ANNALES SCIENTIFIQUES
DE L'UNIVERSITÉ DE CLERMONT-FERRAND 2
*Série Mathématiques*

R. O. GANDY

# General recursive functionals of finite type and hierarchies of functions

*Annales scientifiques de l'Université de Clermont-Ferrand 2*, tome 35, série *Mathématiques*, n° 4 (1967), p. 5-24

<http://www.numdam.org/item?id=ASCFM_1967__35_4_5_0>

# GENERAL RECURSIVE FUNCTIONALS OF FINITE TYPE AND HIERARCHIES OF FUNCTIONS

R.O. GANDY
UNIVERSITY OF MANCHESTER, ENGLAND

## INTRODUCTION

Kreisel (in GK1) has argued that if one takes the totality N of natural numbers as a basis, then the number theoretic functions which can be *predicatively* defined are exactly the hyperarithmetic (HA) functions. But when we go beyond the hyperarithmetic hierarchy we can, nevertheless, distinguish different degrees of impredicativity. For example, let $\sigma$ be the least ordinal such that all the constructible (*sensu* Gödel) number-theoretic functions of order less than $\sigma$ form a model of classical second-order arithmetic, and let S be the first binary relation over N to be constructed having order-type $\sigma$. These definitions are surely thoroughly impredicative ; $\sigma$ is, as it were, defined from above[1]. On the other hand, Spector has shown (in CS) that a complete $\pi^1_1$ predicate (e.g., Kleene's O or Spector's W) can be expressed in the form

$$(E\alpha)_{HA}\ A(a, \alpha)$$

where A is arithmetic. This is not a predicative definition because the totality over which the quantifier ranges has not been given a previous definition. But it is a definition, so to speak, from below, because each object in the range of the quantifier has been previously (and predicatively) defined. I believe that Russell would have admitted it as a non-circular definition. One of the objects of the work prescribed in this lecture is to study and to characterise more exactly the class of number theoretic function which can be defined from below[2].

---

(1) I do not hold that there is anything unsound or philosophically objectionable in the use of thoroughly impredicative definitions. But it is clear (witness the recent independence results obtained by Cohen) that the more impredicative the definition of an object or class, the less the information which we can expect to obtain about it from that definition.

(2) I do not now regard this as a very fruitful programme. If the operation of collecting together all previously defined functions is only repeated a finite number of times, then one obtains just those functions which are primitive recursive in $E_1$. If the operation is repeated any transfinite number of times then one obtains the functions of ramified analysis. If one seeks to control the transfinite applications of the operation by requiring that the ordinal used should be the type of a well-ordering which has already been obtained, then I believe that the functions one obtains are just those which result from applying the operation less than $\xi_0$ times, where $\xi_0$ is the first $\xi$-number as defined in section 4.

More promising, I consider, is the programme of successive diagonalisation which is considered in this lecture. But for functionals of type 3 or greater, $D_{max}$ (see section 5) is thoroughly impredicative. Therefore, if one seeks to obtain a more detailed knowledge of particular hierarchies of functions by avoiding operations which are thoroughly impredicative, one should investigate the functionals which arise by application of $D_{min}$. I have found this difficult to do.

However, the line of attack to be followed here is a rather indirect one. Let us first consider the class HA of hyperarithmetic functions. One way of expressing the presupposition of N is to permit curselves the use of Kleene's functional E defined by

$$E(\alpha) = \begin{cases} 0 & \text{if } (Ex)\ (\alpha(x) = 0) \\ 1 & \text{if not.} \end{cases}$$

Now Kleene has shown (in SCK1) that HA coincides with the class of functions [E] which are recursive in E. This suggests immediately where we should look if we wish to extend our notion of function beyond HA - namely at the results of diagonalising [ E ]. In other words, we are to consider the predicate

$$D(x, E) \equiv \{x\}\ (E) \quad \text{is defined,}$$

Where $\{x\}\ (E)$ denotes the partial recursion with index number x and type 2 argument E. Or more generally, we consider

$$D(x, \alpha, E) \equiv \{x\}\ (\alpha, E) \quad \text{is defined.}$$

It is not hard to show that the former predicate has the same Turing degree as Kleene's O, as here and that the latter predicate is equivalent to $x \in O^{\alpha}$ and to the functional $E_1$ (introducel by Tugué in T) defined by

$$E_1(\alpha) = \begin{cases} 0 & \text{if } (\beta)\ (Ex)\ (\alpha(\bar{\beta}(x)) = 0) \\ 1 & \text{if not.} \end{cases}$$

'Equivalent' means that each of the three objects considered is recursive in either of the others. We remark that E itself is equivalent to D, where

$$D(x, \alpha) \equiv \{x\}\ (\alpha) \quad \text{is defined.}$$

These considerations suggest investigation of the class of functions generated by repeated application of the following three principles :

(i) use of functionals of finite type ;

(ii) use of general recursions applied to such functionals ;

(iii) use of diagonalisation over classes generated by (ii).

## 2 - RECURSIONS IN FUNCTIONALS OF FINITE TYPE

In this section we give a brief resumé of some of the definitions and results of SCK1. A recursive functional has a numerical value and a finite number of arguments ; each argument ranges over a *pure* type - type 0 is N and pure type n + 1 consists of the functionals with numerical values and a single argument of type n. A partial recursive functional (p. r. f. ) is defined, perhaps in terms of other p. r. f. s. , by a particular instance of one of the schemata S 1 - 9 ; to each such instance a particular index number is allocated. This number encodes the number and the types of the arguments of the p. r. f. which is being defined, the number of the schema, and the index number (s) of the p. r. fs. , (if any) in terms of which the definition is being made.

S 1 - 3 define respectively the successor, constant and identity functionals. S4 defines the result of substituting the value of a p. r. f. for a numerical argument of an other p. r. f. S 5 is the

schema for primitive recursion ; the numerical value of the functional for a numerical argument n + 1 is defined in terms of the value for argument n - this contrasts with the schema used by Gödel in KG. Instances of S 6 permute the arguments of a given type ; this schema can be dropped if in each instance of the other schema (and their index numbers) we are allowed to specify freely which arguments of appropriate type are the relevant ones ; whereas Kleene always takes the first argument as the relevant one. S 7 applies a type 1 argument to a type 0 argument. S $8^j$ (with $j \geq 2$) applies a type j argument to a given previously defined functional ; in symbols :

$$\Phi(\alpha^j, \mathfrak{u}) = \alpha^j(\lambda\tau^{j-2}\ \chi(\tau^{j-2}, \alpha^j, \mathfrak{u}))$$

where $\mathfrak{u}$ denotes a string of arguments, and $\chi$ has been previously defined. The computation of $\Phi(\alpha^j, \mathfrak{u})$ for particular $\alpha^j, \mathfrak{u}$, is said to be *defined* if the computation of $\chi(\tau^{j-2}, \alpha^j, \mathfrak{u})$ is defined for *every* functional $\tau$ of type j-2. S 9 is a reflection schema ; one of the type 0 arguments c is considered as an index number, and the value of the functional being defined for a given set of arguments is the value of the p. r. f. with index number c for a specified selection from that set of arguments ; in symbols

$$\Phi(c, \mathfrak{u}, \mathfrak{b}) = \{c\}\ (\mathfrak{u})$$

Naturally such a computation will only be defined if c has the form of an index number for the specified list of arguments and the computation on the right is defined. In conjunction with S 4, S 9 allows the progress of a computation to be determined by the result of a subsidiary computation.

Functionals which can be defined without the use of S 9 are called *primitive recursive* ; they are automatically defined for all values of their arguments. For the definition of p. r. fs (S9 permitted) S5 can be omitted. If a p. r. f. is defined for all values of all arguments it is said to be a *total* or a *general recursive* functional. When the types of the arguments are all 0 or 1, these definitions are equivalent to the customary ones.

By the description of a computation we mean an index number and a set of arguments of appropriate types. We write $\{z\}\ (\mathfrak{u})$ both for the description of a computation and for its value (if this is defined). The course of a computation $\{z\}\ (\mathfrak{u})$ can be set out on a (horizontal) tree. At each point there stands a description of a computation and, if that computation is defined, eventually there will also stand its value. We start by placing $\{z\}\ (\mathfrak{u})$ at the vertex at the extreme left of the tree. Suppose $\{y\}\ (\mathfrak{b})$ stands at a point P ; the progress of the computation beyond this point is set out as follows.

(i) If y refers to an outright calculation by schemats S1-3, S7, then P is a *tip*, and at it we insert the appropriate value.

