F. W. KROON

The Intrinsic Difficulty of Recursive Functions

Abstract. This paper deals with a philosophical question that arises within the theory of computational complexity: how to understand the notion of INTRINSIC complexity or difficulty, as opposed to notions of difficulty that depend on the particular computational model used. The paper uses ideas from Blum's abstract approach to complexity theory to develop an extensional approach to this question. Among other things, it shows how such an approach gives detailed confirmation of the view that subrecursive hierarchies tend to rank functions in terms of their intrinsic, and not just their model-dependent, difficulty, and it shows how the approach allows us to model the idea that intrinsic difficulty is a fuzzy concept.

Key words: recursive functions, computational complexity, subrecursive hierarchies.

Introduction

Thanks to the massive evidence in favour of the Church-Turing thesis, we are accustomed to thinking of effectiveness as an absolute or intrinsic notion: a property of functions and sets rather than of algorithmic ways of representing functions and sets.[1] What about the notion of difficulty? Here matters look less promising. We may think that effective functions and problems that cannot be computed or solved when applied to small arguments, even if all the space and time of the physical universe is at the disposal of the fastest computer that physics can allow for, must surely count as difficult for any agent and in any situation. But it is not hard to convince oneself that while effectiveness is a logico-mathematical absolute, the same is not true of such a notion of difficulty. From the point of view of possible universes where space and time are infinite and matter continuously replenished, difficulty of this kind will not seem such an important notion.

[1] For a good summary, see [9]. [26] contains a sensitive discussion of the extent to which the Church-Turing Thesis admits of proof.

Presented by **Jan Zygmunt**; *Received* August 8, 1994;

Studia Logica **56**: 427–454, 1996.

This paper is about absolute or intrinsic notions of (comparative) difficulty that are not seriously subject to such complaints.[2] The problems facing such notions are formidable. Consider again the main body of evidence favouring Church's Thesis: the fact that a host of different ways of delineating a class of effective or computable functions all happen to pick out the same class of functions. In the case of the notion of difficulty, however, we usually have in mind some designated way of delineating this class. The problem that then inevitably arises is that any fine-grained classification of functions in terms of the complexity of associated definitions or algorithms may really be quite parochial even though there is nothing parochial about the class of all the functions thus classified. This is so even if we rank functions in terms of their most 'efficient' associated definitions or algorithms (assuming these exist), simply because what counts as an 'efficient' definition or algorithm itself depends crucially on the chosen way of delineating computability.

Thus consider the various familiar operations that define the class of primitive recursive functions, and suppose we classify primitive recursive functions according to the number of times such operations are applied in their least derivation. But such a classification depends crucially on the choice of basic operations, with certain operations such as primitive recursion looking decidedly more 'complex' than others. In addition, there is no way of extending this procedure to all recursive functions, and no guarantee that such an operation-based classification of the (primitive) recursive functions will correspond to the sort of classifications we get once we move to computation-based characterizations of the recursive functions (e.g., via Turing machines).

In turn, computation-based characterizations have their own problems of non-invariance. Such characterizations will often yield somewhat different classifications depending on the computational model used (one-tape, multi-tape; one-dimensional, multi-dimensional; and that is in the case of Turing-machines alone) as well as the type of resource being counted (the number of tape-cells used up and the number of steps taken, for example). So on the computational approach too there is a danger of a lack of invariance that

[2] Alan Cobham ([8]) was the first to talk of intrinsic difficulty, and he introduced many of the important questions in this area. Note also the rather different use of the word 'intrinsic difficulty' in some other places. Thus in Stockmeyer and Chandra [27], 'intrinsic difficulty' refers to the obstacles to computation imposed by certain unalterable properties of the universe: its resource-limits in terms of space and time. The term is also sometimes used for the idea that a *function's* difficulty should be independent of the algorithms commonly used for computing it, and should only reflect the fastest algorithms for computing that function — an idea which still leaves the idea of 'intrinsic difficulty' somewhat dependent on the machine-model used for running the algorithms (see Glymour [11], p. 329).

threatens the notion of *intrinsic* comparative difficulty.

Is there a way out? In the case of the structural approach, these worries look particularly devastating. Thus consider the following version of the idea: tie intrinsic structural difficulty to place in a chosen subrecursive hierarchy that is more sensitive to what operations should be primitive. Thus:

(i) f is *intrinsically more difficult* than g (in the structural sense) if g belongs to a lower level than f in a sufficiently fine grained subrecursive stratification of functions.

Because classes in subrecursive hierarchies tend to be closed under crucial operations, this suggests that functions higher up can only be generated using structurally more complicated patterns of generation, or structurally more complicated functions, rather than just more applications of some arbitrarily chosen set of operations; that is the sense in which (i) counts them as intrinsically more difficult than functions lower down.[3] Thus put, the structural approach faces the problem that there are many subrecursive hierarchies: which do we choose, and why these? To suppose that we should go for the finest subrecursive stratification possible suggests that such a notion makes independent sense, but that is surely far from clear.[4] In addition, there is the problem that subrecursive classifications defined over *standard* well-orderings over codes for recursive ordinals tend to miss many recursive functions, so that we are left unable to place an informative classification on the functions missed out. On the other hand, subrecursive classifications that fudge even slightly, for example by allowing all elementary-recursive well-orderings, quickly fall foul of what is sometimes known as the Collapsing Phenomenon, which predicts that every recursive function will suddenly appear at a low level (ω or ω^2).[5]

[3]Thus Calude ([5], p. 78) talks of the Grzegorczyk hierarchy as one that classifies the primitive recursive functions according to 'intrinsic difficulty'.

[4]It might be thought that sense could be made of the idea of the 'finest' subrecursive stratification possible by considering the so-called 'slow growing hierarchy' $\{G_\alpha\}$ discussed by S. S. Wainer and others (see, for example, Cichon and Wainer [6]). The G_α are broadly defined as follows:
$G_0 =$ constant 0,
$G_{\alpha+1} = G_\alpha + 1$,
$G_\lambda = \text{Diagonal}(G_{\lambda_x})_{x<\omega}$.
But while the slow growing hierarchy may provide a good way of measuring computational complexity, with the complexity of a function measured by the least α such that the function is computable via G_α-bounded time or space, it doesn't give us a purely structural account of the complexity of functions apart from the G_α. For the latter we also need operations to generate all the other functions that we want to assess in purely structural terms.

[5]One of the classic papers here is Feferman [10].

The second, computational approach may seem to offer more hope. The most obvious way out of the quandary mentioned is to try something like the following:

(ii) f is *intrinsically more difficult* than g (in the computional sense) if for all algorithm-types P and all resource-types R,[6] there is a P-algorithm A for g such that, for any P-algorithm B for f, B uses up more resources of type R than A does when applied to x as input (for almost all x, or almost everywhere — that is, for all except a finite number; I abbreviate this to 'a.e.')

But this characterization in turn raises the question of how we are to understand the general concept of an algorithm-type and of a resource-type, given that (ii) appeals to *all* algorithm-types and resource-types. More particularly, it raises the question of how we are to understand these general concepts in a way that allows different functions to have different levels of difficulty according to (ii). Questions like these do not bother those who work in low-level complexity theory because of their willingness to work with selected algorithm-types and resource-types. They bother us, however, because of our logico-philosophical interest in the notion of *intrinsic* comparative difficulty.

The approach I shall take in this paper is the following. Given the problems facing a general structural notion of difficulty, I shall not develop (i) further, although the connection between subrecursive classifications and intrinsic difficulty will continue to interest us and will in fact be one of the main themes of this paper (see especially section 5). I shall instead describe a way of understanding (ii) that satisfies the guiding constraint of generality implicit in talk of all algorithm- and resource-types, and yet is not so general that it prevents us from achieving rankings of functions according to intrinsic computational difficulty. Here I rely on the co-relative idea of a reasonable or *natural* measure of difficulty: intrinsicness of difficulty on this account has to do with a certain kind of invariance across natural measures. The abstract basis of this account is sketched in sections 1 to 3, while section 4 looks at the idea in more concrete terms. In section 5 the account is complicated by introducing the view that intrinsic difficulty is a fuzzy notion since the idea of a natural measure is a fuzzy notion. I show that even so familiar hierarchies of classes of functions rank functions in a highly intrinsic way, a way that holds good for a large number of ways of resolving the vagueness in the description 'natural measure'. This result thus demonstrates a degree

[6] Alternatively, R could be a fixed resource-type such as number of steps used in the course of a computation, with intrinsic difficulty being relative to resource-type used.

of convergence between the structural and computational approaches to intrinsic difficulty. While neither this result nor the other results in the paper are particularly difficult to prove or answer outstanding *technical* problems in the theory of complexity, they seem inherently interesting, and should thereby help to confirm the methodological virtues of the paper's approach to what remains a puzzling philosophical question.

