R. Gandy, M. Hyland (Eds.), LOGIC COLLOQUIUM 76
© North-Holland Publishing Company (1977)

# Computable and recursively countable functions of higher type

By R.O. Gandy and J.M.E. Hyland

We present some results concerning Kleene's generalisation of ordinary recursion theory to objects of higher type. Kleene has given a number of equivalent formulations of his ideas (see Kleene 1959a, 1962a 1962b, 1962c); we shall refer only to the original formulation based on the schemes S1-9. The motive underlying our work is the desire to establish connections between Kleene's generalisation and the ideas of constructive mathematics. So we investigate the continuity properties of 'recursive' functionals and the way in which they are related to the 'recursively countable' functions introduced in Kleene 1959b, or, equivalently, to the 'recursively continuous' functions of Kreisel 1959. These two latter notions were specifically introduced to embody ideas of constructivity. To avoid confusion we shall always use 'computable' to describe the objects introduced by S.1-9, reserving 'recursive' for the construction notions. At type 2 both notions coincide with the concept of recursive functional familiar in ordinary recursion theory. We shall be concerned almost entirely with objects of type 3. We include expositions of the principal notions discussed; we hope that the paper will be intelligible to any interested reader who is familiar with ordinary recursion theory.

Constructive mathematics considers functions as rules and so as intentions, while the computable functions and the computations by which their values are defined are wholly extensional. This means that constructive functionals can make use of information (e.g. a modulus of continuity) which is provided by their constructively presented arguments but which is not available in Kleene's computations. So it is not surprising that our most important results consist of constructive operations which are not computable. But we do not think that the moral to be drawn from these results is that Kleene's notion of computability is without interest for constructive mathematics. Hyland's work on limit spaces (see Hyland 1975, 1977) and the theory in Scarpellini 1971 of L-spaces show how a thoroughly extensional treatment can be given to objects (continuous functionals) which had previously been handled intensionally. The result is a considerable gain in clarity, intuitive appeal and ease of manipulation. There may be extensional ways of supplementing Kleene's schemes which correspond to the use of additional information provided by intensions. Indeed Feferman has already suggested possible ways of doing this (see his paper in this volume). But, we regard Kleene's schemes as likely to remain significant because they provide a basic minimum which any theory of extensional computation must include.

## §1 Type structures and computable functions

1.1. We find it convenient somewhat to modify the definitions of Kleene 1959a. We give an informal sketch of Kleene's notion thus modified, and of some facts about it. The reader who is familiar with Kleene's paper should have no difficulty in filling in formal details of our definitions and proofs. We hope that the reader who is unfamiliar with it will nevertheless be able to grasp the ideas and the lines of argument.

Kleene works with the <u>full</u> structure $\{T_n : n \in \omega\}$ of <u>pure</u> types, where $T_o$ is $\omega$ (the set of natural numbers) and $T_{n+1}$ is the set of all maps from $T_n$ into $\omega$. But Kleene's theory makes perfectly good sense if one hereditarily restricts the maps considered to sets $A_n$. We shall be considering in particular the case where $A_n$ consists of continuous maps or of effective operations. Also, to avoid tiresome details of coding, we will not confine ourselves to pure types but shall allow, hereditarily, numerical valued functions of several arguments.

1.2. <u>Definition</u>. (i) 0 is a <u>type symbol</u> of level 0.

(ii) if $\sigma_1,\ldots,\sigma_s$ are type symbols then $(\sigma_1,\ldots,\sigma_s \to 0)$ is a type symbol; its level is one greater than the maximum of the levels of $\sigma_1,\ldots,\sigma_s$.

We write $\vec{\sigma}$, $\vec{\tau}$ for $\sigma_1,\ldots,\sigma_s$, $\tau_1,\ldots,\tau_t$ respectively. We denote the unique pure type of level n by $\bar{n}$ : $\bar{0} = 0$ and $\overline{n+1} = (\bar{n}\to 0)$.

1.3. <u>Definition</u>. A <u>type structure</u> is a collection of sets $\{A_\sigma : \sigma$ a type symbol$\}$ such that (i) $A_o$ is $\omega$, (ii) if $\sigma$ is $(\vec{\tau} \to 0)$ then $A_\sigma$ is a set of maps from $A_{\tau_1} \times A_{\tau_2} \times \ldots A_{\tau_t}$ into $A_o$.

We adopt the convention that $\vec{a}$, $\vec{b}$ stand respectively for $a^{\sigma_1},\ldots a^{\sigma_s}$, $b^{\tau_1},\ldots,b^{\tau_t}$ where $a^{\sigma_i} \in A_{\sigma_i}$ $(1 \le i \le s)$, $b^{\tau_j} \in A_{\tau_j}$ $(1 \le j \le t)$.

1.4. Kleene's notion of computation for $\{T_n : n \in \omega\}$ and our modification of it are precisely delimited by the following principles, all of which are familiar in the ordinary theory of recursive functionals of level $\le 2$.

1.41. A <u>computation</u> is completely specified by giving a method of computation (programme, algorithm) and a list $\vec{a}$ of arguments (the inputs). The result (value, output) of the computation may be defined or undefined; if it is defined it is a natural number.

1.42. Methods of computation are coded by natural numbers which are called <u>indices</u>. The index also codes the list $\vec{\sigma}$ of the types of the arguments. We write $\{e\}(\vec{a})$ both for the computation and for its value (if defined); we write $\{e\}(\vec{a})\downarrow$, $\{e\}(\vec{a}) = z$, $\{e\}(\vec{a})\uparrow$ to mean, respectively, that $\{e\}(\vec{a})$ is defined, is defined and has the value z, is undefined.

1.43 A function f of type $\vec{\sigma} \to 0$ is <u>computable</u> <u>from</u> <u>parameters</u> $\vec{b}$ iff there is an index e such that

$$f(\vec{a}) = \{e\}(\vec{a}, \vec{b}) \text{ for all } a^{\sigma_i} \in A_{\sigma_i} \quad (1 \le i \le s)$$

We require that the type structure be closed under computation: if $b^{\tau_j} \in A_{\tau_j}$ $(1 \le j \le t)$ then $f \in A_{(\vec{\sigma} \to 0)}$

1.44. The computations of ORT (ordinary recursion theory) are computations of the generalised theory. More precisely: if $\{p\}(\vec{x}, \vec{f})$ is a computation of ORT and the numbers $\vec{x}$ and the level 1 functions $\vec{f}$ all occur in the list $\vec{a}$, then there is an index e such that

$$\{e\}(\vec{a}) \simeq \{p\}(\vec{x}, \vec{f}).$$

1.45 Computation is closed under the substitution of computed values for numerical arguments. More precisely, if f and g are indices and if the objects in the lists $\vec{b}$, $\vec{c}$, $\vec{d}$ all occur in the list $\vec{a}$, then there is an index e such that

$$\{e\}(\vec{a}) \simeq \{f\}(\vec{b}, \{g\}(\vec{c}), \vec{d}).$$

1.46 A computation can use an oracle to determine the value of one of its arguments (of level $\ge 2$) for given computable arguments. More precisely, if $b^{\vec{\tau}\to 0}$ occurs in the list $\vec{a}$, and if the functions $f^{\tau_j}$ $(1 \le j \le t)$ are computable from the parameters $\vec{a}$ with indices $f_i$, then there is an index e such that

$$(S.8_{(\vec{\tau}\to 0)}) \qquad \{e\}(\vec{a}) = b^{\vec{\tau}\to 0}(f^{\tau_1}, \ldots, f^{\tau_t}).$$

The special case of this when $\vec{\tau} \to 0$ is the pure type of level n+2 is Kleene's S.8:

$$(S.8_{n+2}) \qquad \{e\}(\vec{a}) \simeq b^{\overline{n+2}}(\lambda c^{\bar{n}}.\{f\}(\vec{a}, c)).$$

If the level of $\vec{\tau} \to 0$ is 1, then the use of an oracle is already provided for by 1.44, 1.45; the corresponding Kleene scheme is S.7.

1.47 The result of a computation can be used as an index for a further computation. More precisely, for any list $\vec{\sigma}$ with, say $\sigma_i = 0$ there is an index e such that

$$\{e\}(\vec{a}) \simeq \{a^{\sigma_i}\}(\vec{a}).$$

This corresponds to Kleene's S.9; in connection with 1.45 it has the stated effect.

1.5. Numerous equivalent formulations, each with an attendant system of indexing, have been given; we mention two. Gandy 1967 describes the methods of computation in terms of finite register machines, working, so to speak, in infinitary time. This has the advantage that indexing plays no part in the definition, and that the significance of S.8 is made clearer. But the definition is unhandy for routine manipulations and proofs. Bergstra 1975 uses four schemes (as opposed to Kleene's nine) corresponding to our 1.44-1.47.

It is an unfortunate fact that in most treatments, particularly in Kleene 1959a the details of index manipulation often obscure the general lines of constructions and arguments. That is why in this paper we prefer not to specify any system of indexing. But it is necessary, especially for applications of 1.47, that the assignment of indices should be done in an orderly, computable fashion. We therefore add to 1.44-1.47 the stipulation that e can be readily computed from the other indices mentioned together with the lists of types of the mentioned strings of arguments, and conversely that those indices and type-lists can be readily computed from e. ('Readily computed' can be taken, for example, as 'computed by a Czillig-Kalmar elementary function').

1.6. Once details of indexing have been fixed, 1.44-1.47(or whatever other schemes are taken as primitive) determine inductive definitions of the sets $\{(e,\vec{a}) : \{e\}(\vec{a})\downarrow\}$ and $\{e,\vec{a},z) : \{e\}(\vec{a}) = z\}$, each scheme yielding a clause of the definitions. For 1.46 the clause is infinitary, since it must require that the computations $\{f_i\}(\vec{a})$ can be defined for all $a^{\sigma_i} \in A_{\sigma_i}$ $(1 \leq i \leq s)$. Corresponding to these definitions one uses a form of induction for proofs and constructions. We shall refer to this as 'induction on the computation'. (Kleene calls it 'induction on $\{z\}(-)$', but this is slightly misleading as it suppresses the arguments and these occur as parameters in the induction). Since 1.46 is the single scheme which distinguishes higher type computations from those of ORT we shall usually only give the inductive step corresponding to this case.