(ii) If y refers to S4, $\{y\}\ (\mathfrak{b}) = \{y_1\}\ (\{y_2\}\ (\mathfrak{b}), \mathfrak{b})$ say, then P is a *node*. Immediately to its right there stand two points $Q_1, Q_2$ ; $Q_1$ is above $Q_2$. At $Q_2$ we place the computation description $\{y_2\}\ (\mathfrak{b})$ ; if this is defined, eventually its value - u say - will be placed at $Q_2$. Then we place the computation description $\{y_1\}\ (u, \mathfrak{b})$ at $Q_1$. If this is defined, then, when we eventually place a value at $Q_1$, we shall also place it at P.

(iii) If y refers to S9, $\{y\}\ (a, \mathfrak{b}, c) = \{a\}\ (\mathfrak{b})$ then just one point Q stands immediately to the right of P. At Q we place the computation description $\{a\}\ (\mathfrak{b})$, and when a value is placed at Q it is also to be placed at P.

(iv) If y refers to $S8^j$,

$$\{y\}(\alpha^{j-1}, \mathfrak{b}) = \alpha^{j-1}(\lambda\tau^{j-2}\{y_1\}(\tau^{j-2}, \alpha^{j-1}, \mathfrak{b})),$$

then P is a *branch point* ; immediately to its right stand infinitely many points $Q_\tau$, one for each $\tau$ in type j-2. At $Q_\tau$ stands the computation description $\{y_1\}(\tau, \alpha^{j-1}, \mathfrak{b})$. When values have been placed at all the $Q_\tau$, the functional $\lambda\,\tau^{j-2}\{y_1\}(\tau^{j-2}, \alpha^{j-1}, \mathfrak{b})$ is defined, and the value of $\alpha^{j-1}$ applied to it is placed at P. Now we can use the computation tree to obtain normal forms for '$\{z\}(\mathfrak{u}) = w$' and '$\{z\}(\mathfrak{u})$ is defined' ; the tree plays a role analogous to that played be a derivation in recursive function theory. Towards that and we make a number of remarks.

1. There are only a finite number of points ('predecessors') to the left or any given point.

2. The non-numerical arguments in the computation description at any point form a selection from the list $\mathfrak{u}'$ of the non-numerical members of $\mathfrak{u}$.

3. Any type can be mapped one-to-one into any higher type. Hence, if r is the highest type occurring in $\mathfrak{u}$, the different branches at a node or a branch point can each be described by an appropriately chosen object in type r-2. Therefore a point on the tree can be uniquely specified by a finite sequence of type r-2 objects, and hence by a single type r-2 object. Such an object is called a position.

4. By remark 2, we can specify the computation description (for a given tree - so that $\mathfrak{u}$ is given) at any position $\gamma$ of the tree by a single number $\beta(\gamma)$. Also the value (if defined) at $\gamma$ will be denoted by $\eta(\gamma)$. Thus $\beta$ and $\eta$ are functionals of type r-1.

5. If $\gamma_0$ is the vertex then evidently $\beta(\gamma_0)$ will be given primitively recursively in terms of z and $\mathfrak{u}_0$, where $\mathfrak{u}_0$ is the numerical part of $\mathfrak{u}$. Examination of the case (i) - (iv) shows that elsewhere $\beta(\gamma)$ can be simply defined in terms of the value of $\beta$ at the immediate predecessor of $\gamma$, $\gamma$ itself, and, if $\gamma$ is the upper point at a node, the value of $\eta$ at the point immediately below $\gamma$. So we have

$$\beta(\gamma) = \theta(z, \mathfrak{u}_0, \eta, \gamma)$$

where $\theta$ is primitive recursive.

6. The value of $\eta$ at a position $\gamma$ is easily determined from $\beta(\gamma)$, $\mathfrak{u}$, and the values of $\eta$ at the points immediately to the right of $\gamma$. There may be branches of arbitrarily great length through the position $\gamma$. Hence, even if $\beta(\gamma)$ is defined, we cannot give a primitive recursion for $\eta(\gamma)$. But we can give a primitive recursive predicate $L(z, \mathfrak{u}, \gamma, \eta)$ which expresses that the value of $\eta$ at $\gamma$ is correctly related to its values at the points immediately to the right of $\gamma$. $\eta$ is then said to be *locally correct* at $\gamma$.

7. We are interested only in the values $\beta(\gamma)$, $\eta(\gamma)$ when $\gamma$ describes a point on the tree. The precise expression of this latter predicate depends on the particular way in which the choice at nodes, or at branch points where $j < r$, is represented by a type r-2 object. But a little thought will show that the condition can be expressed by quantifying over certain variables of type r-3, a predicate which is primitive recursive in those variables, $\beta$ and $\gamma$. (For example, if choosing the lowest point at a node is represented by the zero functional of type r-2, then expressing that such a choice has been made will obviously require universal type r-3 quantifier). Thus using remark 5 we obtain :

$$\gamma \text{ is a position} \equiv (Q\tau)P(z, \mathfrak{u}, \beta, \gamma, \ldots)$$

where .... denotes the type r-3 variables, and $(Q\tau)$ quantifiers over them. (In fact a single universal quantifier suffices.) In case r = 2 the quantifiers are omitted. To emphasise the dependance on $\eta$ we shall refer to $\eta$-positions.

8. To clarify the relation of the definitions in remarks 6 and 7 to $\eta$, we introduce the following definition : if $\gamma_1$, $\gamma_2$ are two putative positions (i.e. represents finite sequences of type r-2 objects) then $\gamma_2$ is said to be *below* $\gamma_1$ if there is an $\eta$-position $\delta$ which is a node and $\gamma_1$ (resp. $\gamma_2$) is an extension of the upper (resp. lower) $\eta$-position immediately to the right of $\delta$. Then the value of $\beta(\gamma)$, and whether $\gamma$ is an $\eta$-position *depend only on the values of* $\eta$ *at* $\eta$*-positions which lie below* $\gamma$. Furthermore 'below' is a well-founded relation. We say that $\gamma_1$ is *beyond* $\gamma_2$ if $\gamma_1$ (as a sequence) is an extention of $\gamma_2$. Now $\eta(\gamma)$ is determined by $\beta(\gamma)$ and the values of $\eta$ at points beyond $\gamma$ ; hence if $\eta$ is locally correct at $\gamma$ then its value there is *uniquely determined by its values at all the* $\eta$*-positions which are below or beyond* $\gamma$.

We are now able to obtain the required normal forms. First suppose that $\{z\}(\mathfrak{u})$ is defined. Then every branch of the tree comes to a tip, and every point $\gamma$ of the tree there will be assigned a value $\eta(\gamma)$. Evidently this $\eta$ is locally correct, and the $\eta$-positions are just the points on the tree. Suppose, conversely, that $\eta'$ is locally correct at all $\eta'$-positions. Observe that the relation 'below or beyond' between points on the tree is well-founded ; since also 'below' is well-founded both for $\eta$ and $\eta'$, it follows by transfinite induction from remark 8 that the $\eta'$-positions coincide with the points on the tree, and that the values of $\eta$ and $\eta'$ are the same at all these positions. In particular, they are the same at the vertex. Hence

$$\begin{aligned}\{z\}(\mathfrak{u}) = w \equiv (\eta)\,[(\gamma)\,(&\gamma \text{ is an } \eta\text{-position} \longrightarrow \\ &\eta \text{ is locally correct at } \gamma) \longrightarrow \eta(\gamma_0) = w] \\ \equiv (E\eta)\,[(\gamma)\,(&\gamma \text{ is an } \eta\text{-position} \longrightarrow \\ &\eta \text{ is locally correct at } \gamma)\ \&\ \eta(\gamma_0) = w]\end{aligned}$$

Thus, by remarks 6, 7 and manipulation of quantifiers, we see that if $\{z\}(\mathfrak{u})$ is defined, then

$$\{z\}(\mathfrak{u}) = w \equiv (\alpha^{j-1})\,(E\gamma^{j-2})\,I(z, w, \mathfrak{u}, \gamma^{j-2}, \alpha^{j-1})\ ; \tag{2.1}$$

$$\equiv (E\alpha^{j-1})\,(\gamma^{j-2})\,J(z, w, \mathfrak{u}, \gamma^{j-2}, \alpha^{j-1})\ , \tag{2.2}$$

where I and J are primitive recursive in the indicated arguments. Consider now the case $\{z\}(\mathfrak{u})$ is not defined. There may be an infinite branch on the tree, or there may be a point at which the computation description $\{y\}(\mathfrak{b})$ is defined, but y has not the form of an index number for a schema with the arguments $\mathfrak{b}$. In the latter case let us conveniently continue the branch indefinitely by simply repeating the computation description at each successive point, so that in either case there is an infinite branch. Now there must be a *lowest* such branch, $\rho$ say, such that any branch which passes through the lower point, at a node where $\rho$ takes the upper point, does reach a tip. Hence a value is defined for all points below (points of) $\rho$, and will be correctly given by any $\eta'$ which is locally correct at all $\eta'$-positions lying below $\rho$. And the computation descriptions along $\rho$ will be correctly specified by the function $\lambda n.\ \theta(z, \mathfrak{u}_0, \rho_n, \eta')$ for any such $\eta'$. Here the branch $\rho$ is identified with the infinite sequence $\rho_n$ of positions along it. If r>2, then $\rho$ can be taken to be of type r-2 ; if r = 2, then $\rho$ will be of type 1.