1. Intrinsic difficulty and measures of computational complexity

Account (ii) above talked of algorithm-types P and resource-types R in general. The most productive approach to this schematic idea is Manuel Blum's axiomatic approach which studies implications of a purely recursion-theoretic way of defining measures of computational complexity, one that makes no reference to particular algorithmic languages P or particular resource-types R ([4]). Such an approach appears the best place to start if something akin to (ii) is indeed to provide us with an understanding of differences in intrinsic difficulty between functions.

As usual, we define complexity measures as follows:

DEFINITION 1.1 Let $(\phi_i)_{i\in N}$ (where N is the set of non-negative integers) be an acceptable numbering of the unary partial recursive (p.r.) functions.[7] A sequence $\Phi = (\Phi_i)_{i\in N}$ of p.r. functions is called a *complexity measure* or *Blum measure* (with respect to the acceptable numbering $(\phi_i)_{i\in N}$) if the following two axioms are satisfied:

(a) $\phi_i(x)$ is defined iff $\Phi_i(x)$ is defined

(b) $\Phi_i(x) = y$ is a recursive predicate in i, x and y.

It is easily verified that the usual ways of counting resources used (such as the number of steps taken by Turing Machine programs or Random Access Machine programs, the number of tape-cells used during the course of (halting) Turing Machine computations, and so on, all correspond to measures of complexity. And this suggests the following manoeuvre. Why not reformulate account (ii) above by quantifying over all measures of complexity rather than, less precisely, over all algorithm-types P and resource-types R? But this won't do, obviously. We then sacrifice the ability to grade functions according to their intrinsic difficulty, for given any recursive function there are infinitely many measures that assign it zero complexity everywhere. Hence

[7] See, for example, Rogers [24]. Note that measures are defined relative to some acceptable numbering of the partial recursive functions.

any two recursive functions are identical in intrinsic difficulty according to this way of interpreting (ii).

Consider instead:

(iii) f is *intrinsically more difficult* than g iff the following is true for all $\Phi \in \mathbf{Na}$, where **Na** is the class of 'natural' measures: (a) given any index i for f there is an index j for g such that $\Phi_j(x) \leq \Phi_i(x)$ a.e., while (b) there is an index j for g such that for all indices i for f, $\Phi_i(x) > \Phi_j(x)$ for infinitely many x.

(This definition improves on (ii) in so far as part (a) also embodies an account of g's being intrinsically no more difficult than f; when this holds in both directions, f and g are of the same intrinsic difficulty.)

What we should aim for, it seems, is some reasonable definition of the class **Na** of 'natural' measures. But what counts as reasonable? Much work has been done on this problem (see, e.g., [2] and [13]). Arguably the most promising approach is to agree on 'natural' computational models and 'natural' resource-types first, and then to develop constraining axioms that capture crucial elements of these 'natural' models and resource-types. Unfortunately, however, no agreement on what is to count as natural is in sight. Here are two approaches:

(a) *concentrate on measures that count resources used in the course of Turing-machine-type computations (Turing machine 'path measures')*

(b) *also acknowledge step-counting measures based on flowchart programs (or random access machine programs), conceptualized in terms of instructions that allow operations on numbers stored at register-addresses (both 'test' operations '$P_{j_1,\ldots,j_n}$' and 'assignment' operations '$i := F_{j_1,\ldots,j_n}$'), where no a priori bound is placed on the kind of recursive predicates and functions allowed to interpret P and F, and where any such instruction counts as a simple instruction, effectively requiring unit time.*

These two approaches are quite different. (a), with its emphasis on symbol-manipulation, considers the difficulty of mathematical objects from the point of view of implementation of appropriate programs on a computing device that manipulates 'bits' rather than numbers. (b) emphasizes structural and hierarchical features of computing devices. Both approaches have merit. (a) is committed to a certain strong kind of computational 'nominalism', while (b) captures abstract 'inductive' features of computations (see, for example, [2] and [29], section 2.2).

If we accept (b), then certain proposals based on (a) need to be rejected. In an important paper of some years ago, Theodore Baker appealed to flowchart measures in order to cast doubt on a number of properties variously proposed as further constraints on Blum measures: properties like 'finite invariance' and 'density' ([2]). Still, (b) presents us with a problem from the point of view of the present paper. Nothing has been said about the nature of the predicates and functions that are allowed to feature in flowchart programs; in fact, from the point of view of the abstract framework these may be arbitrarily complex. One effect of this is that the recursive functions become non-gradable under the proposed account of intrinsic difficulty (iii). Thus suppose that we are given an interpretation J of the predicate and function names in flowcharts, and suppose that the resulting enumeration of programs yields an acceptable numbering of the p.r. functions. We can define a flowchart measure based on J as a measure that counts each instruction used in a computation based on such a program as taking unit time.[8] It is now clear that for any (unary) recursive function f, there is a flowchart measure that assigns to f a constant complexity-function $\lambda x[k]$ for some small k (viz., flowchart measures for flowchart programs that tolerate '$i := Fj$' as a simple assignment instruction, where 'F' is interpreted as f). Hence if f is more difficult than g according to one flowchart measure, it is at least as easy as g according to another. Counting all flowchart measures as natural, this makes grading impossible.

A weaker view is to acknowledge degrees of naturalness where flowchart measures are concerned: if only *easy* (or perhaps computationally constructive) functions and predicates are allowed to interpret the function- and predicate letters 'F' in simple test and assignment instructions, then the resulting flowchart measure is natural to a *high* degree, whereas if relatively complex functions are allowed to interpret 'F', then the ensuing flowchart measure is natural to at most a relatively *low* degree. Larry Stockmeyer, for example, thinks that no reasonable model should do an unrealistic amount of computation in one step, for example adding a number of length 2^n ([28], p. 10), although he doesn't say what is to count as realistic.

In this paper I want to bypass the question of how best to give an intensional characterization of naturalness of measures in terms of additions to the familiar Blum axioms. Like many others, in fact, I am not convinced that one can rule out all obviously pathological measures by introducing further structural constraints of this kind. Following Theodore Baker, I believe it

[8] One intermediate option sometimes taken in the case of ordinary random access machine programs is to assign a logarithmic cost to every executed instruction, equal to the sum of the lengths of all data manipulated implicitly or explicitly by the instruction. See [29] for a summary of results.

may be more useful to concentrate on the notion of a computational model:

> The naturalness of a measure seems unavoidably connected to the existence of a corresponding model of computation, with recognizable steps and intermediate results related in some way to the results of the computations. ... [E]ach step counted should have a recognizable 'effect' and produce a partial result (via a recognizable transformations of a previous result) in any step measure. ([2], p. 22)

Rather than develop this sort of approach directly, however, I shall develop an intermediate extensional approach, based on the following ideas. Beginning with a paradigmatically natural measure (defined on a paradigmatically natural model of computation) such as the step-counting measure T associated with a one-tape Turing machine model using binary code, we opt for a judicious choice of a class S of simulation-bounds or 'overheads' b such that the resources $M_f(x)$ of time (space, etc.) needed for computing an arbitrary partial recursive function f on argument x on some other natural computational model amount to no more than $b(T_g(x))$ for some bounding function b in S, where $T_g(x)$ is the time needed for computing $g(x)$ according to T, and such that $T_f(x)$ similarly amounts to no more than $b'(M_f(x))$ for some b' in S. We say that T and M are S-similar in that case. (A formal definition appears below.) Using measure T, we then disregard 'slight' differences — up to the size of bounds in S — between the computation times of two partial recursive functions, and we decide that f is more difficult than g in this fudged S-dependent sense only when f takes a lot more in the way of resources to compute than g does (roughly: when there is a way of computing g such that each way of computing f on x takes more than $b(T_g(x))$ steps for any simulation-bound b in S infinitely often, while the converse doesn't hold). If we understand 'f is more difficult than g' in this way, it is easily seen that the assessment 'f is more difficult than g' can be made highly measure-independent, holding true for all measures which are S-similar to T, and hence, in virtue of the way we choose S, for many, perhaps all, measures of difficulty that we are prepared to regard as *natural* measures. In short, once f is more S-difficult in this sense than g, then it may also be the case that f thereby becomes *intrinsically* more difficult than g in the sense of proposal (iii).

Such an approach may seem disappointingly circular. For how do we decide in the first place which simulation bounds are liberal enough to allow us to capture all 'natural' measures? But the situation is not nearly as bleak as it appears. There are a large number of paradigmatically natural measures which we know ought to belong. In addition, it may be easy to see that any

measures not able to be simulated within the given bounds cannot rest on any recognizable variation of familiar computational models, so that there may be good a posteriori reasons for excluding them. This is presumably why Stockmeyer thinks that 'a TM can simulate any reasonable model of computation with at most a polynomial increase in time or space' ([28], p. 10), and why van Emde Boas proposes an Invariance Thesis according to which "'[r]easonable" machines can simulate each other within a polynomially bounded overhead in time [and a constant-factor overhead in space]' ([29], p. 5).[9]

But there is another reason why I have adopted the extensional approach. If I am right in suggesting that there may be a degree of vagueness involved in the notion of a natural measure, we need a way of describing and assessing the parameters of vagueness. As we shall see in section 5, the extensional approach turns out to be particularly well-suited to this task.