## §2. Sections and degrees

2.1. Let $A = \{A_\sigma : \sigma$ a type symbol$\}$ be a given type structure closed under computation; the letters a,b,c range over A. If $a \in A_\sigma$ and s is the level of $\sigma$ we say s is the level of a. We write '$a \leq b$', '$a \leq b$ Mod$(\vec{c})$' for 'a is computable from b', 'a is computable from b, $\vec{c}$ respectively. We define:

$$a \equiv b \text{ iff } a \leq b \text{ and } b \leq a.$$

This is an equivalence relation which partitions A into equivalence classes called <u>degrees</u> which are partially ordered by the quotient relation $\leq/\equiv$, which we also denote by $\leq$. We write '$\underset{\sim}{a}$' for the degree of a and for an arbitrary degree. (Similar definitions may be made for 'degree mod $(\vec{c})$'). Besides the usual problems associated with a degree structure there are problems about the interplay between the degree structure and the type structure.

2.2. Definitions The $\sigma$-degree of a ('$\sigma$-deg(a)') is $A_\sigma \cap \underset{\sim}{a}$. A degree $\underset{\sim}{a}$ is represented in type $\sigma$ iff $A_\sigma \cap \underset{\sim}{a} \neq \emptyset$.

2.3. Proposition If a degree is represented in a type $\sigma$ of level s it is represented in all types of level $\geq$ s.

This is an immediate consequence of the existence of invertible functions which code $a^\sigma$ in $A_\tau$ whenever level $\tau \geq s$ (see Kleene 1959a or Gandy 1967).

2.4. Definition The level of a degree is the least level at which it is represented. An object $a^\sigma$ is irreducible if its level and the level of its degree coincide; i.e. if $a^\sigma \equiv b^\tau$ implies level of $\tau \geq$ level of $\sigma$.

2.5. Definition The $\sigma$-section of an object a, or of a degree $\underset{\sim}{a}$ is defined by

$$\sigma\text{-sc}(a) = \sigma\text{-sc}(\underset{\sim}{a}) = \{b^\sigma : b^\sigma \leq a\}.$$

Because constant functions are computable the $\sigma$-section of a is never empty. We shall see in §5 that, for a particular $\sigma$, distinct degrees may have coincident $\sigma$-sections.

2.6. Definition An object a or a degree $\underset{\sim}{a}$ is t-obtainable if there is a $b^\tau$ of level t such that $a \leq b^\tau$.

By 2.3 if t $\geq$ level of $\underset{\sim}{a}$, then $\underset{\sim}{a}$ is t-obtainable, but the converse may be false. For example if A is the type-structure of continuous functionals then every type-2 object is 1-obtainable, since it can be computed from any of its associates. But (see Hinman 1973) and §5 below) there are irreducible continuous type 2 objects.

## §3. Countable functionals

3.1. Notations. We record here a number of more or less standard notations which will be used extensively in the following sections.

Since we shall only be concerned with objects of level $\leq 3$ we use different founts, rather than superscripts as follows:

Level 0 (numbers): $i,j,k,m,n,p,q,x,y,z$;

Level 1: $\alpha,\beta,\gamma,\delta$;

Level 2: F,G,H;

Level 3: $\Gamma,\Delta,\Phi,\Psi$.

Unless otherwise indicated a letter stands for an object of pure type of the appropriate level.

We suppose that any finite sequence $(u_o, u_1, \ldots, u_{r-1})$ of natural numbers is coded by a number $u = \langle u_o, u_1, \ldots, u_{r-1}\rangle$ in one of the standard ways. Seq is the set of such codes; Biseq the set of codes of binary sequences. The letters u,v,w will always stand for members of seq. $\ell h(u)$ is the length, $u_i$ or $(u)_i$ is the i-th member, of the sequence with code u. $u{*}n$ is the code for the sequence got by adding n at the end of the sequence coded by u. '$\bar{\alpha}(n)$' stands for '$\langle\alpha(0), \alpha(1), \ldots, \alpha(n-1)\rangle$'. $u \subseteq v$ means that the sequence (coded by) v is an extension of the sequence u, and $u \subset \alpha$ means that $\bar{\alpha}(\ell h(u)) = u$. $V_u = \{\alpha: u \subset \alpha\}$. We write '$\alpha \leq \beta$' to mean that $\forall x\ \alpha(x) \leq \beta(x)$.

The letters A,B will be used only for finite sets of pairs (u,p). Dom A = $\{u: \exists p(u,p) \in A\}$, and the support of A is $\cup\{V_u: u \in \text{Dom } A\}$.

If a and b are functions of the same type, then $a =_X b$ means that they agree on the subset X of their domain; in particular '$\alpha =_n \beta$' stands for '$\bar{\alpha}(n) = \bar{\beta}(n)$'.

We often think of the sequence numbers as being laid out on a tree with < > at the top. (Although we are not from the antipodes, nevertheless our trees grow downwards). Each $\alpha$ determines a path through this tree. If R(u) is such that $R(\bar{\alpha}(n))$ and $\forall m < n \neg R(\bar{\alpha}(m))$ then we say R secures $\alpha$ at $\bar{\alpha}(n)$. If R secures every $\alpha$, then the set of points at which R secures some $\alpha$ form a bar; the points on and above the bar are the non-past-secured points, and the relation $u \subseteq v$ is well founded on the set consisting of them. Recursion on this well-founded relation is called bar recursion (of type 0).

3.2. We denote by $C = \{C_\sigma: \sigma$ a type-symbol$\}$ the type structure of Kleene's 'countable' or Kreisel's 'continuous' functionals. For a fairly exhaustive account of C and of the different ways in which it may be defined see Hyland 1977. We discuss here just the cases $\sigma = \bar{0}, \bar{1}, \bar{2}, \bar{3}$. We set $C_o = T_o = \omega$, and $C_1 = T_1$. We topologise $C_1$ by taking $\{V_u: \text{Seq } u\}$ as a neighbourhood base.

The guiding principle of Kleene's definition is: a map $f : C_n \to \omega$ belongs to $C_{n+1}$ if its value at $b^n \in C_n$ is determined by a finite amount of information about $b^n$. For n = 1 the obvious interpretation of 'finite amount of information about $\beta$' is 'finitely many values of $\beta$'. Then $F \in C_2$ can be expressed in the following ways.

3.21 F is a continuous map from $C_1$ into $\omega$.

3.22 The predicate 'F is constant on $V_u$' secures every $\beta$: i.e. $\forall\beta\ \exists n$ F is constant on $V_{\bar{\beta}(n)}$.

3.23 There is a function $\alpha_F$, called an <u>associate</u> of F which satisfies

(i) $\forall\beta\ \exists n\ \alpha_F(\bar{\beta}(n)) > 0$

(ii) $(\forall\beta,n)[\alpha_F(\bar{\beta}(n)) > 0 \rightarrow \alpha_F(\bar{\beta}(n)) = F(\beta)+1]$.

3.24 The advantage of 3.23 is that it suggests immediately what should be meant by 'a finite amount of information about F'; namely 'finitely many values of $\alpha_F$'. Observe that (i) and (ii) do not specify $\alpha_F$ uniquely. Indeed, given $\alpha_F$ there are 'finer' associates $\alpha_F'$ such that if $\alpha_F(u) = 0$ then $\alpha_F'(u) = 0$ but not conversely. We could make $\alpha_F$ unique (the 'principal associate of F') by adding

(iii) F is constant on $V_u \rightarrow \alpha_F(u) > 0$.

But to do so would be disastrous. Firstly because if F is a recursive functional (in the sense of ORT) it will have a recursive associate, but may not have a recursive associate satisfying (iii). Secondly, F may be specified by <u>any</u> $\alpha_F$ satisfying (i) and (ii), but one cannot constructively determine whether $\alpha_F$ satisfies (iii) or not. So, 'finite amount of information about F' is to mean 'finitely may values of <u>any</u> associate $\alpha_F$ satisfying (i) and (ii)'.

3.3. With this agreed, we can give the definition of $C_3$ as follows:

A map $\Gamma$: $C_2 \rightarrow \omega$ belongs to $C_3$ iff it has an associate $\alpha_\Gamma$ which satisfies

(i) $F \in C_2\ \forall\beta[\beta$ is an associate for $F \rightarrow\ \exists n\ \alpha_\Gamma(\bar{\beta}(n)) > 0]$;

(ii) $F \in C_2 (\forall\beta,n)[\beta$ is an associate for $F \wedge \alpha_\Gamma(\bar{\beta}(n)) > 0$
$\rightarrow \alpha_\Gamma(\bar{\beta}(n)) = \Gamma(F) + 1]$.

3.4. We note that - in accord with 3.24 - the following functionals do NOT belong to $C_3$.

$$\Gamma_1(F) = \begin{array}{l} 0 \text{ if F is constant} \\ 1 \text{ otherwise} \end{array}$$

$$\Gamma_2(F) = \begin{array}{l} 0 \text{ if } \quad (\alpha) = 0 \\ 1 \text{ otherwise.} \end{array}$$

3.5. The requirements of 3.3 can be expressed as a continuity property in the following way. Let A be any finite set of ordered pairs (u,p) where u is a sequence number. We set

$$V_A = \{F \in C_2 : \text{ if } (u,p) \in A \text{ then } F(V_u) = \{p\}\}.$$

For any $\alpha$ the collection

$$F^\alpha = \{V_A : A \text{ is a finite subset of } \{(u,p) : \alpha(u) = p+1\}\}$$

of subsets of $C_2$ forms a filter base which generates a filter $[F^\alpha]$. We say that $F^\alpha$ and any filter including $F^\alpha$ converge to F just in case $\alpha$ is an associate for F. One may think of $F^\alpha$ as giving a method of approximating to F. Now $\Gamma \in C_3$ iff it is eventually constant on every convergent filter; or, taking the only filter on $\omega$ which converges to n to be $\{X: n \in X\}$, $\Gamma \in C_3$ iff whenever a filter converges to F its image under $\Gamma$ generates a filter which converges to $\Gamma(F)$. This is precisely the definition of continuity which is appropriate for 'limit spaces'.

3.6. If $\alpha_F'$ is a finer associate than $\alpha_F$ then $[F^{\alpha'}]$ is strictly included in $[F^\alpha]$, since some $V_A \in F^\alpha$ is too small to belong to $F^{\alpha'}$. Thus, by 3.24, there is no minimal filter converging to F, no most rapid method of approximation. Hence the limit space structure of $C_2$ is not associated with any topology; for the neighbourhood of a point generate a minimal filter converging to it. In particular note that the topology generated by the $V_A$'s is not appropriate to $C_2$; for example, it makes $\Gamma_1$ of 3.4 continuous. There is a topology (the 'induced' topology of Hyland 1977) whose continuous maps into $\omega$ constitute $C_3$; but it is thoroughly eccentric - for example, it does not have a countable base.