Hence

$$\{z\}\ (\mathfrak{u}) \text{ is defined} \equiv (\eta)\ (\rho)\ [(\gamma)\ (\gamma \text{ is an } \eta\text{-position below } \rho \longrightarrow \eta \text{ is locally correct at } \gamma) \longrightarrow (En)\ (\beta\ (\rho_n) \text{ specifies a tip})].$$

Now it can be shown that :

$$\gamma \text{ is an } \eta\text{-position below } \rho \equiv (En)\ (\tau^{r-3})\ R(z, \mathfrak{u}_0, \tau^{r-3}, \gamma, \rho_n, \eta),$$

with primitive recursive R. Also 'x specifies a tip' is primitive recursive.

Thus for all $r \geq 2$ we have

$$\{z\}\ (\mathfrak{u}) \text{ is defined} \equiv (\alpha^{r-1})\ (E\gamma^{r-2})\ M(z, \mathfrak{u}, \gamma^{r-2}, \alpha^{r-1}) \tag{2.3}$$

with primitive recursive M.

Alternatively

$$\{z\}\ (\mathfrak{u}) \text{ is defined} \equiv$$
$$(E\eta)\ [(\gamma)\ (\gamma \text{ is an } \eta\text{-position} \longrightarrow \eta \text{ is locally correct at } \gamma)$$
$$\&\ (\rho)\ (En)\ (\beta(\rho_n) \text{ specifies a tip}]\ .$$

If $r > 2$, then this can be reduced to the form

$$(E\eta^{r-1})\ (\delta^{r-2})\ N(z, \mathfrak{u}, \delta^{r-2}, \eta^{r-1})\ . \tag{2.4}$$

If $r = 2$ it can be reduced to the form

$$(E\eta^1)\ [(x)\ A(z, \mathfrak{u}, x, \eta^1)\ \&\ (\rho^1)\ (En)\ B(z, \mathfrak{u}_0, n, \rho^1, \eta^1)]\ , \tag{2.5}$$

where A and B are primitive recursive. In general it cannot be further reduced, but we remark that if $E_1$ is primitive recursive in $\mathfrak{u}$, then the second term of the conjunction can be reduced to a primitive recursion in $z, \eta^1, \mathfrak{u}$, and so we have

$$\{z\}\ (\mathfrak{u}) \text{ is defined} \equiv (E\eta^1)\ C(z, \mathfrak{u}, \eta^1)\ ; \tag{2.6}$$

(the numerical quantifiers (x) is of course equivalent to a primitive recursion in E and hence also in $E_1$).

## 3 - FURTHER RESULTS ABOUT RECURSIVE FUNCTIONALS

In this section we shall suppose that $r = 2$ – i.e. that no variable in the list $\mathfrak{u}$ has type greater than 2. (The results can be extended to the case $r > 2$, but to do so a specific well-ordering of the objects in type r-2 must be assumed).

There is then a natural extension of the relation *below*. Namely, we say that a point R on the tree for $\{z\}\ (\mathfrak{u})$ is *beneath* the point S, if R is below S or if there is a branch point P such that R is on a branch through $Q_i$, S is on a branch through $Q_j$, and $i < j$. Then the relation 'beneath or beyond' gives a *total* ordering of points on the tree. If $\{z\}\ (\mathfrak{u})$ is defined then this is a well-ordering. If not, then there is a 'furthest down' branch $\rho$, such that every branch which (eventually) goes beneath $\rho$ comes to a tip. In this case the relation 'beneath or beyond' is a well-ordering of all the points on the tree which lie beneath $\rho$ ; and a unique value is assigned to all these points. Thus whether $\{z\}\ (\mathfrak{u})$ is defined or not, there is an ordinal $|\{z\}\ (\mathfrak{u})|$ which gives the order type of the well-ordered part of the tree. We define $\omega_1^{\mathfrak{u}} = \sup_z |\{z\}\ (\mathfrak{u})|$. This is a rational notation, because

it can be shown that if we adapt Kleene's system of ordinal notations, or Spector's notion of recursive well-orderings, by substituting 'recursive in $\mathfrak{a}$' for 'recursive', then the supremum of the ordinals represented by such notation is $\omega_1^{\mathfrak{a}}$ as defined above. Note that we could replace $\mathfrak{a}$ by $\mathfrak{a}'$, for it is only the non-numerical part of $\mathfrak{a}$ which is relevant.

The notion of the ordinal of a computation proves to be a very useful tool, especially if one also uses Kleene's E. In particular, suppose one has compared a given ordinal with the ordinals of the computations at the points $Q_i$ immediately to the right of a branch point P ; then using E one can also compare it with the ordinal of the computation at P. Given two computations from $\mathfrak{a}$, at least one of which is defined, there is a method which is recursive in $\mathfrak{a}$ and E for comparing the ordinals of the computation. (The method is fairly elaborate ; details will appear in ROG1). And from this one can obtain the following theorem :

3.1. There is a p.r.f. $\nu(z, \mathfrak{a}, \varepsilon^2)$ which satisfies

$$(Ex)\ (\{z\}\ (x, \mathfrak{a}) = 0) \longrightarrow \{z\}(\nu(z, \mathfrak{a}, E), \mathfrak{a}) = 0).$$

Of course the existence of such selection operators for partial predicates in recursive function theory is well known. An analogue of 3.1 for $r > 2$ can also be proved, but then a well-ordering of the type r-2 will appear as an argument of $\nu$ and E will be replaced by a functional $E^r$ which represents the existence operator for predicates with type r-2 arguments.

Theorem 3.1 obviously has a particularly simple form if E is recursive in $\mathfrak{a}$ ; and this will be the case in the applications we shall make. For the rest of this section *we therefore assume that E is recursive in* $\mathfrak{a}$ .

In what follows we are going to be concerned with formulae in which the quantifiers are limited in range ; this will be indicated by placing '$\restriction$ C' (where C is the range) after the binding occurence of the variable. We shall also apply this notation to the $\pi, \Sigma, \Delta$ notation of Addison-Mostowski-Shoenfield (N.B. $\Delta = \pi \cap \Sigma$). For example Spector's theorem mentioned in the introduction can be written as $0 \in \Sigma_1^1 \restriction HA$. If the recursive predicate which is the scope of the quantifiers involves constant functionals, these will be exhibited in parentheses. In the same parentheses, and separated from the constant symbols by a colon, we may place numerals to indicate the types of the arguments ; (we shall not need to indicate the number of arguments of a type - if this were required it could be done by giving each type numeral a subscript). We shall normally omit the numerals for all types except the highest. For example, we write $\pi_1^1(1)$ for a predicate of numbers and functions - though in many contexts we could omit the '(1)' ; it is obvious that $\Sigma_2^1(1) = \Sigma_1^1(E_1 : 1)$, while $\Sigma_2^1(2) \neq \Sigma_1^1(E_1 : 2)$. Spector's theorem can be written $\pi_1^1(0) = (\Sigma_1^1 \restriction HA)\ (0)$, but '0' cannot be replaced by '1', and it would therefore be misleading to omit the '0'.

Finally, we write $[\mathfrak{a}]$ to mean the set of all objects - possibly of a prescribed type - which are recursive in the list of functionals $\mathfrak{a}$ ; of course the numerical part of is irrelevant. It will never be necessary to distinguish between functionals and predicates, but, as in the previous paragraph, it may be necessary to exhibit the (highest) type (s) of the arguments ; we use exactly the same notation as above. Thus $[\mathfrak{a}]\ (0)$ means the set of all number-theoretic predicates or functions which are recursive in $\mathfrak{a}$ . Theorem 2.1 can be expressed by $[\mathfrak{a}](r) \subseteq \Delta_1^{r-1}(\mathfrak{a} : r)$, where the list $\mathfrak{a}$ may be empty, but must not include functionals of type greater than r.