2. S-invariant theories of complexity

The approach in this paper is the extensional approach just described. Elements of it were described some time ago, but without the philosophical overtones I have introduced, in a short paper by Michael Arbib and Manuel Blum ([1]). Its elaboration requires not just the abstract concept of a measure of complexity but also the concept of what I shall call a *smear monoid*, so called because relativizing complexity theories to these monoids results in a smearing of computation times (disregarding incremental differences between them, the permitted level of disregard being determined by the smear monoid) and a consequent enlarging of the number of measures over which the induced complexity ordering remains valid.

Here are some central definitions:

DEFINITION 2.1 A *smear monoid* S is a monoid of 2-variable recursive functions $\sigma : N \times N \to N$, increasing with respect to the second variable, with composition $*$ satisfying $\sigma * \sigma'(x,y) = \sigma(x, \sigma'(x,y))$, and with identity $e(x,y) = y$.

[9] In the case of both Stockmeyer and van Emde Boas, support for the polynomial bound is mainly in terms of known alternatives to Turing machine models. van Emde Boas also considers very fast parallel machines, and describes the evidence in favour of the Parallel Computation Thesis, according to which 'whatever can be solved in polynomially bounded space on a reasonable sequential machine can be solved in polynomially bounded time on a reasonable parallel machine, and vice versa' ([29], p.5). This makes it clear that 'reasonable' in his Invariance Thesis applies in the first instance to machine-types (e.g., sequential or parallel).

The complexity ordering induced by a complexity measure and smear monoids is formally described in the next definition.

DEFINITION 2.2 Let S be a smear monoid, Φ a complexity measure, and f and g p.r. functions. We say that

(1) *f is no more difficult than g with respect to (Φ, S)* ($f \leq_{\Phi,S} g$) if (a) the domain of $g \subseteq$ domain of f, and (b) for every index i satisfying $\phi_i = g$, there is an index j satisfying $\phi_j = f$ and a σ in S such that $\sigma(x, \Phi_i(x)) \geq \Phi_j(x)$ for almost all x in domain g;

(2) *g is more difficult than f with respect to (Φ, S)* ($f <_{\Phi,S} g$) if $f \leq_{\Phi,S} g$ but not $g \leq_{\Phi,S} f$; and

(3) *f is of the same difficulty as g with respect to (Φ, S)* ($f \cong_{\Phi,S} g$) if $f \leq_{\Phi,S} g$ and $g \leq_{\Phi,S} f$.

(I shall call $\leq_{\Phi,S}$, $\cong_{\Phi,S}$ and $<_{\Phi,S}$ **S-dependent complexity relations**, and the orderings they induce **S-dependent complexity orderings.** Note that both $\leq_{\Phi,S}$ and $<_{\Phi,S}$ are transitive relations.)

The next definition defines a notion of equivalence of measures based on the idea that different measures may induce the same S-dependent complexity ordering:

DEFINITION 2.3 Two complexity measures Φ and Ψ are *S-equivalent* ($\Phi \equiv_S \Psi$) if, for all p.r. functions f and g, $f \leq_{\Phi,S} g$ if and only if $f \leq_{\Psi,S} g$.

The relation $\equiv_S$ is obviously an equivalence relation. The set of all complexity measures S-equivalent to Φ will be designated $[\Phi]_S$. Since the S-dependent complexity ordering induced by Φ and S is invariant up to membership in $[\Phi]_S$, we call (Φ, S) an *S-invariant theory of complexity.*

The following definition suggests a useful criterion for membership in $[\Phi]_S$. Informally, two (step-counting, say) measures are S-similar if their underlying machine models can simulate each other with (time) overhead taken from S. Formally,

DEFINITION 2.4 Let Φ and Ψ be be complexity measures. Then

(1) $\Phi \leq_S \Psi$ if for every j there exists an i and a σ in S such that $\phi_i = \phi_j$ and $\sigma(x, \Psi_j(x)) \geq \Phi_i(x)$ a.e. in the domain of ϕ_j; and

(2) *Φ and Ψ are S-similar* ($\Phi \approx_S \Psi$) if both $\Phi \leq_S \Psi$ and $\Psi \leq_S \Phi$.

If follows almost immediately that S-similarity is a sufficient condition for S-equivalence. (It is easy to show that S-similarity is not a *necessary* condition of S-equivalence.)

Central to complexity theory is the notion of a complexity class. Arbib and Blum [1] did not introduce an appropriately relativized analogue of this notion, but such a notion is important to my development of the approach. Informally, if R is a complexity class for theory (Φ, S) then, whenever a recursive function f is in R, any recursive function that is no more difficult than f relative to theory (Φ, S) should also be in R.

DEFINITION 2.5 Let Φ be a complexity measure, t a total unary function, and S a smear monoid. Then $R_t^{(\Phi,S)}$ is a *complexity class with respect to theory* (Φ, S) if

$$R_t^{(\Phi,S)} = U_{\sigma \in S} R^{\Phi}_{\sigma * t}$$

(here $\sigma * t$ is just $\lambda x[\sigma(x, t(x)]$ and R_t^{Φ} is a complexity class in the usual sense i.e. the set of all total unary recursive f such that for some index i for f, $\Phi_i(x) \leq t(x)$ a.e.)

3. S-invariant analogues

Many of the most important results of axiomatic complexity theory apply also to theories of S-invariant difficulty. The fact that such theories inherit the rich structure uncovered in these results helps to confirm the naturalness of an approach that looks to the behaviour of classes of measures rather than single measures; more importantly, these theories thereby cast further light on the topic of intrinsic difficulty. In this section I provide analogues of a small number of important results of this type, largely chosen with an eye on what they tell us about intrinsic difficulty. For the most part the proofs involve routine modifications of existing proofs, and so for the most part I omit the proofs or provide only a brief sketch.

The first result simply states that there is a systematic way of generating functions that are more S-difficult than some given function. To begin with, let us say that a set F of k-variable total functions : $N^k \to N$ is **recursively bounded** if there is an k-variable recursive function f such that, given any t in F, $f \geq t$ for almost all members of N^k.

We can now show that:

THEOREM 3.1 *Given measure Φ and a recursively bounded smear monoid S, there is a recursive h such that $\phi_i <_{\Phi,S} \phi_{h(i)}$ for all i.*

SKETCH OF PROOF. Let t be a recursive bound for S, and given i define the p.r. function g as follows:

$$g(0) = 1;$$

and for $n > 0$,

$$g(n) = \begin{cases} \text{undefined if } \Phi_i(x) \text{ is undefined for some } 0 \leq x \leq n; \\ 0, \quad \text{if } \phi_k(n) = 1, \text{ where } k = \mu m[m < n \;\&\; \\ \qquad \Phi_m(n) \leq t(n, \Phi_i(n)) \;\&\; \\ \qquad \text{for every } x < n, \text{ if } \Phi_m(x) \leq t(x, \Phi_i(x)) \text{ then } g(x) = \phi_m(x)]; \\ 1, \quad \text{otherwise} \end{cases}$$

It can now be shown that every index j for g, we have $\Phi_j(x) > t(x, \Phi_i(x))$ at almost all x where ϕ_i is defined. (For easily adapted details, see, for example, Hartmanis and Hopcroft [14]; Calude [5].) Because the definition of g depends effectively on index i of ϕ_i, the theorem follows. ■

Note how this theorem bears on the topic of intrinsic difficulty: if the class of natural measures $\mathbf{Na} \subseteq [\Phi]_S$, then the construction of Theorem 3.1 shows that there is an effective way of generating functions intrinsically more difficult than given functions.

Blum's Compression Theorem, too, has an S-invariant analogue, which is worth describing because of its role in the hierarchy result at the end of this section. First we define a *measured* set:

DEFINITION 3.2 A *measured* set is a recursively enumerable[10] set of unary p.r. functions g_i for which the 3-place predicate $g_i(n) = m$ is recursive. (Clearly, the set $\{\Phi_i \mid i \in N\}$ of complexity functions of a given measure Φ is a measured set of functions.)

THEOREM 3.3 *Let $\{g_i \mid i \in N\}$ be a measured set, S a recursively bounded smear monoid, and Φ a measure. Then there is a 2-place recursive function r such that*

$$R_{g_i}^{(\Phi,S)} \subset R_{r*g_i}^{(\Phi,S)}.$$

The proof is once again a routine adaptation of the proof found in the literature. See, for example, [5], p. 239.

We now state an analogue of Borodin's Enumerability Theorem for complexity classes. This, and the Union Theorem to follow, play a central role in the proof of the hierarchy result at the end of the section.