3.7. We have followed Kleene in presupposing the existence of higher type objects, and then narrowing down to the countable ones. In the alternative approach of Kreisel 1959 one first characterises the associates, and then introduces the higher type objects as equivalence classes of associates. Ershov (1972, 1974) and Hyland (1975, 1977) have shown that the associates may be given an elegant characterisation in terms of certain algebraic lattices.

3.8. Recursion theory. A countable functional is recursively countable iff it has a recursive associate (in the sense of ORT). One needs also a notion of relative recursion based on partial recursive operations. The following was first proposed by Gandy; Hyland 1975 gives equivalent definitions in terms of limit spaces and cogent arguments that it is the correct definition for countable functions. An object $a \in C$ is <u>recursively countable</u> in $b \in C$ if there is an associate-invariant recursion from b to a: that is if there is an index e (of ORT) such that $\lambda_x . \{e\}(x, \alpha_b)$ is an associate for a whenever $\alpha_b$ is an associate of b; we then write $a \leq_c b$. This gives rise to a partition of C into 'countable' degrees; a countable degree is a union of the degrees of §2. This is a consequence of the following theorem, which also shows that C is closed under computations (as required by 1.43).

3.91 <u>Theorem</u> (Kleene 1959b). There is a partial recursive function $\phi$ such that if $\{e\}(\vec{a})$ is a defined computation over C then $\phi(e)$ is defined, and if $\vec{\alpha}$ are associates for $\vec{a}$ then

(1) $$\{e\}(\vec{a}) = \{\phi(e)\}(\vec{\alpha}).$$

(Observe that, by 1.44, it makes little difference whether one regards the RHS as a computation by S1-9 or a computation of ORT). (1) shows that $\phi(e)$ is an index for an associate invariant recursion. Hence one readily proves:

3.92 <u>Corollary</u>. For any $a,b \in C$, if $a \leq b$ then $a \leq_c b$.

In proving 3.91 when the arguments $\vec{a}$ are of level $\leq 2$ one arrives at a stronger result which will be used in the next section. We say that $\alpha'_F$ is a <u>restricted associate</u> of F if in 3.23 (i) is changed to

(i)' $\forall \beta\, [\beta \in 1\text{-sc}(F) \rightarrow \exists n\, \alpha'_F(\bar{\beta}(n)) > 0]$

Then in 3.91 '$\vec{\alpha}$ are associates' can be replaced by '$\vec{\alpha}$ are restricted associates'. We deal only with the case of a single argument of type 2, and outline the proof of:

3.93. <u>Lemma</u> There is a partial recursive function $\chi$ such that if $\{e\}(\vec{p},F)\downarrow$, then $\chi(e,\langle\vec{p}\rangle)$ is defined, and if $\alpha'_F$ is any restricted associate for F then

(2) $$\{e\}(\vec{p},F) = \{\chi(e, \langle\vec{p}\rangle)\}\ (\alpha'_F).$$

The proof is by induction on the computation. When e is defined by 1.47 the inductive clause in the definition of $\chi$ is trivial; this is the reason for computing the index on the RHS of (2) from e <u>and</u> the numerical arguments of the LHS. Once 3.93 is established a function $\phi$ as in 3.91 is obtained by routine manipulations of indices. Now we consider only the case where e is defined by 1.46 (S.8). So

$$\{e\}(\vec{p},F) = F(\beta) \text{ where } \beta = \lambda q.\ \{f\}(\vec{p},F,q)$$

By induction hypothesis:

(3) $\beta(q) = \{\chi(f, <\vec{p},q>)\}\ (\alpha'_F)$ for all $q$.

Then $\chi\ (e,\vec{p})$ is defined as an index for the following process: compute $\beta(0)$, $\beta(1),\ldots,$ using the RHS of (3) until an $n$ is reached for which $\alpha'_F(\overline{\beta}(n)) > 0$. Since $\beta \in$ 1-sc(F) and $\alpha'_F$ satisfies (1)' such an $n$ will be reached. The required value is then $\alpha'_F(\overline{\beta}(n)) - 1$. This concludes our sketch of the proof of 3.93.

## 4. Non-computability of the fan-functional

The problem solved in this section was first posed in Kreisel 1959. It was solved by Tait (unpublished) around 1962; the solution presented here is a somewhat streamlined version of one which was presented by Gandy at the 1965 Leicester summer school and colloquium.

4.1. Any $\gamma \in C_1$ determines a compact subset $K_\gamma$ of $C_1$ defined by

$$K_\gamma = \{\beta\colon \beta \le \gamma\};$$

every compact subset of $C_1$ is included in some $K_\gamma$. The fan functional $\Phi$ is defined by

$$\Phi(F,\gamma) = (\mu n)(\forall\, \beta,\beta' \in K_\gamma)[\beta =_n \beta' \rightarrow F(\beta) = F(\beta')];$$

it gives a uniform modulus of continuity for $F$ on $K_\gamma$.

4.2. Proposition. $\Phi$ is recursively countable.

Proof. Let $\alpha_F$ be any associate for $F$. By König's lemma there is a finite subset, $X$ say, of $\{u\colon \alpha_F(u) > 0\}$ which secures every $\beta$ in $K_\gamma$; $X$ can be computed from $\alpha_F$, $\gamma$. And then, for any $u$ on the tree for $K_\gamma$ above the bar $X$, $F$ is constant on $V_u \cap K_\gamma$ iff $\alpha_F$ takes the same value on all those points of $X$ which lie below $u$. Hence $\Phi(F,\gamma)$ can be computed from $\alpha_F$, $\gamma$. QED.

4.3. Let us modify the limit space structure introduced in 3.5 by associating with each $F \in C_2$ the collection

$$\Lambda(F) = \{[F^\alpha]\colon \alpha \text{ is a restricted associate of } F\}$$

of filters. Then, by 3.93, if $\{e\}(F)\downarrow$, then the functional $\lambda G.\ \{e\}(G)$ is eventually defined and constant on every filter of $\Lambda(F)$. Thus

4.31. <u>Theorem</u> Any computable functional of type 3 over $C_2$ is continuous on the limit space structure defined by $\Lambda$.

On the other hand we shall prove

4.4. <u>Theorem</u> For any $\gamma$ which is not eventually zero, and any F there is a restricted associate $\alpha$ of F such that $\lambda G\Phi(G,\gamma)$ is not eventually constant on $F^{\alpha}$.

<u>Proof</u>. Let $\beta \in K_{\gamma} - 1\text{-sc}(F)$. Let $\alpha$ be a restricted associate of F such that $\alpha(\bar{\beta}(n)) = 0$ for all n. Let $V_A \in F^{\alpha}$. We shall construct G so that $G \in V_A$ and $\Phi(G,\gamma) \neq \Phi(F,\gamma)$. Choose n so that

(i) $n > \Phi(F,\gamma)$

(ii) $u \in \text{Dom } A \to \ell h(u) < n$.

The choice of $\alpha$ and condition (ii) on n mean that $V_{\bar{\beta}(n)}$ is disjoint from the support of A. Hence if G differs from F only on $V_{\bar{\beta}(n)}$ then $G \in V_A$. And if, for example, we set

$$\begin{aligned} G(\delta) &= F(\delta)+1 \text{ if } \bar{\beta}(n+1) \subset \delta \\ &= F(\delta) \quad \text{otherwise} \end{aligned}$$

then, by condition (i), $\Phi(G,\gamma) \neq \Phi(F,\gamma)$. QED.

4.5. <u>Corollary</u> $\Phi$ is not computable.

4.6. <u>Remarks</u> (1) The above proof shows what is sufficiently obvious, that $F^{\alpha}$ converges to F more rapidly than any filter of the structure considered in 3.5.

(2) A very short proof of 4.5 is given in 5.3 below.

(3) Theorems 4.31, 4.4 remains true if we consider computations in a parameter $G \in C_2$ and understand by 'restricted associate of F' one which secures every $\beta \in 1\text{-sc}(F,G)$. Thus $\Phi$ is irreducible.

(4) Another, particularly simple, example of a recursively continuous non-computable functional of level 3 is:

$$\begin{aligned} \Psi(F,G,\gamma) &= 0 \text{ if } \forall\beta \in K_{\gamma}.\ F(\beta) = G(\beta), \\ &= 1 \text{ otherwise} \end{aligned}$$

## §5. <u>A revealing counter-example</u>.

We give a simple construction due to Hyland for an irreducible type 2 continuous functional. The basis of the construction is the following familiar lemma.

5.1. There is a recursive predicate R which secures every recursive element, but not every element, of $2^\omega$. For example take:

$$R(u) \text{ iff } (\exists x < \ell h(u))[T(x,x,\ell h(a)) \wedge U(\ell h(u)) = (u)_x].$$

Let $W_e = \{x: \exists y\, T(e,x,y)\}$ be the e-th r.e. subset of $\omega$.

5.2. Theorem For each e there is an $F_e \in C_2$ such that:-

(i) $F_e$ is computable from $W_e$;

(ii) $W_e$ is computable from $F_e$ and $\Phi$;

(iii) there is a partial recursive functional $\phi$ such that

$$\phi(d) = F_e(\alpha) \text{ for all recursive } \alpha;$$

(iv) there is a partial recursive $\psi$ such that if $\{z\}(\vec{x}, F_e) = n$, then $\{\psi(z)\}(\vec{x}) = n$ for any list $\vec{x}$ of numerical arguments;

(v) $1\text{-sc}(F_e)$ consists of the recursive functions.

Proof. For any $X \in 2^\omega$, any $n \in \omega$ set

$$F_e(X,n) = \begin{array}{l} 0 \text{ if } \exists y[T(e,n,y) \wedge (\forall x \le y) \neg R(\bar{X}(x))], \\ 1 \text{ otherwise,} \end{array}$$

with R as in 5.1. $F_e$ can of course be coded by an object in $C_2$.

Then (i) and (iii) are obvious; (iv) follows from (iii) by an argument similar to our proof of 3.93, and (v) is an immediate consequence of (iv). And to prove (ii) observe that if $n \notin W_e$ then $\lambda X.F_e(X,n)$ (= $G_n$, say) is constantly 1, while if $n \in W_e$ then $G_n(X) = 0$ if X is not secured by R; using $\Phi$ we can compute the values taken by $G_n$ on $2^\omega$ and so decide whether $n \in W_e$ or not.

5.3. Corollaries 1. $C_2$ contains irreducible objects. For suppose $F_e \equiv \alpha$; by (v) $\alpha$ is recursive, and so, by 3.91, $F_e$ is recursively countable. But then, by 4.2 and (iii), $W_e$ is recursive, which is false for suitable e.

