. No. **MAC TR-110**

9. Performing Organization Name and Address

PROJECT MAC; MASSACHUSETTS INSTITUTE OF TECHNOLOGY,
545 Technology Square, Cambridge, Massachusetts 02139

10. Project/Task/Work Unit No.

11. Contract/Grant No. **NSF-GJ-34671 and ONR-N00014-70-A-0362-0001**

12. Sponsoring Organization Name and Address

Associate Program Director
Office of Computing Activities
National Science Foundation
Washington, D. C. 20550

13. Type of Report & Period Covered: **Interim Scientific Report**

14.

15. Supplementary Notes

16. Abstracts

In part one we develop the properties of honest functions and measured sets as defined by Blum, Meyer and McCreight. An improved proof of the "honesty" theorem is given and several open problems are solved.

In part two we prove an operator embedding theorem for complexity classes of recursive functions.

In part three we give an extensive survey of research on subrecursive hierarchies.

17. Key Words and Document Analysis. 17a. Descriptors

Subrecursive hierarchy
Honest function
Complexity class
Universal embedding

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Unlimited Distribution Write Project MAC Publications	UNCLASSIFIED	96
	20. Security Class (This Page) UNCLASSIFIED	22. Price

FORM NTIS-35 (REV. 3-72) THIS FORM MAY BE REPRODUCED USCOMM-DC 14952-P72

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 2/28/1996

M.I.T. Libraries
Document Services