[10] As always, we say that a set C of k-ary p.r. functions is *recursively enumerable* if there exists a total recursive function h such that $C = U_i \phi_{h(i)}^{(k)}$, where $(\phi_i^{(k)})$ is an acceptable enumeration of the k-ary p.r. functions.

THEOREM 3.4 *Let h and t be any recursive functions such that not only h but all functions that differ from h at finitely many arguments belong to R_t^Φ. If S is a recursively enumerable smear monoid then, for any recursive f such that $f(x) \geq t(x)$ for all x, the complexity class $R_f^{(\Phi,S)}$ is recursively enumerable.*

h is commonly specified to be the zero function in statements of Borodin's original theorem. For a proof of that theorem, see, for example, [14], or [5], 3.4.28. Theorem 3.4 can be established by a routine generalization of that proof. Note that recursive enumerability of associated complexity classes is often regarded as one of the halmarks of 'natural' measures ([2], [13]).

The next theorem is an analogue of McCreight and Meyer's Union Theorem. First, let us say that a set F of k-variable total functions is **self-bounded** if for every finite $F_0 \subseteq F$ there is a t in F such that $t(x_1...k_k) \geq [\max\{f(x_1,...,x_k) \mid f \in F_0\}]$ a.e.

THEOREM 3.5 *Let S be a recursively enumerable, self-bounded smear monoid and let T be a recursively enumerable, self-bounded set of (unary) recursive functions. For any complexity measure Φ there is a recursive t' such that*

$$U_{t\in T} R_t^{(\Phi,S)} = R_{t'}^{(\Phi,S)}$$

The proof is again a routine generalization of existing proofs (e.g., in [14]).[11]

The main result of the present section is a hierarchy result based on Theorems 3.3 to 3.5, as well as the following well-known result due to McCreight and Meyer:

THEOREM 3.6 (HONESTY THEOREM)[12] *For each Blum measure Φ there is a measured set $\{\phi_{s(i)} \mid i \in N\}$ such that if ϕ_i is total then $\phi_{s(i)}$ is total and $R_{\phi_i}^\Phi = R_{\phi_{s(i)}}^\Phi$.*

[11] Here is a quick sketch. Suppose $T = \{t_0, t_1, ...\}$ and $S = \{\sigma_0, \sigma_1, ...\}$. Without loss of generality, we assume for $i \geq j$ that $t_i \geq t_j$ and $\sigma_i \geq \sigma_j$ everywhere. With each index i, we associate an integer-valued label $g(i)$. The function t' can be constructed in stages (starting at stage zero) as follows: Stage x: Let $g(x) = x$ and let $A(x) = \{i \leq x \mid \Phi_i(x) > \sigma_{g(i)} * t_{g(i)}(x)\}$. If $A(x) = \emptyset$ then set $t'(x) = \sigma_x * t_x(x)$. If $A(x) \neq \emptyset$ then set $t'(x) = \min\{\sigma_{g(i)} * t_{g(i)}(x) \mid i \in A(x)\}$. For all $i \in A(x)$, set $g(i) = x$; now go to Stage $x+1$. We can now verify the following two claims, which jointly imply Theorem 3.5 (cf. [14]). Claim 1 For all indices i, $t' \geq \sigma_i * t_i$ a.e. Claim 2 If there exist indices r and k such that $\sigma_r * t' > \Phi_k$ a.e., then there exists index i such that $\sigma_i * t_i > \Phi_k$ a.e. ∎

[12] The term 'Honesty Theorem' captures the fact that, given a measured set M, there is a total recursive function g such that functions in M are g-honest, where f is g-honest if there is an index i for f such that $\Phi_i(x) \leq g(x, f(x))$ for almost all x in the domain of f (i.e., f's complexity 'honestly' reflects its size, modulo g).

For a proof, see [5], pp. 241ff., for example.

Using the Honesty Theorem and the Union Theorem, it can now be shown that:

FACT: Let Φ be a measure and S a recursively enumerable and self-bounded smear monoid. Then there is a measured set $\{\phi_{s(i)} \mid i \in N\}$ such that for all i, if ϕ_i is total then $\phi_{s(i)}$ is total, and $R_{\phi_i}^{(\Phi,S)} = R_{\phi_{s(i)}}^{\Phi}$.

We finally come to our hierarchy result, Theorem 3.7. Although others have noted that the Compression and Honesty Theorems harbour a method for constructing hierarchies (e.g., [5], p. 242), Bass and Young provide the full details of such a hierarchy extended into the constructive transfinite ([3]), using Kleene's familiar system O of notations for the constructive ordinals (see, e.g., Rogers [24]). The following result generalizes their main result to the case of S-invariant complexity theories:

THEOREM 3.7 *Let Φ be a measure of complexity and S a recursively enumerable and self-bounded smear monoid. Then there exist recursive functions t_d $(d \in O)$ such that $(R_{t_d}^{(\Phi,S)})_{d\in O}$ forms an increasing ordinal progression of complexity classes (i.e., if $d <_O b$ then $R_{t_d}^{(\Phi,S)} \subset R_{t_b}^{(\Phi,S)}$). Furthermore, the t_d can be defined so that $U_{d\in O} R_{t_d}^{(\Phi,S)}$ does not exhaust all recursive functions.*

SKETCH OF PROOF. Given Φ, let G be a measured set associated with Φ in accordance with the FACT above (so that G gives us an 'honest' way of naming all the complexity classess of theory (Φ, S)). Let c be a uniform recursive compression function in the sense that, for some initial total function ϕ_k, $R_{\phi_k}^{(\Phi,S)} \subset R_{\phi_{c(k)}}^{(\Phi,S)} \subset R_{\phi_{c(c(k))}}^{(\Phi,S)} \subset \ldots$, and where we either have (i) $\phi_{c(i)} = \psi(\phi_i)$ or (ii) $\phi_{c(i)} = h(\phi_i)$ for some fixed recursive function h. (Thus in both cases, c is truly a compression mechanism on functions.[13] The procedure for obtaining c and G is an easy generalization of the procedure described by Bass and Young.) For a sufficiently large total recursive function $\phi_k \varepsilon G$ (in case (ii), at least as large as the compression function r of Theorem 3.3), we define the ordinal hierarchy based on ϕ_k as follows:

(I) $t_1 = \phi_k$

(II) Notations for successor ordinals:
If $t_\alpha = \phi_i$, set $t_{2^\alpha} = \phi_{c(i)}$.

[13] The method of construction is easy if we don't insist on this. Thus if $G = (\phi_{s(i)})$, we can set $\phi_{c(i)} = \underline{H}(\phi_{s(i)})$, for any function or total effective operator $\underline{H}$ at least as big as the compression function r of Theorem 3.3. (Note that $\phi_i = \phi_j$ doesn't guarantee $\phi_{s(i)} = \phi_{s(j)}$.)

(III) Notations for limit ordinals:
If $\alpha = 3.5^{\beta} \in O$, choose t_α such that $R_{t_\alpha}^{(\Phi,S)} = U_n R_{\phi_{\beta(n)}}^{(\Phi,S)}$, and $t_\alpha \in G$.

To do this, use the Union Theorem (Theorem 3.4) and the Honesty Theorem, the latter in order to keep t_α within G so that (II) can be reapplied.

Note that in case (ii), $U_{d \in O} R_{t_d}^{(\Phi,S)} \neq$ set of all recursive functions. Reason: According to Blum's speed-up theorem, for every recursive r there are arbitrarily difficult recursive functions f with r-speed-up (i.e. such that if i is an index for f then there is an another index j for f such that $\Phi_i(x) \geq r(x, \Phi_j(x))$.[14] It is now not difficult to show that if r is a monotonically increasing function growing at least as fast as h, then for any recursive function f with r-speed-up, $f \in U_{\alpha \in O} R_{t_\alpha}^{(\Phi,S)}$ iff $f \in R_{t_1}^{(\Phi,S)}$. It follows there are arbitrarily difficult functions not in our hierarchy. The result again uses ideas in [3]. ■

COMMENT: If $\mathbf{Na} \subseteq [\Phi]_{\approx S}$, this result has the following bearing on the topic of intrinsic difficulty. It establishes the existence of hierarchies of recursive functions $(H_d)_{d \in O}$ that are not only ordered according to difficulty with respect to measure Φ but are also ordered according to their *intrinsic* difficulty: functions in $H_b - H_d$ (for $d <_O b$) are never as intrinsically easy as functions in H_d.

4. Natural measures and the problem of how to select an appropriately invariant theory of complexity

In this section, I briefly consider the question of the selection of an appropriate Φ and S, for the actual grading of recursive functions depends on making such a selection. Before doing this, however, let me briefly survey what is involved. Our fundamental interest is in the following relations between p.r. functions.

DEFINITION 4.1 Let **Na** be the set of natural measures, as before. Then

(1) *f is intrinsically as easy as g* ($f \leq_{\mathbf{Na}} g$) iff (a) domain $g \subseteq$ domain f and (b) given any measure Φ in **Na**, for all i such that $\phi_i = g$ there is a j satisfying $\phi_j = f$ such that $\Phi_i(x) \geq \Phi_j(x)$ a.e. in domain g.