2. $\Phi$ is not computable. For if it were $W_e$ would be computable from $F_e$ by (ii) and so recursive by (v).

5.4. Remarks (1) Hinman (1973) first proved corollary 1 by a rather elaborate 'spoiling' construction. We see that this is not necessary. But Hinman's argument is not wasted since it can be used to construct an F which does not have the same countable degree as a function.

(2) Unlike the proof in the previous section one cannot immediately relativise the argument given here to prove that $\Phi$ is not computable from any $G \in C_2$.

(3) For each e, $F_e$ is an extension to $C_1$ of an effective operation; that is why (v) holds. Harrington has shown that there are objects $F \in C_2$ which are not extensions of effective operations, but which do satisfy (v).

(4) If one modifies the definition of $F_e$ to:

$$F_e(X,n) = y+1 \text{ if } T(e,n,y) \wedge \forall x < y\,[\neg T(e,x,y) \wedge R(\bar{X}(x))],$$
$$= 0 \quad \text{otherwise,}$$

and chooses an e with $W_e$ not recursive, then the functionals $G_n$ have the following properties: (a) one can compute a modulus of continuity of $G_n$ at X (<u>viz.</u> $G_n(X)$); (b) one cannot compute, uniformly in n, a bound for the value of $G_n$. This shows that one can hardly hope to find a constructive proof of the classical theorem that if a function is pointwise continuous on a compact set K then it is bounded on K.

5.5. Bergstra 1975 gives a significant extension of Hyland's construction. Let $R_a$ be defined by

$$R_a(u) \text{ iff } (\exists x \le \ell h(u))[T(a,x,\ell h(u)) \wedge (u)_x < \ell h(u)],$$

so that if $W_a$ is not recursive then R secures every recursive $\alpha \in C_1$ but not every $\alpha \in C_1$. We define $B_a^b : C_1 \to \{0,1\}$ by

$$B_a^b(\alpha) = \begin{array}{l} 0 \text{ if } \exists y[T(b,\alpha(0),y) \wedge (\forall x \le y)\ \neg R_a(\bar{\alpha}(x))], \\ 1 \text{ otherwise.} \end{array}$$

We state without proof the fundamental properties of $B_a^b$.

5.51 <u>Theorem</u> Let a,b be chosen so that $\underset{\sim}{0} < \underset{\sim}{W}_a \le \underset{\sim}{W}_b$; then

(i) $B_a^b$ is computable from $W_b$,

(ii) $W_b$ is computable from $W_a$ and $W_a^b$,

(iii) 1-sc($B_a^b$, $\Phi$) consists of the recursive functions.

5.52 <u>Remarks</u> (1) By (ii), use of $B_a^b$ allows one to jump from $W_a$ up to $W_b$; interesting results are obtained by constructing functionals which allow a sequence of such jumps; see Bergstra 1976 Bergstra & Wainer 1976 and Norman 1976.

(2) We <u>conjecture</u> that only the recursive functions are recursively countable in $B_a^b$ - this would give a significant strengthening of (iii) of the theorem. A disproof of the conjecture would show that countable recursion at type 3 is more powerful than computation using $\Phi$. We now turn to an entirely different proof of this fact.

## §6. The functional $\Gamma$

Gandy spent some time trying to prove the conjecture: every recursively countable object of type 3 can be computed from $\Phi$. He then discovered the object $\Gamma$ which made the conjecture implausible. Hyland proved that it is false.

6.1. Definitions (A) Let $n, u \in$ seq, $\alpha$, $F$ be given. We define $n*\alpha$, $u^\frown\alpha \in C_1$ and $F*n$, $F^\frown u \in C_2$ as follows.

$$\begin{aligned} (n*\alpha)(0) &= n, & (u^\frown\alpha)(i) &= u_i \text{ if } i < \ell h(u), \\ (n*\alpha)(i+1) &= \alpha(i), & &= \alpha(i-\ell h(u)) \text{ otherwise;} \\ (F*n)(\beta) &= F(n*\beta), \\ (F^\frown u)(\beta) &= F(u^\frown\beta) \quad \text{for all } \beta. \end{aligned}$$

This $F^\frown u$ mirrors on $C_1$ the behaviour of $F$ on $V_u$.

(B) $\Gamma \in C_3$ is defined by

$$(1)\ \Gamma(F) = (F*0)(\lambda n.\ \Gamma(F*(n+1))).$$

6.2. Theorem $\Gamma$ is uniquely specified by the above equation, and is recursively countable.

Proof. If $\Gamma$ satisfies (1) then

(2) $\Gamma(F^\frown u) = (F^\frown u*0)(\lambda n.\ \Gamma(F^\frown u*(n+1)))$ for all $u$. Given $\beta$ and an associate $\alpha_F$ for $F$ we can compute an $n$ such that $F^\frown\bar\beta(n)$ is constant and so

(3) $\Gamma(F^\frown u) = F(\beta)$ if $\bar\beta(n) \subseteq u$.

Since every path $\beta$ is secured at some point $\bar\beta(n)$ where (3) applies, equation (2) defines $\Gamma(F^\frown u)$ at all points of the well-founded tree of non-past-secured sequence numbers $u$. Further if we set $\gamma(u) = \Gamma(F^\frown u)$ then $\gamma$ satisfies

$$\gamma(u) = (F^\frown u*0)(\lambda n.\ \gamma(u*(n+1))),$$

so that $\gamma$ is recursive in any associate $\alpha_F$. Finally $\Gamma(F) = \gamma(\langle\ \rangle)$.
QED

6.3. Remarks (1) The computation of $\Gamma(F)$ from $\alpha_F$ is an instance of bar-recursion. The determination of $\Gamma(F)$ from $F$ differs from bar-recursion in that there is a single equation (2) which applies at all points $u$, rather than a division into cases according as to whether $u$ is secured or merely securable.

(2) Another way of seeing that $\Gamma$ is well-defined is to regard it as defined by recursion with respect to Brouwer's inductive definition of $C_2$:

(i) $\Gamma(\lambda\alpha.\ k) = k$

(ii) If $F(\alpha) = G_{\alpha(o)}(\lambda n.\ \alpha(n+1))$,

then $\Gamma(F) = G_o\ (\lambda n.\ \Gamma(G_{n+1}))$.

(3) The functionals $\Phi$ and $\Gamma$ can be combined into one functional, $\Delta$ say, defined by

$$\Delta(F) = \Phi(F{*}O,\ \lambda n.\ \Delta(F{*}(n+1))).$$

(4) For a given associate $\alpha_F$ of F there will be an n such that the computation of $\Gamma(F)$ does not require knowledge of $F(\alpha)$ for any $\alpha$ with $\alpha(0) > n$. But, by suitably choosing $\alpha_F$, we can make n arbitrarily large. Thus we cannot determine in an associate-invariant way a compact set K such that $\Gamma(F)$ depends only on the behavious of F on K. So it is implausible that we could compute $\Gamma$ from $\Phi$.

6.3. Before we prove that $\Gamma$ cannot be computed from $\Phi$ we introduce a device due to Bergstra by which computations from $\Phi$ can be replaced by computations using only arguments of type 2. For any F let $H_F$ be defined by:

$$\{<u,p>:\ \ell h(u) \le \Phi(F,\gamma) \wedge (\forall i < \ell h(u))(u_i \le \gamma(i)) \wedge F(V_u \cap K_\gamma) = \{p\}\}.$$

Thus $H_F(\gamma)$ codes, in effect, the behaviour of F on the compact set $K_\gamma = \{\beta:\ \forall i.\ \beta(i) \le \gamma(i)\}$.

6.32 Theorem (Bergstra)

(i) $H_F$ is computable from F, $\Phi$.

(ii) $F(\gamma)$ and $\Phi(F,\gamma)$ are computable from $H_F(\gamma)$.

(iii) There is a partial recursive $\phi$ such that if $\{e\}(F,\Phi) = y$, then $\{\phi(e)\}(H_F) = y$.

(i) and (ii) are obvious from the definitions. The proof of (iii) is by a fairly tedious induction on the computation and can be found in Bergstra 1975.

We also need to connect the filters associated with $H_F$ with those associated with F. Put

$$L(u) = \{v:\ \ell h(v) = \ell h(u) \wedge (\forall i < \ell h(u))(v_i \le u_i))\},$$

$$L_u = \bigcup \{V_v:\ v \in L(u)\}.$$

For any finite set A of ordered pairs (u,p) let

$$A^F = \{(v,\ F(b \frown \underline{O})):\ (\exists u \in \text{Dom } A)(v \in L(u))\}$$

where $\underline{O} = \lambda n.O$.

6.33 Lemma

(i) $H_F$ is constant on $V_u$ iff $(\forall v \in L(u))$($F$ is constant on $V_v$).

(ii) If $G = {}_{L_u}F$ and $H_F(V_u) = \{p\}$, then $H_G(V_u) = \{p\}$.

(iii) $F \in V_{A^F}$ iff $H_F \in V_A$.

(iv) If $G \in V_{A^F}$ then $H_G \in V_A$.

All these facts are immediate consequences of the definitions involved. We may note in passing that although for simplicity we used $F$ to define $A^F$, in fact it can be defined (so as to satisfy (iii) and (iv)) recursively from $A$. Now we are ready to prove the main result of this section.

6.4. Theorem (Hyland) $\Gamma$ cannot be computed from $\Phi$.

Given any defined computation $\{e\}(F, \Phi)$ we shall construct $G$ so that

$$\{e\}(G, \Phi) = \{e\}(F, \Phi) \text{ but } \Gamma(G) \neq \Gamma(F).$$

We consider the computation $\{\phi(e)\}(H_F)$ of 6.32(iii) and choose a restricted associate $\alpha$ for $H_F$ which is fine enough to ensure that if $W_A \in \delta^\alpha$ then $W_{A^F}$ contains a $G$ satisfying $\Gamma(G) \neq \Gamma(F)$.

Choose $\delta$ to satisfy:

(i) if $\delta' \geq \delta$ then $\delta \notin$ 1-sc$(H_F)$;

(ii) $\forall x\ \delta(x) > 0$.

Let $\gamma^F_m = \lambda n.\ \Gamma(F^\frown\bar{\delta}(m)*(n+1))$, and let

$\beta^F_m = (\bar{\delta}(m)*0)^\frown\gamma^F_m$; then

(1) $\Gamma(F\ \ \bar{\delta}(m)) = F(\beta^F_m)$ for each $m$.