Now we can state some corollaries to 3.1 :

3.11 (Axiom of choice for $[\mathfrak{u}]$). *If* R *is recursive, then*

$$(x)\,(E\alpha^1 \upharpoonright [\mathfrak{u}])\ R(x, \alpha^1, \mathfrak{u}) \longrightarrow (E\alpha^1 \upharpoonright [\mathfrak{u}])\ (x)\ R(x, \lambda t.\ \alpha^1(2^t 3^x), \mathfrak{u})\ .$$

3.12 $$[\mathfrak{u}](0) = (\Delta^1_1 \upharpoonright [\mathfrak{u}])\ (\mathfrak{u} : 0)$$

That the right hand side is included in the left hand side follows from 3.11 ; the converse is shown by examining the proof of 2.1, and establishing that all the bound variables may be limited to $[\mathfrak{u}]$.

3.12 is an analogue of the familiar equality

(*) $$\text{Recursive} = \Delta^0_1$$

Indeed, we shall continually find that when type 2 objects appear as parameters, then a function quantifier limited to functions recursive in those parameters is the analogue of a numerical quantifier in ordinary recursive function theory. For the particular case $\mathfrak{u} = E$, the basis of this analogy has been investigated by Kreisel (in GK2 and GK3). He proposes that the analogue of a natural number should be taken to be a hyperarithmetic set (or an index for it) ; the analogue of a recursive set is the intersection of a hyperarithmetic set of functions (or numbers) with the set of all hyperarithmetic functions (or notations for them)[1]. For ease of expression we will take for the analogue of N the set of all hyperarithmetic *functions* (rather than sets) ; this change is evidently of no importance. Then Kreisel's analogue for (*) can be written :

$$[E]\ (1 \upharpoonright [E]) = (\Delta^1_1 \upharpoonright [E])\ (E : (1 \upharpoonright [E]))\ ,$$

where $(1 \upharpoonright C)$ means that the predicates are to be restricted to type 1 arguments in [C] - but the predicates must still be defined for all arguments in type 1. Now the methods of proof of 3.12 also suffice to prove

3.13 $$[\mathfrak{u}]\ (1 \upharpoonright [\mathfrak{u}]) = (\Delta^1_1 \upharpoonright [\mathfrak{u}])\ (\mathfrak{u} : (1 \upharpoonright [\mathfrak{u}]))\ ,$$

which is just the generalisation to an arbitrary $\mathfrak{u}$ in which E is recursive of Kreisel's analogue of (*).

A further corollary of 3.1 is that there is an undefined computation $\{z\}(\mathfrak{u})$ whose ordinal is $\omega_1^{\mathfrak{u}}$. This may be compared with the fact that there is a recursive linear ordering whose initial well-ordered segment has ordinal $\omega_1$. And just as the latter fact leads to a proof of Spector's theorem (see ROG2), so does the former lead to a proof of :

3.2 $$\lambda x.\ D(x\ ;\ \mathfrak{u}) \in (\Sigma^1_1 \upharpoonright [\mathfrak{u}])\ (\mathfrak{u})$$

(where, as in the introduction, $D(x\ ;\ \mathfrak{u}) \equiv \{x\}(\mathfrak{u})$ is defined). This theorem shows that the remarks which were made in the introduction about obtaining a diagonal predicate for E by means of quantification from below also apply to any type 2 functional in which E is recursive. Hierarchies based on this idea will be discussed in section 5.

By using 3.1 it is easy to show that $\lambda x.\ D(x\ ;\ \mathfrak{u})$ is a complete predicate for $(\Sigma^1_1 \upharpoonright [\mathfrak{u}])\ (\mathfrak{u} : 0)$. Let us denote by $D[\mathfrak{u}]$ the set of all predicates $\lambda \mathfrak{b}.\ D(x\ ;\ \mathfrak{b}, \mathfrak{u})$ as x ranges over N ; we adopt the same conventions as before for indicating the (highest) type (s) of the arguments. Then :

---

(1) Kreisel's analogy is incorrectly stated. In our notation his analogue for a recursive set is

$$\lambda\alpha.\ \alpha \in [E] \quad \& \quad \{z\}(\alpha, E) = 0\ ,$$

where the p.r.f. $\{z\}(\alpha, E)$ need only be defined for hyperarithmetic $\alpha$. This arises naturally in the consideration of $\omega$-models, whose hard core consists of the hyperarithmetic functions.

However for our purpose the analogy described in the text is easier to handle, and is sufficiently suggestive.

3.21 $$D[\mathfrak{u}](0) = (\Sigma^1_1 \restriction [\mathfrak{u}])\ (\mathfrak{u} : 0)\ ,$$

and also :

3.22 $$D[\mathfrak{u}]\ (1 \restriction [\mathfrak{u}]) = (\Sigma^1_1 \restriction [\mathfrak{u}])\ (\mathfrak{u} : 1 \restriction [\mathfrak{u}])\ .$$

The class of predicates denoted by either side of this equation is the analogue, in our generalisation of Kreisel's analogy, of the class of recursively enumerable predicates.

## 4 - THE HIERARCHIES [E] AND [$E_1$]

We first give proofs of some of the more important properties of the hyperarithmetic hierarchy using 'recursive in E' as the definition of 'hyperarithmetic'. We do this to provide illustrations for the theorems and notations of the last two sections, and also to single out those theorems which form the basis of a partial analogy to be explored in the final section.

4.1 Primitive recursive in E = Arithmetic.

This is true both for predicates of numbers and of functions. The proof either way is a straightforward inductive one ; it can be found in SCK1.

4.2 $$[E]\ (1) \subseteq \Delta^1_1\ (1)$$

This is an immediate consequence of 2.1, 2.2 and 4.1

4.3 $$D[E]\ (1) \subseteq \pi^1_1(1)$$

This is an immediate consequence of 4.1 and 2.3.

4.4. <u>For any type 2 functional F and any function</u> $\alpha$,

$$\lambda x\,.\, D(x\ ;\ \alpha\ ,\ F) \ \underline{\text{is complete for}}\ \pi^1_1(\alpha\ :\ 0)\ .$$

For simplicity we shall omit all reference to $\alpha$ in our proof. Let a $\pi^1_1$ predicate $(\beta)\ (Ey)\ R(\bar{\beta}(y), a)$ be given, and let $S_a$ be the set of non-past-secured sequence numbers for $R(u, a)$. Now by the recursion theorem we can find an index number e which satisfies :

$$\{e\}\ (u\ ,\ a\ ,\ F) \simeq 0 \qquad \text{if } u \notin S_a\ ,$$

$$\simeq F(\lambda t\,.\,\{e\}\ (u^* 2^t\ ,\ a\ ,\ F) \qquad \text{if } u \in S_a$$

Recall that there is a primitive recursive substitution function $S^2$ satisfying

$$\{S^2(e\ ,\ 1\ ,\ a)\}\ (F) \simeq \{e\}\ (1\ ,\ a\ ,\ F)$$

I claim that $$D(S^2(e\,.\,1\,.\,a)\ ;\ F) \equiv (\beta)\ (Ey)\ R(\bar{\beta}(y)\ ,\ a)\ .$$

Consider the tree for the computation $\{e\}\ (1\ ,\ a\ ,\ F)$. Corresponding to each $u \in S_a$ there will be a branch point on the tree, and conversely. Further the computation at any point beyond which there are no branch points will surely be defined, since it must be a part of a computation according to the top clause in the definition of $\{e\}$. Thus the tree has infinite branches if and only if there are unsecured infinite sequences ; this proves the theorem.

4.5 $$D[E]\ (1) = \pi^1_1(1)\ .$$

This follows immediately from 4.3 and 4.4. Since $\pi^1_1(1 \restriction [E])$ is Kreisel's definition of the analogue

of 'recursively enumerable', we were justified in calling $D[\mathfrak{u}]$ $(1 \upharpoonright [\mathfrak{u}])$ the generalisation of his analogue.

4.61 $$D[E, \alpha](0) = (\Sigma^1_1 \upharpoonright [E, \alpha])(\alpha : 0)$$

This is a consequence of 3.21.

4.62 $$\pi^1_1(\alpha : 0) = (\Sigma^1_1 \upharpoonright [E, \alpha])(\alpha : 0) .$$

This follows from 4.5, 4.61. Evidently we may interchange '$\pi$' and '$\Sigma$'. Therefore

4.7 $$\Delta^1_1(\alpha : 0) = (\Delta^1_1 \upharpoonright [E, \alpha])(\alpha : 0) .$$

4.8 $$(\Delta^1_1 \upharpoonright [E, \alpha])(\alpha : 0) = [E, \alpha](0) .$$

This is an instance of 3.12 ; it shows that any class of functions which is generated by applying hyparithmetic operations to a finite set of functions provides a model for the $\Delta^1_1$ comprehension axiom with free function variables.

4.9 $$\Delta^1_1(1) = [E](1) .$$

This is an immediate consequence of 4.7 and 4.8.