(2) *g is intrinsically more difficult than f* ($f <_{\mathbf{Na}} g$) iff $f \leq_{\mathbf{Na}} g$ and $g \not\leq_{\mathbf{Na}} f$.

[14] Presentations of the Speed-up Theorem can be found in Blum [4], Calude [5], Hartmanis and Hopcroft [14] and Seiferas [25], among other places.

(3) *f has the same intrinstic difficulty as g* ($f \equiv_{\mathbf{Na}} g$) iff $f \leq_{\mathbf{Na}} g$ and $g \leq_{\mathbf{Na}} f$.

This is just account (iii) of section 1 above.

The connection between the S-invariant approach and the theory of intrinsic difficulty is then given by:

[C]: If $\mathbf{Na} \subseteq [\Phi]_S$ and $f \not\leq_{\Phi,S} g$, then $f \not\leq_{\mathbf{Na}} g$.

(Note that it is not universally true that if $\mathbf{Na} \subseteq [\Phi]_S$ and $f <_{\Phi,S} g$, then $f <_{\mathbf{Na}} g$, since $f \leq_{\Phi,S} g$ is consistent with f being substantially harder than g (although within bounds dictated by S). In specific cases, however, we can often say more: thus in the case of Theorem 3.1, the construction ensures that if $\mathbf{Na} \subseteq [\Phi]_S$ we not only have $\phi_i <_{\Phi,S} \phi_{h(i)}$ but also $\phi_i <_{\mathbf{Na}} \phi_{h(i)}$.)

Claim [C] provides the connecting link between the theory of intrinsic computational difficulty and the S-invariant approach to complexity theory which offers only S-dependent complexity orderings. As before, let T be the natural step-counting measure associated with some familiar version of one-tape Turing machines using binary code. By choosing an S that is large enough to make all natural measures S-equivalent to this paradigmatically natural measure T, we can then say specific things about the intrinsic difference in computational compexity between various recursive functions. But note that if we make S so large that it contains *all* 2-place recursive functions increasing in their second variable, we lose the ability to grade functions, since all complexity measures are recursively related ([4], [14], [25]). In that case, we never have $f \not\leq_{\Phi,S} g$, and can never apply Claim [C]. If, on the other hand, we choose the smallest S possible, $S_{\min} = \{e = \lambda xy[y]\}$, then our theories will have only a very limited degree of invariance; variations in the encoding technique or in the number of working tapes available can then be sufficient to invalidate results.

We list two rather more useful choices of S. The first choice is:

DEFINITION 4.2 $S* = \{f \mid f(x,y) = p(l(x),y)$ for all x, y, where p is a polynomial, increasing with respect to its second variable and $l(x)$ is the 'length' of x in binary $\}$

Simulation proofs show that the equivalence class $[T]_{\approx_{S*}}$ and hence also $[T]_{S*}$ includes an impressive array of different step-counting measures for Turing machines, for example step-counting measures based on a Turing machine model that includes a fixed number of multi-dimensional tapes as well as facilities like fast rewinds and head-to-head jumps. It also includes a number of step-counting 'flowchart' measures associated with certain familiar random access models, for example the model that uses addition and

subtraction as its only arithmetic instructions. (For a survey of simulation results of this type, see [29].)

The following simple result says that the class P of functions whose complexity can be bounded by some polynomial in the (binary) 'length' of their inputs (a class first introduced in Cobham [8]) comprises the easiest possible functions in the complexity theory associated with measure Φ and smear monoid $S*$, for any Φ $S*$-equivalent to T. (Strictly speaking, the recursive functions of this and subsequent results are assumed to be unary functions, and in talking of sets of recursive functions like P we are usually talking of their restrictions to unary recursive functions.)

THEOREM 4.3 *Let Φ be in $[T]_{S*}$. Then for every recursive f and g, if $f \in P$ then $f \leq_{\Phi,S*} g$. More particularly, we have: $P = R_0^{(T,S*)}$.*

Consider also the following smear monoid:

DEFINITION 4.4 $S_{\mathcal{E}} = \{f \mid f$ is a 2-variable function in $\mathcal{E}$, increasing with respect to $y\}$ (where the class $\mathcal{E}$ of elementary functions is the smallest class of functions containing the zero function, the successor function, the projection functions, exponentiation, and closed under composition and bounded recursion).

In the associated theory $(T, S_{\mathcal{E}})$ we have:

THEOREM 4.5 *Let Φ be in $[T]_{S_{\mathcal{E}}}$. Then for every recursive f and g, $f \in \mathcal{E}$ implies that $f \leq_{\Phi,S_{\mathcal{E}}} g$. In particular, $\mathcal{E} = R_0^{(T,S_{\mathcal{E}})}$.*

Because of the much faster growth rate of many of the functions in $S_{\mathcal{E}}$, the S-dependent complexity assessments we make in this theory are much cruder than assessments possible in the theory $(T, S*)$. In particular, most familiar functions turn out to be $S_{\mathcal{E}}$-equivalent in difficulty. On the other hand, there is one consideration that possibly favours working with a monoid like $S_{\mathcal{E}}$, or at least one containing some fast-growing functions such as exponentiation. There are a number of seemingly reasonable step-counting measures that do not belong to $[T]_{S*}$ but do belong to $[T]_{S_{\mathcal{E}}}$. Thus consider the step-counting measure associated with Turing machines using unary code. Better still (since the former are notoriously unwieldy), consider the flowchart measure whose underlying random access machine model counts multiplication as well as division as fundamental (see, for example, Hartmanis and Simon [15]), which is equivalent in power to Pratt and Stockmeyer's vector machine model ([21]); or consider machines that can do parallel computing (for details, see

van Emde Boas [29]). By using $S_\mathcal{E}$, we make sure that the corresponding S-dependent complexity assessments hold for these measures as well, and thereby increase the likelihood that our S-dependent complexity assessments hold for natural measures in general; we thereby also increase the likelihood (using claim [C]) that judgements of the form $f \not\leq_{T,S_\mathcal{E}} g$ always correspond to genuinely *intrinsic* complexity orderings $f \not\leq_{\mathbf{Na}} g$.

All this is consistent, of course, with saying that in general there may be a difference in intrinsic complexity between f and g even though there is no difference between them in terms of smeared complexity relations like $\cong_{T,S_\mathcal{E}}$ or $\cong_{T,S*}$ (that is, we may have $f \cong_{T,S_\mathcal{E}} g$ or $f \cong_{T,S*} g$ even though $f \not\equiv_{\mathbf{Na}} g$). In particular, claim [C] is consistent with saying that there are fine-grained intrinsic complexity differences among the elementary functions, as well as among the functions computable in polynomial time. Theories like $(T, S*)$ or $(T, S_\mathcal{E})$ don't deny this; in conjunction with [C], their main contribution is to show how extensional ways of delimiting the class of natural measures can then be used to yield insights into intrinsic complexity orderings. From that point of view, my approach is not in competition with intensional approaches to the definition of 'natural' or 'reasonable' measures, but merely supplements it.

5. Naturalness as a fuzzy concept

We should not, in any case, expect naturalness to be a black-and-white notion. I expect that some measures are natural to a degree only. Here are some reasons for this belief. (a) A measure may measure resources used on a computational model whose manner of moving information from one memory location to another, or of retrieving information, or of representing output, is exceedingly cumbersome — unnatural, as we might say; being cumbersome, however, admits of degrees. (b) We may wish to class some flowchart measures as unnatural on the grounds that they employ arithmetical instructions $i := Fj_1, ..., j_n$ where 'F' is interpreted in terms of unnaturally complex functions; being complex, however, admits of degrees.

The obvious way of allowing for fuzziness on our approach is to work with a set of smear monoids, each member of which corresponds to one delineation parameter; one way, that is, of resolving the vagueness in 'natural measure'. (Here I am working with something like David Lewis's account of vagueness in [18]). Thus:

DEFINITION 5.1 W is a *sm-system* (smear monoid system) iff

(i) W is a non-empty set of smear monoids; and

(ii) W is totally ordered under set inclusion.

Sm-systems W will be used to resolve the vagueness in 'natural measure'. In addition, they can be used to define a sense in which some functions can be said to differ much more in complexity than other functions.

DEFINITION 5.2 If Φ is a complexity measure, W an sm-system, and f, g, h, k are p.r. functions, let

(1) $d_{\Phi,W}(f,g) = W - \{S \in W \mid f \cong_{\Phi,S} g\}$.