Let $v_m = \bar{\beta}^F_m(m + \delta(m))$, and let $\alpha$ be a restricted associate for $H_F$ which satisfies:

(a) $\alpha(\bar{\delta}'(m)) = 0$ for all $m$, all $\delta' \geq \delta$;

(b) $\alpha(u) = 0$ if $\exists m, u'.\ u \subseteq u' \wedge v_m \in L(u')$.

By 4.31 there is an $A$ such that $V_A \in \delta^\alpha$ and $\lambda H.\ \{\phi(e)\}(H)$ is constant on $V_A$; then, by 6.33,

(2) if $G \in V_{A^F}$, then $\{e\}(G, \Phi) = \{e\}(F, \Phi)$.

By (a) we can find $M$ such that if $\delta' \geq \delta$ then $V_{\bar{\delta}'(M)}$ does not intersect the support of $A$; therefore

(3) $V_{\bar{\delta}'(M)} \cap$ support $(A^F) = \emptyset$.

For $m < M$ define

$$P_m = \max\{(u)_{m+1+\delta(m)}: u \in \text{Dom } A\}$$

and $W_m = v_m * (p_m+1)$.

Now by (b) if $v \in \text{Dom}(A^F)$ then $v \not\subseteq v_m$. Hence

(4) $V_{W_m} \cap \text{Support } (A^F) = \emptyset$.

Finally let G agree with F except as required by:

$$G(V_{\bar{\delta}(M)}) = \{P_{M-1} + 1\},$$
$$G(V_{W_m}) = \{P_{m-1} + 1\} \text{ for } 0 < m < M,$$
$$G(V_{W_0}) = \{\Gamma(F) + 1\}.$$

Then, by (3) and (4), G satisfies the premise of (2).

Also $\Gamma(G^\frown\bar{\delta}(M)) = P_{M-1} + 1$.

Hence $\gamma^G_{M-1}(n) = \gamma^F_{M-1}(n)$ if $n < \delta(M-1) - 1$

$= P_{M-1}+1$ if $n = \delta(M-1) - 1$.

Therefore $\beta^G_{M-1} \in V_{W_{M-1}}$ and, by (1),

$$\Gamma(G^\frown\bar{\delta}(M-1)) = P_{m-2} + 1.$$

Repeating this argument for m a M-2,...,0 we find that $\beta^G_0 \in V_{W_0}$ and hence that $\Gamma(G) = \Gamma(F)+1$.

This conclude the proof.

Remark. Hyland's original proof proceeded directly by an induction on the computation of $\{\phi(e)\}(H_F)$. In the context of this paper it seemed more appropriate to appeal to Theorem 4.31.

## §7 Effective Operations

In a thoroughgoing constructivist theory of mathematics all objects considered will be, in some way, recursively presented. In this section we investigate the lower levels of the type structure $R = \{R_\sigma : \sigma$ a type symbol$\}$ of recursive objects. We set $R_0 = \omega$, $R_1 = \{\alpha : \alpha$ is recursive$\}$. For higher types we require an object to be presented by a recursive associate which defines a value whenever the arguments are recursively presented. Following the pattern of the argument in 3.1-3.3 we define '$\alpha \in R_n$' and '$\alpha$ is a quasi-associate for a' (abbreviated to '$\alpha \in QA(a)$') by induction on n as follows.

7.1 Definition

(i) If $a \in R_1$, then $\alpha \in QA(a)$ iff $\alpha = a$.

(ii) If $a: R_{n+1} \to \omega$ then $\alpha \in QA(a)$ iff:

(a) $(\forall b \in R_{n+1})(\forall \beta \in QA(b))$ [$\beta$ is recursive $\to \exists m.\alpha(\bar\beta(m)) \neq 0$];

(b) $(\forall b \in R_{n+1})(\forall \beta \in QA(b))(\forall m)$ $[\alpha(\bar\beta(m)) \neq 0 \to \alpha(\bar\beta(m)) = a(b)+1]$.

(iii) $a \in R_{n+2}$ iff $\exists \alpha.[\alpha \in QA(a) \wedge \alpha$ is recursive].

7.2 Remarks (1) $R_2$ consists of just those partial recursive functionals from $R_1$ into $\omega$ which are defined on $R_1$ and which are continuous on $R_1$ considered as a subspace of $C_1$.
(2) If $F \in C_2$ is recursively countable, then $F \restriction R_1 \in R_2$.
(3) But the converse of (2) is false. For example, if P secures every recursive but not every sequence and $F = \lambda\alpha.(\mu m)P(\bar\alpha(m))$ then $F \in R_2$, but F cannot be extended to an element of $C_2$. Thus

$$R_2 \nsubseteq \{F \quad R_1 : F \in C_2\},$$

and R is not a sub-structure of C.
(4) The significance of the clause '$\beta$ is recursive' will be clarified below (see 8.3(2)).
(5) R can also be defined as a category of limit spaces. We discuss the definition for $R_2$(cf.3.5). Let

$$V_A = \{F: F \text{ is a continuous map from } R_1 \text{ into } \omega \wedge \forall u,p.[(u,p)\in A \to F(V_u) = \{p\}]\}.$$

Let <A> be a numerical code for A. We say that the filter $\mathcal{F}_X$ generated by $\{V_A: \langle A\rangle \in X\}$ is recursive just in case X is r.e. It is easy to see that if $\mathcal{F}_X$ converges to G then G has a recursive associate iff $\mathcal{F}_X$ is recursive. Thus $R_2$ consists of those points of the limit space $R_1 \to \omega$ which can be approximated to by recursive filters; and for the filters on $R_2$ we take precisely these. This construction is readily extended to all types.

7.3 Theorem (Kleene) There is a partial recursive function $\phi$ such that if $\vec{a} \in R$, $\{e\}(\vec{a}) = z$ and $\vec\alpha$ are quasi-associates (not necessarily recursive ones) for $\vec{a}$, then

$$\{\phi(e)\}(\vec\alpha) = z.$$

The proof of the theorem is similar to the proof of 3.91 and we omit it, and the proofs of the following.

Corollary 1 There is a partial recursive function $\psi$ such that if

$\vec{f}$ are indices for recursive quasi-associates for $\vec{a}$, and if $\{e\}(\vec{a}) = z$, then

$$\{\psi(e)\}(\vec{f}) = z.$$

Corollary 2 The 1-section of any $a \in R$ consists exactly of the recursive functions.

Corollary 3 If $F \in R_2$, $\{e\}(F) = z$ and $\alpha$ is any quasi-associate for $F$, then $\lambda G.\{e\}(G)$ is eventually constant on the filter

$$\mathfrak{f}^{\alpha} = \{V_A\text{: } A \text{ is a finite subset of } \{(u,p)\text{: } \alpha(u) = p+1\}\}.$$

7.4 Remarks (1) Corollary 1 shows that R is closed under computation.
(2) Corollary 2 shows that for objects in R 'quasi-associate' and 'restricted associate' are interchangeable.
(3) It was natural for us, in defining R, to presuppose the existence of higher type objects and to think in terms of extensions. Corollary 2 suggests that one can also think entirely in terms of intentions (indices) and recursive functions. We now indicate how R can be approached in this way.

7.5 We first define a type structure $Q = \{Q_n\text{: } n \in \omega\}$ and a relation '$e \in I_n(a)$' (to be read as 'e is an index for the object $a \in Q_n$') as follows:

(i) $Q_o = \omega$; $I_o(k) = \{k\}$.

(ii) $I_{n+1}(a) = \{e\text{: } a \text{ maps } Q_n \text{ into } \omega \wedge (\forall b \in Q_n)(\forall f \in I_n(b))[\{e\}(f) = \{a(b)\}]\}$,

where $\{e\}$ is the partial function $\omega \to \omega$ with index e of ORT.

(iii) $Q_{n+1} = \{a\text{: } I_{n+1}(a) \neq \emptyset\}$.

Thus $Q_1 = R_1$ and $I_1(\alpha) = \{e\text{:}\alpha = \{e\}\}$.

One can also define the indices for members of $Q_n$ without reference to higher type objects as follows:

7.6 Definition (i) $E_o = \omega$; (ii) $j =_o k \longleftrightarrow j = k$.

(ii) $E_{n+1} = \{e\text{: } E_n \subseteq \mathrm{Dom}(\{e\}) \wedge (\forall f,f' \in E_n)[f =_n f' \to \{e\}(f) = \{e\}(F')]$.

(iii) $e =_{n+1} e' \longleftrightarrow (\forall f \in E_n)[\{e\}(f) = \{e'\}(f)]$.

From the definitions one easily proves:

$$E_n = \bigcup\{I_n(a)\text{: } a \in Q_n\}.$$

One can, of course, define the members $Q_n$ in terms of the equivalence classes of the relation $=_n$. (See, for example, Kreisel 1959).

We shall call the members of Q (rather than their indices) effective operations. Observe that if $e \in E_{n+1}$ then $\{e\}$ is total on $E_n$. A weaker definition, due to Myhill and Shepherdson 1955, is concerned with hereditarily partial functions.

7.7 Theorem (Kreisel-Lacombe-Shoenfield)

For each n, $Q_n = R_n$.

The difficult part of the proof (which we omit) is to show that $Q_n \subseteq R_n$. Details (at least for the crucial case n = 2) may be found in Kreisel, Lacombe and Shoenfield 1959, Gandy 1962 (a very succint proof), or Rogers 1967. The proof is effective in the sense that it provides a primitive recursive function $\phi$ such that if $e \in I_n(a)$ then $\phi(e)$ is an index for a recursive quasi-associate for a. For $n > 2$ one needs an effectively indexed dense basis for $R_{n-1}$ and the decidability of various facts about the filter bases in $R_{n-1}$. The necessary proofs are sketched in Kreisel 1959 and given in detail in Tait 1963 and in Hyland's thesis. Hyland also gives the definition of $R_\sigma$ and $Q_\sigma$ for arbitrary type symbols $\sigma$ and proves $R_\sigma = Q_\sigma$. He observes that a direct proof of $R_\sigma = Q_\sigma$ seems to require that $\sigma$ has the form $\tau \to 0$.

The proof that $R_{n+1} \subseteq Q_{n+1}$ follows readily from $Q_n \subseteq R_n$ by using corollary 2 of 7.4. Finally it should be observed that the proof of 7.4 is not constructive - it requires, unavoidably an application of Markov's principle. For an exact description of the situation see Beeson 1975.

## §8 An effective operation which is not computable

In this section we construct a $\Delta \in R_3$ which is not computable.