We now turn to the hierarchy $[E_1]$. We can give some idea of the extent of this hierarchy as follows. Let R be a number-theoretic well-ordering relation, which is recursive in $E_1$. We can define what it means to apply the hyperjump operation $|R|$ times starting with the predicate $\lambda x.x=x$, where $|R|$ is the ordinal of R. Evidently the result of this process is a predicate which is recursive in $E_1$. (We do not here discuss the degree of invariance of this process for constant $|R|$, but varying R). Thus the extent of $[E_1]$ can be approciated by considering what ordinals are less than $\omega_1^{E_1}$. We define an *initial* ordinal to be one which is $\omega_1^A$ for some number-theoretic predicate A, or which is a limit point of such ordinals ; we denote by $\omega_\eta$ the $\eta$-th initial ordinal. If $\omega_\eta < \omega_1^{E_1}$, then so is $\omega_{\eta+1}$. Hence $\omega_\eta < \omega_1^{E_1}$, for $n \in N$ ; indeed these are just the ordinals of well-orderings which are primitive recursive in $E_1$. Further if we can construct recursively in $E_1$ a sequence of notations for an ascending sequence of initial ordinals each $< \omega_1^{E_1}$, then its limit point will be $< \omega_1^{E_1}$. Hence if $\eta < \omega_1^{E_1}$, then also $\omega_\eta < \omega_1^{E_1}$. So $\omega_\omega, \omega_{\omega_\omega}, \ldots$, are $< \omega_1^{E_1}$. And it is not hard to see that if we call any solution of the equation $\zeta = \omega_\zeta$ a $\zeta$-number, then if $\eta < \omega_1^{E_1}$, then so is the $\eta$-th $\zeta$-number. An so on ! We may remark that the ordinals defined in Addison-Kleene A-K are certainly all less than $\omega_1^{E_1}$ ; (indeed, they are probably all less than $\omega_2$). [For 'probably' read 'not'. W.H. Richter tells me (March '64) that he has proved that $\omega_2$ is the least ordinal not represented in the Kleene-Addison second number class. This suggests the conjecture that the Kleene-Addison ordinals are precisely those less than $\omega_1^{E_1}$.]

We now investigate the quantifier form of predicates in $[E_1]$. It is known that $[E_1] \subseteq (\Delta^1_2)$ (see TT and JRS). We give a new proof of this fact, based on ideas which will be used in the next section.

We denote by $\mathfrak{c}$ an arbitrary list of variables of types 0 and 1. We suppose that a standard system of indexing has been adopted for all predicates of $\Delta^1_2(1)$ so that the two formulae and the number of type 0 and of type 1 arguments can all be primitively recursively determined from the index. To distinguish between this index and that for a p.r.f. we shall if necessary refer to the former as a $\Delta$-index. We write $[\![n]\!](\mathfrak{c})$ for the predicate whose index is n. We shall also speak of the index of a function or functional, meaning thereby the index of its graph, and we shall write

$[\![n]\!]$ $(\mathfrak{c})$ for the value of the function. Evidently there are primitive recursive substitution functions (analogous to Kleene's S functions) which determine the index of the predicate which results from a given predicate by substituting particular numbers, or functions with particular index numbers, for some of the arguments. It makes perfectly good sense to speak of an index number for a predicate of no variables.

5.1. There is a primitive recursive function $\Phi(z, f)$ such that, if F is a functional with $\Delta$-index number f, then

$$\{z\}\ (\mathfrak{c}, F) = p \equiv [\![\Phi(z, f)]\!]\ (\mathfrak{c}, p)\ .$$

We first prove this when z is an index for a primitive recursion. The construction of $\Phi$ is by induction on the construction of z (considered as an index). We illustrate the definition of $\Phi$ by considering three cases.

S.7 : We must define $\Phi(z, f)$ so that

$$[\![\Phi(z, f)]\!]\ (x, \alpha, \mathfrak{c}, p) \equiv \alpha(x) = p\ ;$$

this is easily done.

S.4 : We require

$$[\![\Phi(z, f)]\!]\ (\mathfrak{c}, p) \equiv (Eq)\ ([\![\Phi(z_2, f)]\!]\ (\mathfrak{c}, q)\ \&\ [\![\Phi(z_1, f)]\!]\ (q, \mathfrak{c}, p))\ ;$$

We have merely to bring the LHS to $\Sigma^1_2$ and $\pi^1_2$ form, thus defining $\Phi(z, f)$ in terms of $\Phi(z_1, f)$, $\Phi(z_2 f)$.

S.8 : We require

$$\begin{aligned}[\![\Phi(z, f)]\!]\ (\mathfrak{c}, p) &\equiv (E\alpha)\ \{(x)\ (y)\ [\alpha(x) = y \equiv [\![\Phi(z_1, f)]\!]\ (x, \mathfrak{c}, y)] \\ &\qquad\qquad \&\ [\![f]\!]\ (\alpha, p)\} \\ &\equiv (\alpha)\ \ \{(x)\ (y)\ [\alpha(x) = y = [\![\Phi(z_1, f)]\!]\ (x, \mathfrak{c}, y)] \\ &\qquad\qquad \longrightarrow [\![f]\!]\ (\alpha, p)\}\ .\end{aligned}$$

The right hand sides can be brought to $\Sigma^1_2$ and $\pi^1_2$ form respectively, thus defining $\Phi(z, f)$ in terms of $\Phi(z_1, f)$. Evidently, in each case, the defining equations for $\Phi(z, f)$ can be expressed as a primitive recursion.

Thus we can obtain $\Delta$-indices for the primitive recursive predicates I, J, M and A which appear in 2.1, 2.2, 2.3, and 2.5, for the case in which we are interested (viz. r = 2 and F the only type 2 object in $\mathfrak{u}$ ). Then, using 2.1 and 2.3 for the $\pi^1_2$ form, 2.2 and 2.5 for the $\Sigma^1_2$ form, we obtain the required index, $\Phi(z, f)$ for the predicate :

$$\{z\}\ (\mathfrak{c}, F) \text{ is defined } \&\ \{z\}\ (\mathfrak{c}, F) = w\ . \qquad \text{Q.E.D.}$$

5.2. If $F \in \Delta^1_2$, then $[F]\ (1) \subseteq \Delta^1_2$ and $D[F]\ (1) \subseteq \Delta^1_2$.

For $$D(z\ ;\ \mathfrak{c}, F) \equiv (Ew)\ [\![\Phi(z, f)]\!]\ (\mathfrak{c}, w)\ .$$

Since $E_1 \in \Delta^1_2$, it follows that $[E_1]$ is included in and does not exhaust $\Delta^1_2$, as was to be shown.

We close this section with a rather interesting theorem about $[E_1]$.

5.3 $$(\Delta^1_2 \upharpoonright [E_1])\ (0) = [E_1]\ (0)\ .$$

For, let $A(\alpha, \beta)$ be an arithmetic formula whose only free function variables are $\alpha$ and $\beta$. It is well known (see e.g. SCK 2) that

$$(E\beta \upharpoonright [E_1, \alpha])\ A(\alpha, \beta) \equiv (E\beta)\ A(\alpha, \beta) .$$

The RHS is recursive in $E_1$ and ; hence

$$\alpha \in [E_1] \longrightarrow \{(E\beta \upharpoonright [E_1])\ A(\alpha, \beta) \equiv R(\alpha, E_1)\} ,$$

for suitable recursive R.

Therefore

$$(\pi^1_2 \upharpoonright [E_1])\ (0) = (\pi^1_1 \upharpoonright [E_1])\ (E_1 : 0) ,$$

and, since we can then replace $\pi$ by $\Sigma$, we have

$$(\Delta^1_2 \upharpoonright [E_1])\ (0) = (\Delta^1_1 \upharpoonright [E_1])\ (E_1 : 0) .$$

From this 5.4 follows by 3.12. Notice that in 5.4 we could replace $E_1$ by any list of parameters with types not greater than 2 in which $E_1$ is recursive.

As a matter of fact, $[E_1]$ is the least class C of functions and number-theoretic predicates which is closed with respect to the hyperjump and satisfies $(\Delta^1_2 \upharpoonright C)\ (0) = C$ ; this will be proved in ROG1. From this it follows that $[E_1]$ (1) provides a minimum $\beta$-model (i.e. a model which is standard with respect to well-orderings of the natural numbers - see AM1) for a system of second order arithmetic whose strongest comprehension axiom is that for $\Delta^1_2$ (with or without function parameters).