(Thus $d_{\Phi,W}$ is a "measure" in the sense of classical measure theory. $d_{\Phi,W}(f,g)$ — the *distance* between f and g relative to Φ and W — is the collection of points S in W at which f and g differ in S-complexity.) We now say:

(2) $(f,g) \leq_{\Phi,W} (h,k)$ iff $d_{\Phi,W}(f,g) \subseteq d_{\Phi,W}(h,k)$

(3) $(f,g) <_{\Phi,W} (h,k)$ iff $d_{\Phi,W}(f,g) \subset d_{\Phi,W}(h,k)$

Informally, $(f,g) <_{\Phi,W} (h,k)$ when the extent to which f and g differ in complexity is smaller than the extent to which h and k differ. Using W to resolve the vagueness in 'natural measure' and hence 'intrinsic', we can also say that $(f,g) <_{\Phi,W} (h,k)$ when the judgment: 'k is intrinsically different in complexity from h' holds true for more resolutions of vagueness of 'natural', and hence 'intrinsic', than 'g is intrinsically different in complexity from f'. (This assumes that Φ is a natural measure to begin with.)

EXAMPLE 5.3 Let T be the step-counting measure for single-tape TMs using binary notation. Let F^P be the flowchart step-counting measure based on programs whose assignment and test instructions (e.g. '$i := F(j)$') feature functions from a finite set P of recursive functions. On one resolution of vagueness, let us suppose, F^{P1} is natural but F^{P2} is not, where $P1$ contains only constant functions while $P2$ also contains some more complex functions. On another, both F^{P1} and F^{P2} are natural. Corresponding to such a possibility there can generally be found a resolution system W containing smear monoids $S1$ and $S2$ such that F^{P1} and T are equivalent under $S2$ (i.e., $F^{P1} \equiv_{S2} T$) but not under $S1$ (i.e., $F^P 1 \not\equiv_{S1} T$). So if we use S_1, F^{P1} looks natural but not F^{P2}, while if we use S_2, both F^{P1} and F^{P2} look natural.

If W is an sm-system and Φ a complexity measure, we call (Φ, W) a *variable* complexity theory, to contrast with the simple one-monoid complexity theories looked at earlier. It can be shown that many natural hierarchies of classes of recursive functions have features that connect them with particular variable complexity theories. Consider first the idea of honesty, defined for the case of simple complexity theories (Φ, S):

DEFINITION 5.4 Call a set of recursive functions K *honest relative to* (Φ, S) if whenever $f \in K$ and $g \leq_{\Phi,S} f$ we also have $g \in K$. Let us further say that a hierarchy H of sets of recursive functions is *honest relative to* (Φ, S) if each member of H is.

Honesty in this new sense is simply closure under the relevant S-dependent relation of complexity. Honest classes have the property that a function's non-membership is a direct (or honest) indication that it is not as easy, relative to (Φ, S), as functions that are members. Given a plausible theory of intrinsic difficulty, it thus follows that hierarchy results involving honest hierarchies have intrinsic computational significance. This is one area where the structural and the computational notions of intrinsic difficulty show a degree of convergence, for it turns out that numerous familiar hierarchies are honest relative to various plausible theories (Φ, S).

But such results can often be improved using a stronger kind of closure or honesty. First we say:

DEFINITION 5.5 If S is a smear monoid, then the *complexity-restriction of* S *relative to* Φ, $O(\Phi, S)$, is the set of all recursive functions ϕ_i such that $\Phi_i(x) \leq \sigma(0, x)$ a.e. for some σ in S.

DEFINITION 5.6 We say that a set K of recursive functions is *hyperhonest relative to* (Φ, S) if (a) $K \subseteq O(\Phi, S)$, and (b) K is honest relative to (Φ, S). If W is an sm-system, we further say that a hierarchy H of classes of recursive functions is *hyperhonest relative to* (Φ, W) if for each K in H there exists S in W such that K is hyperhonest relative to (Φ, S).

It follows that if f belongs to class K in a hyperhonest hierarchy, then functions higher up in the hierarchy, or not in the hierarchy at all, are never as S-easy as f, where S reflects the complexity of f to the extent that f belongs to the complexity-restriction of S. Compare this to honest hierarchies, where the only conclusion to be drawn is that functions higher up in the hierarchy, or not in the hierarchy at all, are never as easy as f (relative to some fixed theory (Φ, S)).

The literature contains many examples of hyperhonest hierarchies. Given a class A of recursive functions, let $h(A)$ = the smear monoid, if one exists, of all binary functions in A increasing with respect to their second variable. Some examples:

(a) The Grzegorczyk hierarchy $(\mathcal{E}^n)_{n\geq 3}$ is hyperhonest with respect to $(T, (h(\mathcal{E}^3))_{n\geq 3})$, where n may here be taken as ranging over ordinals up to ε_0 and even beyond. (For the original Grzegorczyk hierarchy, see [12]. [9] contains a brief but accessible account of both this and the Péter hierarchy discussed below. Note that beyond $n = 3$ the same hierarchy can be constructed in numerous different ways; for details, see Calude [5], chapter 1. For the extensions of the original Grzegorczyk hierarchy, see Robbin [23] and Löb and Wainer [19].)

(b) R. Péter's hierarchy of multiply recursive functions $(R_k)_{\omega>k\geq 1}$ is hyperhonest with respect to $(T, (h(R_k))_{\omega>k\geq 1})$.

There also exist well-known hierarchies of functions that are not hyperhonest with respect to any variable complexity theories: Ritchie's hierarchy of predictably computable functions is of this kind ([22]).[15] Another hierarchy that is not hyperhonest as it stands is the hierarchy of Theorem 3.7, although a hyperhonest version can also be constructed. Note that the hyperhonest hierarchies (a) and (b) are ipso facto honest with respect to the most restrictive simple complexity theory listed: e.g., the Grzegorczyk hierarchy $(\mathcal{E}^n)_{\varepsilon_0>n\geq 3}$ is honest with respect to $(T, h(\mathcal{E}^3))$.

That (a) and (b) represent hyperhonest hierarchies follows from the fact that each level of the hierarchy satisfies conditions (1) and (2) of the theorem below (I suppress mention of the underlying measure Φ when there is no risk of confusion):

THEOREM 5.7 *Let K be a set of recursive functions satisfying the following conditions:* (1) $K = U_{g\in K} R_g^{\Phi}$ *and* (2) (a) *K is closed under substitution, and* (b) *for all $f \in K$ there is a monotonic $g \in K$ such that $g(x) \geq f(x)$ a.e. Then there is a smear monoid S such that K is hyperhonest with respect to S.*

PROOF. Suppose that K satisfies the conditions of the theorem. Let $s(K)$ = the set of all smear monoids S such that

(i) for all $q \in S$ and $r \in K$ there exists $m \in K$ such that $q(x, r(x)) \leq m(x)$ a.e.; and

[15] It is certainly true that if C is a class in the Ritchie hierarchy, then $f \in C$ and $g \notin C$ implies that $g \not\leq_{T,S_{\min}} f$, where $S_{\min}$ has as sole member the identity function. Hence the Ritchie hierarchy is honest with respect to at least the minimal simple complexity theory $(T, S_{\min})$. But what we do not have is: $f \in C$, $g \notin C$ implies that $g \not\leq_{T,S} f$ for some S to whose complexity-restriction f belongs.

(ii) for all monotonic l in K the function $\underline{l} \in S$, where $\underline{l}(x, y) = l(y)$.

We show (I) that $s(K)$ is non-empty and (II) that if S is any member of $s(K)$ then K is hyperhonest with respect to S.

(I) That $s(K)$ is non-empty follows from the fact that $Q \in s(K)$, where $Q = \{q \mid q$ is a binary function increasing in its second variable such that, if $r \in K$ then $\lambda x[q(x, r(x))] \in K\}$. The proof that $Q \in s(K)$ is straightforward. First, the identity function $e(x, y) = y$ is obviously a member of Q. Secondly, Q is closed under the operation $*$. Hence Q is a smear monoid, one that satisfies (i) by construction. Thirdly, if $l \in K$ and is monotonic, then $\underline{l} \in Q$. (Reason: given any $m \in K$, $\underline{l}(x, m(x)) = l(m(x))$ for every x, and hence, since K is closed under substitution, $\lambda x[\underline{l}(x, m(x))] \in K$; hence $\underline{l} \in Q$.)