Let S be a recursive predicate of sequence numbers which does not secure every sequence, but which does secure every sequence which is dominated by some recursive sequence. E.g.

$$S(u) \longleftrightarrow (\exists x<\ell h(u))[T(x,x,\ell h(u)) \wedge (u)_x < \ell h(u)].$$

(1) If $\alpha \leq \beta \in R_1$, then $(\exists m) S(\bar{\alpha}(m))$.

Consider the continuous map of Baire space into Cantor space (binary sequences) given by:

$$\alpha^B = \underbrace{0\ldots01}_{\alpha(0)}\underbrace{0\ldots01}_{\alpha(1)}\ldots\ldots\underbrace{10\ldots01}_{\alpha(n)}\ldots.$$

This also defines a map of seq onto BiSeq:

$$u^B = \underbrace{0\ldots01}_{u_0}\underbrace{0\ldots01}_{u_1}\ldots\ldots\underbrace{10\ldots0}_{u_r},$$

(where $u = \langle u_o, u_1, \ldots, u_r \rangle$). For $u \in \mathrm{Seq}$ we define

$$\begin{aligned} \beta_u(m) &= (u^B)_m, && \text{for } m < \ell h(u^B), \\ &= 1 && \text{for } m = \ell h(u^B), \\ &= 0 && \text{otherwise.} \end{aligned}$$

Thus $\beta_u$ is an eventually zero binary sequence which codes $u$.

Let $T(= \{u: \forall v \subseteq u.\ \neg S(v)\})$ be the tree of all non-past-secured sequences. Now we define

$$\Delta(F) = \mathrm{Max}\{F(\beta_u): u \in T\}.$$

8.1 <u>Lemma</u> $\Delta \in R_3$.

Let $\alpha$ be any recursive quasi-associate for some $F \in R_2$. We must show that $\Delta(F)$ is defined and can be computed from $\alpha$. Set

$$\phi(u) = (\mu m)(\alpha(\bar{\beta}_u(m)) \neq 0) \dot{-} (1 + \ell h(u^B));$$

$\phi(u)$ is the number of 0's (if any) needed at the end of $\beta_u$ to ensure that it is secured by $\alpha$. Thus

(A) $\phi$ is computable,

(B) if $p \geq \phi(u)$ and $v \supseteq u{*}p$, then

$$F(\beta_v) = F(\beta_{u*\phi(u)}).$$

Now let $\psi$ be recursively defined by:

$$\psi(r) = \mathrm{Max}\{\phi(u): \ell h(u) = r \wedge (\forall i < r)(u_i \leq \psi(i))\}.$$

So $\psi$ is computable. Let $X$ be the tree of sequence numbers bounded by $\psi$; i.e.

$$X = \{u: (\forall i < \ell h(u))(u_i \leq \psi(i))\}.$$

Now by (1) every path through $X$ is secured by $S$. Hence by König's lemma $X \cap T$ is a finite set which can be computed from $\alpha$.

Finally we show

(*) if $v \in T$, then for some $u \in X \cap T$, $F(\beta_v) = F(\beta_u)$.

Let $v \in T-X$. Then there is $u \subset v$, with $\ell h(u) = r$ say, such that $u \in X$ and $u{*}p \subseteq v$ for some $p > \psi(r)$. But $\psi(r) \geq \phi(u)$, and so by (B),

$$F(\beta_v) = F(\beta_{u*\phi(u)}).$$

Now $v$ is not past-secured by $S$, so $u$ is not secured by $S$ and thus $u{*}\phi(u) \in T$. And evidently $u{*}\phi(u) \in X$. This proves (*).

So

$$\Delta(F) = \mathrm{Max}\{F(\beta_u): u \in X \cap T\}$$

and thus is defined and computable from $\alpha$. QED.

8.2 Lemma $\Delta$ is not computable.

The argument is similar to those used in 4.4 and 6.4. Suppose $\{e\}(F)$ is defined, where $F \in R_2$. Let $\gamma$ be not secured by S and let $\alpha_F$ be a restricted associate for F which does not secure $\gamma^B$. By Corollary 3 to Theorem 7.3 there is an A such that:

(i) $G \in V_A \rightarrow \{e\}(G) = \{e\}(F)$;

(ii) $\gamma^B \notin$ Support $(V_A)$.

Hence there is a $v \subset \gamma$ such that

$$V_{v^B} \cap \text{Support } (V_A) = \emptyset.$$

Define $G \in R_2$ by:

$$\begin{aligned} G(\beta) &= \Delta(F)+1 \text{ if } \beta \in V_{v^B}, \\ &= F(\beta) \quad \text{otherwise.} \end{aligned}$$

Then, by (i), $\{e\}(G) = \{e\}(F)$. But $v \in T$, and $\beta_v \in V_{v^B}$. So $\Delta(G) > \Delta(F)$. Thus $\Delta \neq \lambda G.\{e\}(G)$. QED.

Remarks. (1) This counter-example is due to Gandy; it was first presented at the Logic Summer School, Leicester 1965.

(2) The example emphasises the significance of the clause '$\beta$ is recursive' in 7.1(iia). $\Delta$ is eventually constant on any recursive filter converging to some $F \in R_2$, while $\lambda G.\{e\}(G)$ is eventually constant on any filter (determined) by a restricted associate which need not be recursive) converging to some $F \in R_2$.

(3) Does the continuity condition of 7.3 Corollary 3 characterise the computable functionals at type 3? More precisely suppose $\Gamma: R_2 \rightarrow \omega$ has a recursive associate and satisfies the continuity condition: is $\Gamma$ computable?

## §9 Partial Objects

Many of the objects and constructions which occur in constructive mathematics are required to be total; for example, both Bishop and Brouwer treat a real number generator as a function whose domain is $\omega$. That is one of the reasons why we have so far only considered functions which are total over the chosen domains. Another reason is that if total objects are to be treated only as extensions, then functions with partial arguments will appear as intentions.

However, one's understanding of the continuous functionals is a certainly enriched by a knowledge of the partial ones; and, since recursion naturally produces partial functions it is not surprising that it is easier to produce a satisfactory recursion theory for partial than for total objects of higher type. This was first realised and exploited by Platek; it is discussed by Feferman in his paper in this volume. In this section we discuss the relation between the type structure C and the type structure $C^U$ of hereditarily partial continuous functions which was first explicitly introduced by Ershov (1972). [We say 'explicitly' because $C^U$ consists of the lower parts of the lattices defined by Scott (1970). Despite the great elegance of the lattice-theoretic approach, we think that the connections with C are made clearer if one rules out Scott's 'over defined' elements right from the start. The interested reader should consult Scott (1976) and Hyland (1975)]. This section and the next are intended not so much as an exposition but as a comment on the work of Ershov and Feferman. (We have had the advantage - not reciprocated - of reading Feferman's paper before writing this section).

First we describe Ershov's semi-lattice of partial continuous functions for types 1 and 2. We differ slightly from the account given in Feferman's paper in that we allow an undefined elements at type 0. We maintain the conventions of 3.1 and use dotted letters ($\dot{x}$, $\dot{\alpha}$, $\dot{F}$, $\dot{\Gamma}$, etc.) to range over $\dot{C}_0$, $\dot{C}_1$, $\dot{C}_2$, $\dot{C}_3$.

9.0. $\dot{C}_0 = \omega \cup \{\bot\}$; $\dot{x} \subseteq \dot{y} \longleftrightarrow \dot{x} = \bot \vee \dot{x} = \dot{y}$.

9.1 $\dot{C}_1$ consists of all the partial functions $\dot{\alpha}$: $C_0 \xrightarrow{\cdot} \omega$. It is partially ordered by the relation of extension:

$$\dot{\alpha} \subseteq \dot{\beta} \longleftrightarrow \forall\, \dot{x}, y.\ \dot{\alpha}(\dot{x}) = y \rightarrow \dot{\beta}(\dot{x}) = y.$$

A <u>formal</u> <u>neighbourhood</u> d is a finite (possibly empty) set of the form $\{(\dot{x}_i, y_i): i < k\}$. We say it is <u>consistent</u> (and write 'd $\epsilon$ $FCN_1$ ) if

$$y_i = y_j \rightarrow (\dot{x}_i = \bot \vee \dot{x}_j = \bot \vee \dot{x}_i = \dot{x}_j) \quad (i,j < k)$$

It is <u>irreduntant</u> if $\dot{x}_i \subseteq \dot{x}_j \rightarrow i = j$ $(i,j < k)$.

Each d $\epsilon$ $FCN_1$ determines a unique <u>finite</u> (or, in the terminology of lattice theory, <u>compact</u>) element $\dot{d}$ by:

$$\dot{d}(\dot{x}) = y_i \text{ if } \dot{x}_i \subseteq \dot{x} \text{ for some } i < k, \text{ is undefined otherwise.}$$

Conversely each finite element determines a unique irredundant neighbourhood.

Each d $\epsilon$ $FCN_1$ also determines a 'neighbourhood':

$$U_d = \{\dot{\alpha}: d \subseteq \dot{\alpha}\}.$$

The set of neighbourhoods $\{U_d: d \in FCN_1\}$ generates a $T_0$ topology on $\dot{C}_1$.

9.2. $\mathring{C}_2$ consists of all partial $\mathring{F}:\mathring{C}_1 \xrightarrow{\circ} \omega$ which are continuous with respect to this topology. $\mathring{C}_2$ is partially ordered by the relation of extension. A formal consistent neighbourhood A is a finite set $\{d_i,y_i): d_i \in FCN_i \quad , i < k\}$ such that

$$y_i = y_j \rightarrow (d_i \cup d_j) \in FCN_1.$$

It is irredundant iff each $\mathring{d}_i$ is irredundant and if, further, $\mathring{d}_i \subseteq \mathring{d}_j \rightarrow i = j \ (i,j < k)$.

If $A \in FCN_2$ then we set

$$\mathring{A}(\mathring{\alpha}) = y_i \text{ if } \mathring{d}_i \subseteq \alpha \text{ for some } i,$$
$$\text{is undefined otherwise,}$$

and $U_A = \{\mathring{F}:\mathring{A} \subseteq \mathring{F}\}$. As before each A (or $U_A$) determines a unique irredundant neighbourhood. The neighbourhoods generate a $T_o$ topology on $C_2$.

9.3. It will now be obvious how to extend these definitions to all pure types (and, indeed, to arbitrary types). Further the following facts are readily verified.

(1) An $\mathring{a} \in \mathring{C}_{n+1}$ is completely specified by its values on the finite elements of $\mathring{C}_n$.