## 6 - THE SUPERJUMP OPERATION

In the introduction it was observed that the passage from recursive operations (on type 1 objects) to the jump operation E, and the passage from there to the hyperjump operation $E_1$, can both be thought of as the result of diagonalising on a class of functions recursive in a type 2 functional. They are both instances of what we shall call the *superjump* operation :

$$F \longrightarrow \lambda x \alpha .\ (\{x\}\ (\alpha, F) \text{ is defined}) .$$

Now there are good reasons for stopping at the jump operation ; but there seems no particular reason for admitting the hyperjump, but not the hyperhyperjump (i.e. the result of applying the superjump to the hyperjump) ; and so on.

It therefore seems natural to pass straight to the hierarchy of type 1 and type 2 objects which are recursive in the superjump. We proceed to an investigation of this hierarchy.

We need a type 3 functional which is of the same recursive degree as the superjump operation. Two candidates suggest themselves :

(a) $$\mathcal{O}(F) = 0 \text{ if } \{F(\lambda x .\ 0)\}\ (\lambda t .\ F(\delta_t), \lambda \alpha .\ F(\alpha^*)) \text{ is defined}$$

$$= 1 \text{ otherwise}$$

$$\text{where } \delta_t(x) = \begin{matrix} t & \text{if } x = 1 \\ 0 & \text{if } x \neq 1 \end{matrix} , \text{ and } \alpha^*(0) = \alpha(0) + 1 , \alpha^*(x) = \alpha(x) \text{ for } 0 < x .$$

(b) $$\mathcal{E}(F) = \begin{matrix} 0 \\ 1 \end{matrix} \text{ as } (E\alpha \upharpoonright [F])\ (F(\alpha) = 0)$$

$\mathcal{O}$ is obviously recursive in the superjump operation ; and since E is also, we see that

$$\lambda F\,.\,(Ez)\,[(x)\,D(z\,;\,x\,,\,F)\ \&\ F(\lambda x\,.\,\{z\}\,(x\,,\,F))=0]$$

is ; thus $\mathcal{S}$ is recursive in the superjump operation. Conversely that operation is recursive in $\mathcal{O}$ ; and using 3.2, it is not hard to show that it is also recursive in $\mathcal{S}^{(3)}$.

The generalisation of Kreisel's analogy which was discussed in section 3 suggests that if we raise the types of all variables by one, and replace E by $\mathcal{S}$, in the theorems of section 4, then new theorems should result. For a reason to be discussed, this is not always the case ; but the analogy is certainly more fruitful in suggesting theorems than it would be if we simply replaced $E(=E^2)$ by $E^3$, the representing function for $\lambda F\,.\,(E\alpha)\,(F(\alpha)=0)$. The reason why our analogy is imperfect may be exhibited as follows. Consider

$$(*)\qquad \mathcal{S}(\lambda\alpha\,.\,\Phi(\alpha\,,\,\mathcal{S}))\ ,$$

where $\Phi$ is partial recursive. According to Kleene's definition, this computation is only defined if $\Phi(\alpha\,,\,\mathcal{S})$ is defined for *all* functions $\alpha$ ; but the value of the computation depends only on the values of $\Phi$ for those functions which are recursive in $\lambda\alpha\,.\,\Phi(\alpha\,,\,\mathcal{S})$. Kleene's definition is thoroughly impredicative ; for example, using Mostowski's undecidable sentence (sec $AM^2$) one can construct a $\Phi$ recursive in $\in$ which will be defined for all $\alpha$ in some $\omega$-models (and indeed in some well-founded models) of set theory, but will not be defined for all $\alpha$ in any $\omega$-model of the theory obtained by ajoining the axiom of the existence of inaccessible ordinals. We cannot expect therefore that $\lambda z\,.\,D(z\,;\,\mathcal{S})$ should be capable of a definition 'from below' ; and so we cannot expect that the analogue of Spector's theorem (4.26) should hold. But whereas Kleene was concerned with recursive functionals having arbitrary type 3 arguments, we are here concerned only with the particular argument $\mathcal{S}$. It is therefore natural to seek a looser definition of 'is defined', which will apply whenever we can assign unambiguously a value to (*).

Let F be a *partial* functional of type 2, and let $\mathfrak{u}$ be a list of objects of types $\leq 2$ ; we say, naturally, that $\{z\}\,(\mathfrak{u}\,,\,F)$ is defined if it is defined in Kleene's sense, and whenever the computation

---

(3) To show that the superjump is recursive in $\mathcal{S}$, it will be sufficient to show that $(E\alpha\restriction[F])\,R(\alpha\,,\,F)$ is recursive in $F\,,\,\mathcal{S}$. Let G be defined by :

(a) if $\alpha(0)=0$, then $G(\alpha)=\begin{matrix}0\\1\end{matrix}$ as $\begin{matrix}R(\lambda t\,.\,\alpha(t+1)\,,\,F)\\ \text{not}\ ;\end{matrix}$ or

(b) if $\alpha(0)=x+1$, then $G(\alpha)=F(\lambda t\,.\,\alpha(t+1))+1$ .

Evidently $G\in[F]$ and $F\in[G]$, so that $[F]=[G]$.

Then

$$(E\alpha\restriction[F])\,R(\alpha\,,\,F)\equiv(E\alpha\restriction[F])\,(G(\alpha)=0)\ ,$$
$$\equiv \mathcal{S}(G)=0\ ,$$

which gives the required recursion.

requires an application of F (by S8), the value of F is defined for the argument in question. We say that F is *recursively satisfactory* if

$$(z)\,[(x)\,(\{z\}\,(x, \mathfrak{u}, F) \text{ is defined}) \longrightarrow F(\lambda x.\{z\}\,(x, F)) \text{ is defined}]\ .$$

Now we say that $\{z\}\,(\mathfrak{u}, \mathfrak{S})$ is *minimally defined*, and write $D_{min}(z : \mathfrak{u}, \mathfrak{S})$ if it satisfies the clauses of the inductive definition given by Kleene when z refers to a scheme other than S83, while for that schema the clause is :

$$\mathfrak{S}(\lambda\alpha.\{z_1\}\,(\alpha, \mathfrak{u}, \mathfrak{S})) \text{ is defined if and only if } \lambda\alpha.\{z_1\}\,(\alpha, \mathfrak{u}, \mathfrak{S})$$

is recursively satisfactory.

We write $D_{max}(z ; \mathfrak{u}, \mathfrak{S})$ for Kleene's definition ; evidently

$$D_{max}(z ; \mathfrak{u}, \mathfrak{S}) \longrightarrow D_{min}(z ; \mathfrak{u}, \mathfrak{S})\ .$$

But the converse implication is false. However :

6.1. <u>There is a partial recursive functional</u> $\chi(z, \mathfrak{u})$ <u>such that</u> :

$$D_{min}(z ; \mathfrak{u}, \mathfrak{S}) \longrightarrow [\chi(z, \mathfrak{u}) \text{ is defined } \&\ D_{max}(\chi(z, \mathfrak{u}) ; \mathfrak{u}, \mathfrak{S})$$

$$\&\ \{z\}\,(\mathfrak{u}, \mathfrak{S}) = \{\chi(z, \mathfrak{u})\}\,(\mathfrak{u}, \mathfrak{S})]\ .$$

The definition of $\chi$ is by induction up the computation tree ; the crucial point is that we can define a recursive relation $R(\alpha, F)$ which satisfies

$$(E\alpha \upharpoonright [F])\,(F(\alpha) = 0) \equiv (E\alpha \upharpoonright [F])\,R(\alpha, F)\ ,$$

and which is defined for *all* $\alpha$ whenever F is recursively satisfactory. We shall not give the proof here. 6.1 shows that the extent of the class $[\mathfrak{S}](2)$ does not depend on which definition of 'is defined' we adopt.

Before investigating the analogues of the theorems of section 4, we shall examine the relation between $[\mathfrak{S}]$ and $\Delta^1_2$ ; 5.2 leads us to expect that $[\mathfrak{S}] \subseteq \Delta^1_2$, and we shall show that the inclusion is proper.

6.2. <u>Let</u> $\mathfrak{c}$ <u>be a list of objects of types</u> $\leq 1$. <u>There is a primitive recursive function</u> $\psi$ <u>such that</u>

$$D_{max}(z ; \mathfrak{c}, \mathfrak{S}) \longrightarrow [\{z\}\,(\mathfrak{c}, \mathfrak{S}) = p \equiv [\![\psi(z)]\!]\,(\mathfrak{c}, p)]\ .$$

$\psi(z)$ will be defined by means of the recursion theorem ; let e be an index for $\psi$. If z refers to S1-3, S7, then $\psi(z)$ is determined outright. If z refers to S4, we write :

$$\{z\}\,(\mathfrak{c}, \mathfrak{S}) = p \equiv (Eq)\,(Ex)\,(Ey)\,[T(e, z_1, x)\ \&\ T(e, z_2, y)\ \&\ [\![U(y)]\!]\,(\mathfrak{c}, q)$$
$$\&\ [\![U(x)]\!]\,(q, \mathfrak{c}, p)]$$
$$\equiv [\![\sigma(z, e)]\!]\,(\mathfrak{c}, p)\ ,$$

where the primitive recursive function $\sigma$ is determined by the manipulations necessary to bring the RHS to $\Delta^1_2$ form. So in this case $\psi(z) = \sigma(z, e)$.