(II) Let S be an arbitrary member of $s(K)$. Note first that: (*) $K \subseteq O(S)$. (Reason: Suppose $f \in K$. By (1), there is i for f and r in K such that $\Phi(x) \leq r(x)$ a.e. Hence by condition (2)(b) there is monotonic l such that $\Phi_i(x) \leq l(x)$ a.e. By (ii) in the definition of $s(K)$, it follows that for some $\underline{l}$ in S $\Phi_i(x) \leq \underline{l}(0, x)$ a.e. Hence $f \in O(K)$.) Note also that (**) *K is honest relative to S.* (Reason: Suppose that $f \in K$ and $g \leq_S f$. By condition (1) of the theorem, $g \in K$ if there is some complexity function Φ_i for g bounded a.e. by some function in K. But (again using condition (1)) if $f \in K$ and $g \leq_S f$, then there must be a complexity function Φ_i for g such that for some r in K and some σ in S, $\Phi_i(x) \leq \sigma(x, r(x))$ a.e. By condition (i) on $s(K)$, it then follows that Φ_i is bounded a.e. by some function in K. Hence $g \in K$.) The hyperhonesty of K with respect to S, where S is any member of $s(K)$, is an immediate consequence of (*) and (**). ∎

COROLLARY 5.8 (1) *The (extended) Grzegorczyk hierarchy $(\mathcal{E}^n)_{n\geq 3}$ is hyperhonest with respect to $(T, (h(\mathcal{E}^n))_{n\geq 3})$ (e.g., for $n < \varepsilon_0$).*

(2) *R. Péter's hierarchy of multiply recursive functions $(R_k)_{\omega > k \geq 1}$ is hyperhonest with respect to $(T, (h(R_k))_{\omega > k \geq 1})$.*

PROOF. In all these cases, an arbitrary member K of the hierarchy under consideration can be shown to satisfy conditions (1) and (2) of Theorem 5.7 (relative to measure T). It is, in addition, readily seen that $h(K) \in s(K)$, and hence, by the proof of Theorem 5.7, K is hyperhonest relative to $(T, h(K))$. ∎

A theory like $(T, (h(\mathcal{E}^n))_{n\geq 3})$ doesn't, of course, give the most fine-grained hyperhonest partitioning of the functions in $(\mathcal{E}^n)_{n\geq 3}$, just as $(T, (h(R_k))_{\omega > k \geq 1})$ doesn't give the most fine-grained hyperhonest partitioning of the multiply recursive functions $(R_k)_{\omega > k \geq 1}$. There are interesting and

important hierarchies that do much better. Thus Robbin showed that in his extension of the Grzegorczyk hierarchy to level ω^ω ([23]), $U_{n<\omega^k}\mathcal{E}^n$ = the class R_k of k-recursive functions. It can also be shown that Grzegorczyk's original hierarchy of primitive recursive functions admits of a hyperhonest refinement of a very complex order-type. In [20] (Theorem 5), Meyer and Ritchie show that:

FACT: For any $n \geq 3$ (and $< \omega$) there exists a recursively enumerable sequence $t_o^n, t_1^n, \ldots$ of elementary-honest functions such that:

(a) the classes $\mathcal{E}(t_i^n)$ are distinct and totally ordered under set inclusion.

(b) $\mathcal{E}^n \subset \mathcal{E}(t_i^n) \subset \mathcal{E}^{n+1}$ for all i.

(c) If $\mathcal{E}(t_i^n) \subset \mathcal{E}(t_j^n)$, then $\mathcal{E}(t_j^n)$ contains a universal function for $\mathcal{E}(t_i^n)$

(d) If $\mathcal{E}(t_i^n) \subset \mathcal{E}(t_k^n)$, then there is a j such that $\mathcal{E}(t_i^n) \subset \mathcal{E}(t_j^n) \subset \mathcal{E}(t_k^n)$

It follows that, for each $n \geq 3$, there are functions $t_o^n, t_1^n, \ldots$ such that $(\mathcal{E}(t_i^n))_{i<\omega}$ forms a dense chain under set inclusion. Hence $(\mathcal{E}(t_i^n))_{i<\omega}^{n\geq 3}$ is a hierarchy containing infinitely many dense chains. This hierarchy, which exhausts the primitive recursive functions, obviously refines $(\mathcal{E}^n)_{3\leq n<\omega}$. Further, it is hyperhonest: each $\mathcal{E}(t_i^n)$ meets all the conditions of Theorem 5.7. (Thus consider condition 1: the right-to-left inclusion is obvious from the elementary arithmetization of Turing machines, while the left-to-right inclusion follows routinely from the fact that the t_i^n are elementary-honest functions, i.e. for each such function t there is an elementary function g such that some complexity function for t is bounded a.e. by $g(x, t(x))$. Recall that the class $\mathcal{E}$ of elementary functions is identical to class $\mathcal{E}^3$.)

Finally, we note two interesting general properties of hyperhonest hierarchies. First, we might expect that (a) if two functions t and h are high up in a hierarchy hyperhonest with respect to (Φ, W) with t occurring higher than h, then the distance between t and h is greater, complexity-wise, than that between any two functions f and g which are lower down in the hierarchy. That is, we might then expect that $(f, g) <_{\Phi,W} (t, h)$.

In addition, however, we might expect that (b) if a hierarchy is a hyperhonest one then not only are functions high up in the hierarchy never as easy as functions lower down but, further, such rankings are more measure-invariant, hold for a larger class of possibly realistic measures, than comparable results about functions that are lower down in the hierarchy. In short, if we treat the associated sm-system as containing resolutions of vagueness of 'intrinsic' and 'natural', we might well expect that the claim that functions high up are never intrinsically as easy as functions lower down hold for more

ways of resolving the vagueness in 'intrinsic' than results about functions lower down.

The following result, the last one of the paper, demonstrates both these features of hyperhonest hierarchies (part (1) confirms (a) and part (2) confirms (b)).

THEOREM 5.9 *Let H be a hierarchy of classes of recursive functions each of which is non-empty and doesn't exhaust the recursive functions. Suppose that H is hyperhonest with respect to (Φ, W), where each member of W satisfies the following condition:*

(C) *For each σ in S, there is a σ' in S such that $\sigma(0,x) \leq \sigma'(x,0)$ a.e.*

Let K, L belong to H, where $K \subset L$, and suppose that $f \in K$, $g \in L - K$, $h \in L$ and $t \notin L$. Then:

(1) $(f,g) <_{\Phi,W} (t,h)$

(2) *there exists $\mathbb{R}, \mathbb{V} \in W$, where $\mathbb{R} \subset \mathbb{V}$, such that for all smear monoids U*

(i) *if $U \subseteq \mathbb{R}$, then for any $\Psi \in [\Phi]_{\mathbb{R}}$, $g \not\leq_{\Psi,U} f$;*

(ii) *if $U \subseteq \mathbb{V}$, then for any $\Psi \in [\Phi]_{\mathbb{V}}$, $t \not\leq_{\Psi,U} h$;*

(iii) *there exist measures $\Psi \in [\Phi]_V$ such that $g \leq_{\Psi,U} f$.*

PROOF. Suppose that H is hyperhonest with respect to W and that members of W satisfy (C). Assume that K, L belong to H, where $K \subset L$, and suppose that $f \in K$, $g \in L - K$, $h \in L$ and $t \notin L$.

To prove (1), note first that there exists $S \in W$ such that $f, g, h \in O(S)$ and $t \not\leq_S h$ (this is because $f, g, h \in L$ and H is hyperhonest with respect to W). Further, any S such that $f, g \in O(S)$ has the property that $f \cong_S g$. (Reason: Since $f \in O(S)$ there is a $\sigma \in S$ such that $\Phi_j(x) \leq \sigma(o,x)$ a.e. for some index j for f. Hence, by (C), there is a $\sigma' \in S$ such that for all indices i of g, $\Phi_j(x) \leq \sigma'(x, \Phi_i(x))$ a.e.; that is, $f \leq_S g$. Similarly $g \leq_S f$.) Since both $t \not\leq_S h$ and $f \cong_S g$ for some $S \in W$, it follows immediately that (1) $(f,g) <_W (t,h)$.

To prove (2), note first that for any $M \in H$, there exists $S \in W$ such that $M = O(S)$. (Reason: Since H is hyperhonest with respect to W, it follows that for any $M \in H$ there exists $S \in W$ such that (*) $M \subseteq O(S)$. Now suppose that the converse doesn't hold, i.e. that $\phi \in O(S)$ and $\phi \notin M$ for some ϕ. Let γ be an arbitrary member of M. Then, since H is hyperhonest, $\gamma \in O(S)$ and $\phi \not\leq_S \gamma$. But if $\phi \in O(S)$ and $\gamma \in O(S)$, then $\phi \leq_S \gamma$ (as in the proof of (1) above), a contradiction. Hence (**) $O(S) \subseteq M$. Combining (*) and (**), we obtain: $M = O(S)$.)

By the preceding argument, there exist $\mathbb{R}, \mathbb{V} \in W$ such that $K = O(\mathbb{R})$ and $L = O(\mathbb{V})$, where $f \in K$ and $g \in L - K$. But then it follows that $g \not\leq_{\Psi,\mathbb{R}} f$ for all measures $\Psi \in [\Phi]_{\mathbb{R}}$. (Reason: Suppose $g \leq_{\Psi,\mathbb{R}} f$. Then also $g \leq_{\Phi,\mathbb{R}} f$. But since $f \in O(\mathbb{R})(= K)$, it follows that $g \in O(\mathbb{R})$, and hence that $g \in K$, a contradiction.) It follows immediately that (i) if $U \subseteq \mathbb{R}$, then $g \not\leq_{\Psi,U} f$ for all measures $\Psi \in [\Phi]_{\mathbb{R}}$. In addition, $t \not\leq_{\Phi,\mathbb{V}} h$ (since H is hyperhonest). It again follows immediately that (ii) if $U \subseteq \mathbb{V}$, then $t \not\leq_{\Psi,U} h$ for all measures $\Psi \in [\Phi]_{\mathbb{V}}$.