(2) The predicates 'is a consistent formal neighbourhood' and 'is an irredundant consistent formal neighbourhood' and the relation $\subseteq$ between finite elements are all decidable.

9.4. We now turn to the connection between $\mathring{C}$ and C. We start by making some definitions and stating some simple propositions which follow from them. Then we shall discuss their significance and the significance of C itself.

9.5. We are primarily interested in those $\mathring{a}_n \in \mathring{C}_n$ which correspond to (hereditarily) total objects: following Ershov (1974) we call them <u>everywhere defined</u> ('$\mathring{ED}_n$'). For a recursion theory we shall also be interested in partial functions of total objects, so we define $C^*_n$ to be the set of partial maps from $C_n$ into $\omega$.

We set $ED_o = \omega$. We define a map $t_1:\mathring{C}_1 \rightarrow C^*_1$ by

(1) $$t_1(\mathring{\alpha})(x) \simeq \mathring{\alpha}(x).$$

Then we set $ED_1 = t_1^{-1}(C_1)$. Observe that $t_1$ is not one-to-one. For $t_1^{-1}(\lambda x.k)$ has two members; namely the finite element $\lambda\mathring{x}.k$, and the function $\lambda x.k$ (which is undefined at $\iota$) which is extended by $\lambda\mathring{x}.k$. We shall think of $t_1^{-1}(\alpha)$ as an equivalence class, denoted by $[\alpha]$, for the equivalence relation:

$$\dot\alpha \sim \dot\beta \longleftrightarrow_{Df} \dot\alpha, \dot\beta \in ED_1 \wedge t_1(\dot\alpha) = t_1(\dot\beta).$$

We wish now to define a $t_2{:}\dot C_2 \to C_2^*$ which will satisfy

(2) $\qquad t_2(\dot F)(t_1(\dot\alpha)) \simeq \dot F(\dot\alpha)$ for all $\dot\alpha \in ED_1$.

We accomplish this by the definition:

(3) $\qquad t_2(\dot F)(\alpha) = y$ if $\forall \dot\alpha \in [\alpha].\dot F(\dot\alpha) = y$,

is undefined otherwise.

Then we define $ED_2 = t_2^{-1}(C_2)$, or, equivalently

(4) $\qquad ED_2 = \{\dot F\colon \forall \dot\alpha \in ED_1.\dot F(\dot\alpha)\downarrow\}$.

As before we think of $ED_2$ as being partitional into equivalence classes $[F] = t_2^{-1}(F)$. Roughly speaking, different members of $[F]$ correspond to different associates of $F$; the consideration of 3.24 show that $[F]$ has the cardinality of the continuum, has no minimal elements (w.r.t. $\subseteq$) but does have a greatest element (corresponding to the principal associate for $F$). In fact, $([F], \subseteq)$ is a complete lattice without least element. If we define

(5) $\qquad m_2(F) = \mathrm{Sup}[F]$,

then $m_2$ is the embedding of $C_2$ in $\dot C_2$ mentioned by Feferman; it is a right inverse of $t_2$.

The extension of these definitions to higher pure types is immediate:

(6) $t_{n+1}\colon \dot C_{n+1} \to C_{n+1}^*$ is defined by $t_{n+1}(\dot a_{n+1})(b_n) \simeq (\mu y)(\forall \dot b_n \in [b_n])$
$(\dot a_{n+1}(\dot b_n) = y)$;

(7) $ED_{n+1} = t_{n+1}^{-1}(C_{n+1}) = \{\dot a_{n+1}\colon \forall \dot b_n \in ED_n.\dot a_{n+1}(\dot b_n)\downarrow\}$;

(8) $[a_{n+1}] = t_{n+1}^{-1}[a_{n+1}]$; (9) $m_n(a_n) = \mathrm{Sup}[a_n]$.

9.6. Hyland in (1975) stresses, in effect, that to get a true view of the relation between $\dot C$ and $C$ one must fix one's attention on the map $t$ and the equivalence classes of its inverse, rather than on the embedding $m$. Ershov (1974) provides a number of examples which emphasise the naturalness of $t$ and the artificiality of $m$. We state here some of the relevant facts.

(1) The topology of, say, $\dot C_2$ makes $\dot C$, a limit space: with each $\dot F$ is associated with the filter $\{X\colon \exists A \leq \dot F.\ U_A \subseteq X\}$ generated by the neighbourhoods of $\dot F$. If we apply $t_2$ then the filter associated with each $\dot F \in [F]$ is taken to a filter; the set of all these filters for a given $F \in C_2$ is precisely the limit-space structure described in 3.5. This structure is thus the finest which makes $t_2$ continuous.

(2) To make $t_2$ continuous in the topological sense we may define a topology $\tau$ on $C_2$ by taking as the open sets those $X \subseteq C_2$ such that $t_2^{-1}(X)$ is open in $\mathring{C}_2$. This is the 'induced topology' mentioned in 3.6.

(3) Neither $m_2$, nor any other embedding of $C_2$ in $\mathring{C}_2$ is continuous in the sense of limit spaces. Even $m_1$ is not continuous since it takes $\lambda x.k$ into the isolated point $\lambda \mathring{x}.k$ of $\mathring{C}_1$.

(4) Ershov shows (1974 Example 3) that m does not preserve composition, nor (for appropriate, not pure, types) application. For example

$$m_2(\lambda\alpha.(\lambda x.0)(F\alpha)) \neq \lambda\mathring{\alpha}.m_1(\lambda x.0)(m_2F)(\mathring{\alpha})).$$

9.7. In §3 and in this section we have given a number of reasons for adopting C as the correct definition of 'hereditarily total continuous object of finite type'. We are now in a position to argue that the correctness of the definition follows from the following two premises.

(A) Hereditarily total objects are extensions.

(B) They can be approximated by finitely presented hereditarily partial objects.

(B) means that the degree of approximation will be judged by the topology of $\mathring{C}$. As we have seen already at type 1 and certainly at type 2 there will be many different processes of approximation to a given object, F say; the course of such a process of approximation is described by an object $(\mathring{F})$ of $\mathring{C}_2$. Since there are many of these, they must be thought of as different intensions corresponding to the single extension F. Since we are positing extensions, the passage from an intension to an extension is a fundamental and natural one, while the choice of a particular intension to correspond to a given extension must be, at least to some extent, artificial. Further, if we have some continuous operation on extensions, then we must be able to approximate to its value at a given extension F by using <u>any</u> of the processes which approximate to F. And this is <u>all</u> we require of a continuous operation. Thus the map $t:\mathring{C} \to C$ is the fundamental one, and the structure for defining continuity on C must be the finest which makes t continuous. But, as indicated in 9.6, these facts completely determine C and its limit-space or topological structure.

An argument is never so good as when it is against something. The first thought of several investigators in this field has been to topologise $C_1 \to \omega$ by taking as a basis the sets

$$V_A = \{t_2(\mathring{F}) : \mathring{A} \subseteq \mathring{F}\} = t_2(U_A), \quad (\mathring{A} \in FCN_2).$$

We therefore call it the naive topology. [See, for example, the definitions at the bottom of pages 223 and 228 of Ershov (1972). This led him to claim (page 241), erroneously, that the class thus obtained coincided with the Kleene-Kreisel continuous functions. In (1974) he corrects - without acknowledging - his error]. The naive topology is easy to handle, and makes m continuous. We have already seen in 3.6 that it is incompatible with the ideas which motivated Kleene's work. Our argument shows exactly what is wrong with it - the topology is not extensional. One may be able to recognise from a particular definition of F that it is constantly zero (and so belongs to $V_{(\emptyset,0)}$), but this recognition is not extensional and does not apply to total objects. And, finally, if one is not interested in total objects as extensions then there is no reason to construct C at all - everything can be done inside $\mathring{C}$.

9.8. We close this section with two remarks

(1) The effective operations are adequately represented in $\mathring{C}$. One can mimic the definitions of 9.5 so as to get a subset $RED_n$ (recursively everywhere defined) of $\mathring{C}_n$ and a many-one map $r_n$ from $RED_n$ onto $R_n$.

(2) One might suppose (indeed both Kreisel in (1958) and Hyland in (1975) do seem to suppose) that a formal consistent neighbourhood such as A would appear as one of many possible intensions for the set $V_A$ of total functions. But this is so only trivially. For, at any type, we have:

$$\text{if } V_A = V_B \text{ then } \mathring{A} = \mathring{B} \text{ and } U_A = U_B$$

so that a given $V_A \subseteq C_n$ determines a unique irredundant $A' \in FCN$ with $V_{A'} = V_A$.

The proof is an immediate consequence of the conditions given in the literature (Tait (1963), Hyland (1975)) for deciding whether $V_A \subseteq V_B$. It is also stated, in effect but without comment, by Ershov (1974) Remark 3, page 218).

## §10. Recursion on $\mathring{C}$ and C

In this section we discuss the work described by Feferman in his paper in our own terms.

Because every function $\mathring{C}_{n+1}$ is determined by its values on the finite elements of $C_n$, there can be no doubt about which elements are to be regarded as recursive.

10.1. Definition A member $\dot{a}$ of $\dot{C}_{n+1}$ is partial recursive ('$\in$ PR$\dot{C}$') iff there is a partial recursive function $\phi:\omega \xrightarrow{\bullet} \omega$ such that

$$A \in FCN_n \quad \dot{a}(\dot{A}) \simeq \phi(<A>),$$

where $<A>$ is the numerical code for A.

Because '$A \in FCN_n$' and '$\dot{A} \subseteq \dot{B}$' are decidable there are a number of different natural definitions, all of which are equivalent to the above.

It should be observed that the usual Post-Smullyan type inductive definitions for the graphs of partial recursive functions do not provide a monotonic inductive definition of PRC; for they proceed via auxillary relations ( e.g. $\{(e,x,y): T(e,x,y)\}$). Hence the interest of definitions by schemata as discussed by Feferman.

The obvious way to set up a definition of 'partial recursive' in C is to use the maps $t_n$ of 9.5(6).

10.2. Definition We say that a partial function $a:C_n \xrightarrow{\bullet} \omega$ is partial recursive and write '$a \in PRC_{n+1}$ iff there is an $\dot{a} \in PR\dot{C}_{n+1}$ such that $a = t_{n+1}(\dot{a})$.

It is easy to see that this definition coincides with the definition given by Feferman (his section 7) and, when a is total, with Kleene's definition of recursively countable (cf. 3.8). It is also equivalent to an extension of our notion of associate-invariant recursion. For $a \in PRC_{n+1}$ iff there is an index e such that

$$a(b_n) = y \text{ iff } \forall \beta.[\beta \text{ is an associate for } b \rightarrow \{e\}(\beta) = y].$$

All the stated equivalences follow readily from the fact that there is a natural map from the associates of b onto the equivalence class $[b] = t_n^{-1}(b)$.