If z refers to S9, we write

$$\{z\}\,(a, \mathfrak{c}, \mathfrak{b}, \mathfrak{S}) = p \equiv (Ex)\,[T(e, a, x)\ \&\ [\![U(x)]\!]\,(\mathfrak{c}, p)]$$
$$\equiv [\![\tau(e)]\!]\,(\mathfrak{c}, p)\ ,$$

say, with an easily determined primitive recursive $\tau$ ; in this case $\psi(z) = \tau(e)$, and so is determined outright - a fact which explains the possibility of giving a *primitive* recursive definition for $\psi$ .

Towards S8 : let F be the functional whose graph is $\lambda\alpha q . [\![f]\!] (\alpha , \mathfrak{c} , q)$ ; so that F depends on the parameters $\mathfrak{c}$ . Then there is a primitive recursive function $\varepsilon(f)$ such that

$$[\![\varepsilon(f)]\!] (\mathfrak{c} , p) \equiv p = \mathfrak{F}(F) .$$

The proof of this is very similar to the proof of 5.1, and is left to the reader. Now, when z refers to S8, we write

$$\{z\} (\mathfrak{c} , \mathfrak{F}) = p \equiv (Ex) [T(e , z_1 , x) \ \& \ [\![\varepsilon(U(x))]\!] (\mathfrak{c} , p)]$$
$$\equiv [\![\rho(e , z)]\!] (\mathfrak{c} , p) ,$$

say, with primitive recursive $\rho$, and set $\psi(z) = \rho(e , z)$.

Thus $\psi(z)$ is determined primitive recursively from z and e, and so $\psi$ itself is primitive recursive. Q.E.D.

6.3 $$\lambda \mathfrak{c} z . D_{\min}(z ; \mathfrak{c} , \mathfrak{F}) \in \Delta^1_2(1) .$$

We shall not give the proof. The crucial point is that in deciding if $D_{\min}(z ; \mathfrak{c} , \mathfrak{F})$ holds, one only needs to consider a computation tree with countably many branches at each branch point ; further the local correctness of a proposed evaluative function $\eta$ at a branch point is described by a $\Delta^1_2$ condition. Thus one can imitate the analysis described in section 2 for the case r = 2, using only $\Delta^1_2$ formulae.

6.4 $$[\mathfrak{F}](1) \subset \Delta^1_2(1) .$$

The inclusion is an immediate consequence of 6.2, while 6.3 shows that it must be proper. For if $\lambda \mathfrak{c} z . D_{\min}(z ; \mathfrak{c} , \mathfrak{F}) \in [\mathfrak{F}]$, we should have a contradiction by the diagonal argument.

6.5 $$\lambda \mathfrak{c} z . D_{\max}(z ; \mathfrak{c} , \mathfrak{F}) \in \pi^1_2(1) .$$

Using remark 5 in section 2 and 6.2, it is easy to prove the following Lemma : *There is a number* B, *such that for all putative positions* $\gamma$ *which lie below a lowest non-terminating branch of the computation* $\{z\} (\mathfrak{c} , \mathfrak{F})$ ,

$$[\![B]\!] (\mathfrak{c} , \gamma , z , b) \equiv [(b = 0 \text{ and } \gamma \text{ is not a position}) \vee$$
$$(b = \beta(\gamma))] .$$

In particular, if the computation is defined, then the above predicate determines the computation description number b correctly at every position.

Using the above predicate we obtain a $\pi^1_2$ formula for $D_{\max}$ through the equivalence :

$$D_{\max}(z ; \mathfrak{c} , \mathfrak{F}) \equiv (\rho) [(m) (\rho_m \text{ is a position}) \longrightarrow (En) (\rho_n \text{ is a tip})] ,$$

where $\rho_m$ is the $m^{th}$ member of the infinite sequence $\rho$ of putative positions. This proves 6.5.

6.6 $\lambda \mathfrak{c} z . D_{\max}(z ; \mathfrak{c} , \mathfrak{F})$ *is complete for* $\pi^1_2(1)$

Since $(E\beta) A(\alpha , \beta , \mathfrak{c})$ is recursive in $E_1$, it is also recursive in $\mathfrak{F}$, and we can therefore define a primitive recursive function $\Phi(a)$, (where a is the Godel number of the arithmetic predicate A), such that :

$$\{\Phi(a)\}(\alpha, \mathfrak{c}, \mathcal{E}) \simeq 0 \quad \text{if } (E\beta)\, A(\alpha, \beta, \mathfrak{c}),$$
$$\simeq \{\Phi(a)\}(\alpha, \mathfrak{c}, \mathcal{E}) \text{ otherwise.}$$

Then evidently the computation

$$\mathcal{E}(\lambda\alpha.\, \{\Phi(a)\}(\alpha, \mathfrak{c}, \mathcal{E}))$$

is defined (maximally) if and only if $(\alpha)(E\beta)\, A(\alpha, \beta, \mathfrak{c})$. Q.E.D.

In particular, therefore, there is a $\Delta^1_3$ index for the type 2 functional $D^2_1$ which represents $\lambda\alpha.\, D_{max}(\alpha(0);\ \lambda t.\, \alpha(t+1), \mathcal{E})$. Further $\pi^1_2(1) \subseteq [D^2_1]$. Now we can repeat the proofs of 6.2-6.6, using $D^2_1$, $\mathcal{E}$, in place of $\mathcal{E}$, and $\pi^1_3$ in place of $\pi^1_2$, etc. And this process can then be repeated. In general, let $D^2_0 = \lambda\alpha.\, 0$, and let

$$D^2_{n+1}(\alpha) = \begin{matrix} 0 \\ 1 \end{matrix} \quad \text{as} \quad D_{max}(\alpha(0);\ \lambda t.\, \alpha(t+1), D^2_n, \mathcal{E}) \text{ or not}.$$

Then, for $n \geq 0$,

6.7 $$[D^2_n, \mathcal{E}] \subseteq \Delta^1_{n+2}$$

6.8 $$D_{max}[D^2_n, \mathcal{E}](1) = \pi^1_{n+2}(1).$$

These two theorems were announced in ROG3 [4].

## 7 - THE ANALOGY BETWEEN E AND $\mathcal{E}$

In this section we work out the details of the analogy between E and $\mathcal{E}$ which was discussed at the beginning of the last section. In particular, we seek to prove the analogues of theorems 4.2-4.7. One might suppose that the analogue of $\pi^1_1$ would be the set C of all predicates :

$$A(u) \equiv (F^2)(E\alpha \quad [F])\, R(u, \alpha, F) \qquad (\text{types of } u \leq 2).$$

---

(4) While revising this paper I have noticed that the results of section 6 can be extended to higher types ; the only important modification is that the two-function-quantifier forms are replaced by forms with a single quantifier of appropriate type.

Let $0^{n+1} = \lambda\tau^n.\, 0$; let $\tau^n \longrightarrow f(\tau^n)$ be a recursive one-to-one mapping of type n into (type $n - \{0^n\}$). Let $\alpha^{n+1}_+$ be defined recursively from $\alpha^{n+1}$ so as to satisfy $\alpha^{n+1}_+(\tau^n) = \alpha^{n+1}(f(\tau^n))$. We define a generalised superjump operator $\delta^{n+3}$ by :

$$\delta^{n+3}(\alpha^{n+2}) = 0 \quad \text{if} \quad D_{max}(\alpha^{n+2}(0);\ \alpha^{n+2}_+)$$
$$= 1 \text{ otherwise}$$

And we define $\gamma^{n+2}_0 = 0^{n+2}$,

$$\gamma^{n+2}_{m+1}(\alpha^{n+1}) = 0 \quad \text{if} \quad D_{max}(\alpha^{n+1}(0);\ \alpha^{n+1}_+,\ \gamma^{n+2}_m,\ \delta^{n+3})$$
$$= 1 \text{ otherwise}$$

Then, for $n \geq 1$,

$$[\delta^{n+3}](n+1) \subset \Delta^{n+1}_1(n+1)$$

$$D_{max}[\delta^{n+3}](n+1) = \pi^{n+1}_1(n+1).$$

$$[\gamma^{n+2}_m, \delta^{n+3}](n+1) \subseteq \Delta^{n+1}_{m+1}(n+1).$$

$$D_{max}[\gamma^{n+2}_m, \delta^{n+3}](n+1) = \pi^{n+1}_{m+1}(n+1).$$

The results of section 7 can also be extended ; in particular

$$\pi^{n+1}_1(n+1) = \pi^{(n+2)^*}_1(n+1);$$

$$D_{max}[\delta^{n+3}](n+2) = \pi^{(n+2)^*}_1(n+2).$$

However this will not do[5]. Using 3.2 we readily find a z such that

$$(E\alpha \upharpoonright [F])\ R(\alpha, \mathfrak{u}, F) \equiv \{z\}(\mathfrak{u}, F, E) = 0 \ ;$$
$$\equiv (\eta)\ P(\mathfrak{u}, \eta, F, E) \ ,$$

with a primitive recursive P, by 2.1 and 2.3. Therefore, on combining quantifiers,

$$A(\mathfrak{u}) \equiv (F)\ Q(\mathfrak{u}, F, E) \ ;$$
$$\equiv (\alpha)\ S(\mathfrak{u}, \alpha, E) \ ,$$

with primitive recursive S by Kleene's theorem (SCK1 XXXIV) on the reduction of the types of quantifiers. Thus if the types of $\mathfrak{u}$ are $\leq 1$, then $C \subseteq \pi^1_1$, while the analogue of 4.2 would lead us to expect $[\mathfrak{E}] \subseteq C$ ; these two are obviously incompatible.