Finally, (iii) there exist measures $\Psi \in [\Phi]_{\mathbb{V}}$ such that for any U (hence including any member of W) $g \leq_{\Psi,U} f$. For note that since $g \in L = O(\mathbb{V})$,
(#) there exists $r \in \mathbb{V}$ such that for some index j of g $\Phi_j(x) \leq r(x,o)$ a.e.
Let $j*$ be such an index of g. We can now define a new measure Ψ such that

$$\Psi_i(x) = \begin{cases} \Phi_i(x) & \text{if } i \neq j* \\ o & \text{if } i = j* \end{cases}$$

Clearly $g \leq_{\Psi,U} f$ for all smear monoids U. In addition, Ψ is $\mathbb{V}$-similar to Φ (by (#)), and hence $\Psi \in [\Phi]_{\mathbb{V}}$. This proves (iii). ■

Comment: Admittedly, Ψ is a contrived measure; its assessment of g doesn't impact on the assessment of any other function, even those easily definable from g, and this surely offends the ideal of a natural measure. (It especially offends this ideal if we take seriously the view that naturalness of measures has much to do with the existence of corresponding models of computation, a view alluded to in section 1.) But the contrived nature of Ψ is really an artifice of the fact that Theorem 5.9 concerns arbitrary measures Φ rather than recognisably natural measures. Once we focus on natural measures like T, the predicted increase in measure-invariance doesn't simply bring in contrived measures such as Ψ. Thus we can show, for example, that in the case of the Grzegorczyk hierarchy $(\mathcal{E}^n)_{n<\omega}$, $[T]_{h(\mathcal{E}^{n+3})} - [T]_{h(\mathcal{E}^{n+2})}$ includes the step-counting measure, call it T_{n+2}, on which J. P. Cleave bases the construction in [7] of his refinement of the hierarchy of primitive recursive functions at ordinal levels k for which $\omega.n < k < \omega.(n+1)$. T_{n+2} is essentially a flowchart step-counting measure that assigns unit complexity to the computation of the assignment instruction '$i := Fj_1, ..., j_m$', whenever 'F' is interpreted by any m-ary function in $\mathcal{E}^{n+2}$. In an earlier paper [17] Burkhard and I placed this construction in a more general setting, emphasizing the extent to which such functions are in some sense strongly constructive and hence 'natural' because of the way their complexity can be 'predicted' by functions already seen to be constructive. Both the flowchart nature of this measure and the nature of the functions interpreting 'F' suggest that such measures can lay claim to at least a degree of naturalness.

6. Conclusion

As we have seen, the most familiar subrecursive hierarchies tend to be hyperhonest; in addition, they satisfy condition (C). Hence the last theorem provides detailed confirmation of the idea that functions occurring at different levels high up in a subrecursive hierarchy differ more in complexity than do functions lower down. In addition, it shows that such results are in some sense more measure-invariant or 'intrinsic' than results concerning functions lower down.

Nonetheless, I stress that the use of 'intrinsic' in this context remains problematic. Whether the new measures constructed count as 'natural' or 'reasonable' (and hence relevant to the determination of 'intrinsic' differences in complexity) remains moot. I certainly do not claim to have adequately addressed this issue. Indeed, the approach defended in this paper can't really settle the question of naturalness of particular measures. Its main task has been to provide a framework that helps to clarify the problems and prospects facing a study of intrinsic complexity, and then to establish some general results from within this framework. The problem of finding a generally convincing *intensional* characterization of the idea of a natural measure and the co-relative idea of intrinsic difficulty remains a largely open one.[16]

References

[1] ARBIB, M. and M. BLUM, 1965, Machine dependence of degrees of difficulty, *Proceedings of the American Mathematical Society* **16**, 442-447.

[2] BAKER, T. P., 1978, "Natural" properties of flowchart step-counting measures, *Journal of Computer and System Sciences* **16**, 1–22.

[3] BASS, L. and P. YOUNG, 1973, Ordinal hierarchies and naming complexity classes, *Journal of the Association for Computing Machinery* **20**, 668–686.

[4] BLUM, M., 1967, A machine-independent theory of the complexity of recursive functions, *Journal of the Association for Computing Machinery* **14**, 322–336.

[5] CALUDE, C., 1988, *Theories of Computational Complexity*, North-Holland, Amsterdam, New York, Oxford, Tokyo.

[6] CICHON, E. A. and S. S. WAINER, 1983, The slow-growing and the Grzegorczyk hierarchies, *The Journal of Symbolic Logic* **48**, 399–408.

[7] CLEAVE, J. P., 1963, A hierarchy of primitive recursive functions, *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* **9**, 331–345.

[16] My thanks to Walter Burkhard for his help with earlier versions of this paper.

[8] COBHAM, A., 1965, The intrinsic computational difficulty of functions, *Proceedings of the 1964 Congress of Logic, Methodology and Philosophy of Science* North-Holland Publishing Company, 24–30.

[9] EPSTEIN, R. and W. A. CARNIELLI, 1989, *Computability: Computable Functions, Logic, and the Foundations of Mathematics* Wadsworth.

[10] FEFERMAN, S., 1962, Classification of recursive functions by means of hierarchies, *Transactions of the American Mathematical Society* **104**, 101–122.

[11] GLYMOUR, C., 1992, *Thinking Things Through: an Introduction to Philosophical Issues and Achievements,* MIT Press.

[12] GRZEGORCZYK, A., 1953, Some classes of recursive functions, *Rozprawy Matematyczne* **4**, 1–45.

[13] HARTMANIS, J., 1973, On the problem of finding natural computational complexity measures, *Cornell Computer Science Technical Report,* 73–179.

[14] HARTMANIS, J. and J. E. HOPCROFT, 1971, An overview of the theory of computational complexity, *Journal of the Association for Computing Machinery* **18**, 444–475.

[15] HARTMANIS, J. and J. SIMON, 1976, On the structure of feasible computations, in M. Rubinoff and M. C. Yovits, eds., *Advances in Computers* **14**, Academic Press, New York, 1–43.

[16] JOHNSON, D. S., 1990, A catalog of complexity classes, in van Leeuwen, J. (ed.), *Handbook of Theoretical Computer Science,* vol. **A**, *Algorithms and Complexity,* Elsevier, Amsterdam, New York, Oxford, Tokyo, 69–161.

[17] KROON, F. W. and W. A. BURKHARD, 1990, On a complexity-based way of constructivizing the recursive functions, *Studia Logica* **49**, 133–149.

[18] LEWIS, D., 1972, General semantics, in D. Davidson and G. Harman (eds.), *Semantics of Natural Language,* Reidel, Dordrecht, 169–218.

[19] LÖB, M. H. and S. S. WAINER, 1970, Hierarchies of number-theoretic functions I, II, *Archiv für Mathematische Logik und Grundlagenforschung* **13**, 39–51 and 97–113.

[20] MEYER, A. R. and D. M. RITCHIE, 1972, A classification of recursive functions, *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* **18**, 71–82.

[21] PRATT, V. R. and L. J. STOCKMEYER, 1976, A characterization of the power of vector machines, *Journal of Computing System Sciences* **12**, 198–221.

[22] RITCHIE, R. W., 1963, Classes of predictably computable functions, *Transactions of the American Mathematical Society* **106**, 139–173.

[23] ROBBIN, J. W., 1965, *Subrecursive Hierarchies,* Ph.D. dissertation, Princeton University.

[24] ROGERS, H., JR., 1967, *Theory of Recursive Functions and Effective Computability,* McGraw-Hill, New York.

[25] SEIFERAS, J., 1990, Machine-independent complexity theory, in van Leeuwen. *Handbook of Theoretical Computer Science,* vol. **A,** 165-186.

[26] SHAPIRO, S., 1981, Understanding Church's Thesis, *Journal of Philosophical Logic* **10,** 353–366.

[27] STOCKMEYER, L. and A. K. CHANDRA, 1979, Intrinsically difficult problems, *Scientific American,* 124–134.

[28] STOCKMEYER, L., 1987, Classifying the computational complexity of problems, *The Journal of Symbolic Logic* **52,** 1–43.

[29] VAN EMDE BOAS, P., 1990, Machine models and simulations, in van Leeuwen, J. (ed.), *Handbook of Theoretical Computer Science,* vol. **A,** 1–66.

[30] VERBEEK, R., 1978, *Primitiv-Rekursiv Grzegorczyk-Hierarchien* Universität Bonn, Informatik Berichte, Bonn.

DEPARTMENT OF PHILOSOPHY
UNIVERSITY OF AUCKLAND
PRIVATE BAG 92019
AUCKLAND, NEW ZEALAND
f.kroon@auckland.ac.nz