10.3. Because of the universal quantifier over $ED_n$, the definition of $PRC_{n+1}$ is very strong. For example, there is a partial recursive $\Delta: C_2\times\omega \xrightarrow{\bullet} \omega$ such that, for any $F_2 \in RC_2$, $\{x: \Delta(F,x)\downarrow\}$ is a complete $\Pi^1_2$ predicate. Unlike PR$\dot{C}$ or RC, PRC is thus not closed under substitution for higher type arguments. In this it resembles the computable functionals and the example used in Kleene 1963 also applies to PRC. Let

(1) $\dot{\alpha}_m(x) \simeq (\mu z)(z=0 \wedge \neg \exists y < xT(m,m,y))$.

Let $\Delta(F,m) \simeq F(\dot{\alpha}_m)$. Then $\Delta \in PRC_3$, since $\Delta = t_3(\lambda\dot{F}\lambda m.\dot{F}(\dot{\alpha}_m))$. But the domain of $\lambda m.\Delta(F,m)$ is $\{m: \forall y. \neg T(m,m,y)\}$ and so, for any given $F \in RC_2$, $\lambda m.\Delta(F,m) \notin PRC_1$.

10.4. Feferman states that $PRC_3$ is not closed under S.8. We shall disprove this statement, but will first point out why the matter is problematic.

Let $\Phi$: $\mathring{C}_0 \times \mathring{C}_2 \xrightarrow{\bullet} \omega$ belong to $PR\dot{C}$ and let $\Phi = t_3(\dot{\Phi})$ (with the obvious modification required by the presence of the numerical argument).

Let $\dot{\Gamma} = \lambda\dot{F}.\dot{F}(\lambda x.\dot{\Phi}(x,\dot{F}))$,

and $\Gamma = \lambda F.F(\lambda x.\Phi(x,F))$.

Suppose that $\lambda x.\Phi(x,F)$ is not total; it could nevertheless be the case that

$$\lambda x.\dot{\Phi}(x,\dot{F}) \in \text{Dom } \dot{F} \text{ for all } \dot{F} \in [F]$$

(although we have not constructed an example for this). And then $\Gamma(F)\uparrow$ but $t_3(\dot{\Gamma})(F)\downarrow$, so that $\Gamma \neq t_3(\dot{\Gamma})$. But one cannot conclude from this that $\Gamma \notin PRC$.

10.5. <u>Theorem</u> $PRC_3$ is closed under S.8.

W shall show that there is $\dot{\Psi} \in PRC_3$ such that

(1) $\forall \dot{F} \in [F].\dot{\Psi}(\dot{F}) = 1$ iff $\lambda x.\Phi(x,F)$ is total.

Then $\Gamma = t_3(\dot{\Lambda})$, where

$$\dot{\Lambda}(\dot{F}) \simeq \dot{\Psi}(\dot{F}).\ \dot{F}(\lambda x.\dot{\Phi}(x,\dot{F})),$$

and this establishes the theorem.

Let $\mathring{O}_m$ be defined by: $\mathring{O}_m(x) = 0$ if $x < m$, undefined otherwise. We will construct $\dot{\Psi} \in PR\dot{C}_3$ such that, for any $\dot{F} \in ED_2$

(2) $\dot{\Psi}(\dot{F}) = 1 \longleftrightarrow \forall x[\dot{F}(\mathring{O}_x)\downarrow \vee \dot{\Phi}(x,\dot{F})\downarrow]$ (where 'v' is the strong 'or'). Evidently if $\lambda x.\Phi(x,F)$ is total then $\forall \dot{F} \in [F].\dot{\Psi}(\dot{F}) = 1$. Conversely suppose that $\dot{\Phi}(x,\dot{F})\uparrow$ for some $\dot{F} \in [F]$ and least possible x. If $\dot{F}(\mathring{O}_x)\uparrow$ then $\dot{\Psi}(\dot{F})\uparrow$. If $\dot{F}(\mathring{O}_x)\downarrow$, let $\dot{F}'$ be defined by

$$\dot{F}'(\dot{\alpha}) \sim \dot{F}(\dot{\alpha}) \text{ if } \exists x.\ \dot{\alpha}(x) \neq 0,$$
$$\text{or if } \dot{\alpha} \supseteq \mathring{O}_{x+1},$$
$$\text{is undefined otherwise.}$$

Evidently $\dot{F}' \subseteq \dot{F}$, so $\dot{\Phi}(x,\dot{F}')\uparrow$; also $\dot{F}'(\mathring{O}_x)\uparrow$.

Thus, by (2), $\dot{\Psi}(F')\uparrow$, and $\dot{F}' \in [\dot{F}]$. We have shown that (1) follows from (2).

To show that there is a partial recursive $\dot{\Psi}$ satisfying (2), set

$$\dot{\Psi}_1(\dot{A}) = 1 \text{ if } \exists y[\dot{A}(\mathring{O}_{y+1})\downarrow \wedge \dot{A}(\mathring{O}_y)\uparrow \wedge \forall x \leq y.\dot{\Phi}(x,\dot{A})],$$
$$\text{undefined otherwise.}$$

Since $\dot{A}(\dot{d})\downarrow$ is decidable for any $\dot{A}$, $\dot{d}$, $\dot{\Psi}_1(\dot{A})$ is a partial recursive function of <A>; and since if $\dot{\Psi}_1(\dot{A}) = 1$ and $\dot{A} \subseteq \dot{B}$ then $\dot{\Psi}_1(\dot{B}) = 1$, we see that $\mathring{\Psi}_1 \in \mathring{PRC}_3$. And if $\mathring{\Psi}_1(\mathring{A}) = 1$ and $\mathring{A} \subseteq \mathring{F}$ then $\mathring{F}$ satisfies the RHS of (2). Finally if $\mathring{F} \in ED_2$ and satisfies the RHS of (2) let y be the least number such that $\mathring{F}(\mathring{O}_{y+1})\downarrow$. Then for each $x \leq y$ there must be a $B_x$ such that $\mathring{\Phi}(x,\mathring{B}_x)\downarrow$ and $\mathring{B}_x \subseteq \mathring{F}$. Pick one such $B_x$ for each $x \leq y$ and take

$$A = \{(\mathring{O}_{y+1},\mathring{F}(\mathring{O}_{y+1}))\} \cup \bigcup\{B_x: x \leq y\}.$$

Then $\Psi_1(\mathring{A}) = 1$, so $\Psi_1(\mathring{F}) = 1$.

Thus $\Psi_1$ satisfies (2) and the theorem is proved.

Corollary $PRC_3$ is closed under S1-S9. (Closure under the other schemes is easily verified).

10.6. Throughout this paper we have emphasised our interest in total objects. It therefore seems appropriate to end with a question that differs from that asked by Feferman.

Question Does there exist a natural monotonic inductive process which generates RC (instead of PRC)?

Such a process would be interesting for at least two reasons. Firstly it should enable one to prove things about RC without detours through PRC or $\mathring{PRC}$. Secondly one would expect to get a method of generating the continuous functionals by relativising the process to arbitrary functions. The problematic type is 3. The answer to the question at type 1 is, roughly, negative. The answer at type 2 (where one can use $RC_1$) is positive and given by Brouwer's definition of the constructive functionals. The rather strong properties of PRC exhibited in this section suggest that it may be easier to answer our question than Feferman's.

References:

M.J. Beeson 1975, The underivability in intuitionistic formal systems of theorems on the continuity of effective operations, J.S.L. 40 321-346.

J.A. Bergstra 1976, Computability and continuity in finite types, Dissertation, Utrecht.

J.A. Bergstra and S.S. Wainer 1976, The "real" ordinal of the 1-section of a continuous functional, paper presented at the Oxford Logic Colloquium.

Yu.L. Ershov 1972, Computable functionals of finite type, Algebra and Logic 11 203-242 (367-437 in Russian).

Yu.L. Ershov 1974, Maximal and everywhere defined functionals. Algebra and Logic 13 210-225 (374-397 in Russian).

R.O. Gandy 1962, Effective operations and recursive functionals (abstract), J.S.L. 27 378-379.

R.O. Gandy 1967, Computable functionals of finite type I, in: Sets, Models and Recursion Theory, North-Holland, Amsterdam 1967.

P.G. Hinman 1973, Degrees of continuous functionals, J.S.L. 38 393-395.

J.M.E. Hyland 1975, Recursion theory on the countable functionals, D.Phil. Thesis, Oxford.

J.M.E. Hyland 1977, Filter spaces and continuous functionals, to appear.

S.C. Kleene 1959a, Recursive functionals and quantifiers of finite types I, T.A.M.S. 91 1-52.

S.C. Kleene 1959b, Countable functionals, in: Constructivity in Mathematics, North-Holland, Amsterdam 1959.

S.C. Kleene 1962a, Turing machine computable functionals of finite types I, in: Logic, Methodology and Philosophy of Science, Stanford Univ. Press, Stanford 1962.

S.C. Kleene 1962b, Lambda definable functionals of finite types, Fund. Math. 50 281-303.

S.C. Kleene 1962c, Herbrand-Godel style recursive functionals of finite types, Proc. Symp. Pure Math. vol.V 49-75.

S.C. Kleene 1963, Recursive functionals and quantifiers of finite types II, T.A.M.S. 108 106-142.

G. Kreisel 1959, Interpretation of Analysis by means of functionals of finite type, in: Constructivity in Mathematics, North-Holland, Amsterdam 1959.

G. Kreisel, D. Lacombe and J.R. Shoenfield 1959, Partial recursive functionals and effective operations, in: Constructivity in Mathematics, North-Holland, Amsterdam 1959.

J. Myhill and J.C. Shepherdson 1955, Effective operations on partial recursive functions, Ziet. Math. Log. Grund. Math. 1 310-317.

D. Norman 1976, On a problem of S. Wainer, Oslo preprint.

H. Rogers Jr. 1967, Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York 1967.

B. Scarpellini 1971, A model for bar recursion of higher types, Comp. Math. 23 123-153.

D. Scott 1970, Outline of a mathematical theory of computation, Proc. 4th Annual Princeton Conference on Information Science and Systems, 169-176.

D. Scott 1976, Data types as lattices, SIAM Journal on Computing 5 522-587.

W.W. Tait 1963, A second order theory of functionals of higher type, Stanford Seminar on the foundations of Analysis.

29