The reason for this failure is that the manipulations by which one arrives at a *single* numerical quantifier in the normal form for a $\pi^1_1$ predicate cannot all be applied to restricted function quantifiers.

The correct analogue of $\pi^1_1$ is the set (which we shall call $\pi^{2*}_1$) of all predicates :

$$(F)\ (Q_i\ \alpha_i \upharpoonright [F, \mathfrak{u}])\ R(\mathfrak{u}, \alpha_1, \ldots, \alpha_n, F) \qquad (\text{types of } \mathfrak{u} \leq 2) \ ,$$

where the $Q_i (1 < i < n)$ are a string of alternating quantifiers ; note that in the term $[F, \mathfrak{u}]$ which restricts those quantifiers, only the non-numerical members of $\mathfrak{u}$ are relevant.

It is in fact permissible to permute a restricted function quantifier and an *unrestricted* functional quantifier ; this will be shown in 7.3. But the reduction to n = 1 in the above form would require the permutation of a restricted function quantifier with *restricted* functional quantifier of the opposite kind. We shall see that $\pi^{2*}_1 \neq C$ ; it follows that the second sort of permutation is not generally permissible. We first prove two lemmas.

7.1. <u>There is a primitive recursive</u> P <u>such that</u> :

$$(\alpha \upharpoonright [H])\ A(\alpha) \equiv (\alpha \upharpoonright [G, H, E])\ \{(E\beta \upharpoonright [G, H, E])\ P(\alpha, \beta, G, H, E) \longrightarrow A(\alpha)\}$$

For, $\alpha \in [H] \equiv$

$$(Ez)\ [(t)\ (\{z\}(t, G, H, E) = \alpha(t)] \ \&\ (t)\ (E\eta \upharpoonright [G, H, E])$$

($\eta$ is locally correct for $\{z\}(t, G, H, E)$ & each S.8 point of this computation is an application of H)]. The phrase 'each S.8 point ...' can be expressed by a predicate which is primitive recursive in $\eta, t, z$ ; by 3.2, $\{z\}(b, C, H, E) = \alpha(t)$ can be expressed using an existential function quantifier restricted to $[G, H, E]$ ; and then 3.11 allows the function quantifiers to be pulled to the front, leaving the numerical quantifiers to be expressed as recursions in E. This completes the proof.

Let $\pi^{2+}_1$ be defined by replacing '$\upharpoonright [F, \mathfrak{u}]$' in the definition of $\pi^{2*}_1$ by '$\upharpoonright [F, \mathfrak{u}, E]$' and allowing E as an argument of R. Repeated application of 7.1 shows that $\pi^{2*}_1 \equiv \pi^{2+}_1$. But the converse is also true ; for

$$(F)\ (Q_i\ \alpha_i \upharpoonright [F, \mathfrak{u}, E])\ R(\mathfrak{u}, \alpha_1, \ldots, \alpha_n, F, E) \equiv$$
$$(F)\ \{(\beta \upharpoonright [F, \mathfrak{u}]) \left[ (F)_0(\beta) = \begin{matrix} 0 & \text{if } (Et)\ (\beta(t) = 0) \\ 1 & \text{if } (t)\ (\beta(t) \neq 0) \end{matrix} \right] \longrightarrow$$
$$(Q_i\ \alpha_i \upharpoonright [F, \mathfrak{u}])\ R(\mathfrak{u}, \alpha_1, \ldots, \alpha_n, (F)_1, (F)_0)\} \ .$$

---

(5) At the time of giving this paper I wrongly supposed that it would do. The definition of $\pi^{2*}_1$ and theorems 7.1 - 7.3 are therefore later additions.

The implication from right to left is proved by substituting E for $(F)_0$. And if F satisfies the condition on the right, then by transfinite induction on the ordinal of the computation for each $\alpha$ in $[F, \mathfrak{u}]$ we see that $[F, \mathfrak{u}](1) = [(F)_1, \mathfrak{u}, E](1)$ ; thus $(F)_0$ must agree with E for all relevant arguments, from which the implication from left to right follows. Finally, the numerical quantifiers on the right hand side can be replaced by restricted function quantifiers.

This proves

7.2 $$\pi_1^{2+}(2) = \pi_1^{2*}(2) .$$

7.3. *Let* $A(\mathfrak{u})$ *be a predicate of arguments of type* $\leq 2$ *which has a prenex normal form which satisfies* : (a) *all type* 2 *quantifiers are universal and unrestricted* ; (b) *each type* 1 *quantifier is restricted to functions recursive in* $\mathfrak{u}$ *and those type* 2 *variables whose quantifiers precede it* ; (c) *there are no quantifiers of types* $> 2$. *Then* $A(\mathfrak{u}) \in \pi_1^{2*}$.

For, classically, $(E\alpha \upharpoonright [\mathfrak{b}])(G) B(\alpha, G) \equiv (G)(E\alpha \upharpoonright [\mathfrak{b}]) B(\alpha, \lambda\beta . G(<\alpha\beta>))$ ; where $<\alpha\beta>$ is, say, $\lambda t . 2^{\alpha(t)} 3^{\beta(t)}$). Now we can apply 7.1 to alter the restriction from $\upharpoonright [\mathfrak{b}]$ to $\upharpoonright [\mathfrak{b}, G, E]$. In this way all type 2 quantifiers can be pulled to the front, and the result will be a predicate of $\pi_1^{2+}$ and hence also of $\pi_1^{2*}$.

7.4 $$[\mathfrak{S}](2) \subseteq \Delta_1^{2*}(2) .$$

This is the analogue of 4.2. First, by induction on the index x, we show that the graphs of primitive recursions in $\mathfrak{S}$ belong to $\Delta_1^{2*}$. We shall consider only the case that z refers to an application of $S8^3$. By the definition of $\mathfrak{S}$, we have

$\{z\}(\mathfrak{u}, \mathfrak{S}) = w \equiv$

$(F)\{(\alpha \upharpoonright [F])(v)(F(\alpha) = v \equiv \{z_1\}(\alpha, \mathfrak{u}, \mathfrak{S}) = v) \longrightarrow$

$(w = 0 \ \& \ (E\alpha \upharpoonright [F])(F(\alpha) = 0)) \ v \ (w = 1 \ \& \ (\alpha \upharpoonright [F])(F(\alpha) \neq 0))\}.$

By 7.1 we can obtain an equivalent formula in which the function quantifiers are restricted to $[\mathfrak{u}, F]$. By induction hypothesis the graph of $\{z_1\}(\alpha, \mathfrak{u}, \mathfrak{S})$ belongs to $\Delta_1^{2*}$, (and we do not need to exhibit $\alpha$ in the restrictions, since it is itself restricted). So, by 7.3 we obtain the required $\pi_1^{2*}$ form. Similarly we can obtain a $\Sigma_1^{2*}$ form.

Now for general recursions in $\mathfrak{S}$ we apply 2.1, 2.2. Examination of their proofs for the case in hand shows that the inner (type 1) quantifier can be restricted as in condition (b) of 7.3. Then a further application of 7.3 and its dual completes the proof of 7.4.

7.5 $$D[\mathfrak{S}](2) \subseteq \pi_1^{2*}(2) .$$

This is the analogue of 4.3. Examination of the proof of 2.3 shows that here, also, the inner quantifier can be restricted ; so that the argument of the previous paragraph applies. However, the inner quantifier in 2.4 cannot be restricted, since its range must include, if possible, a branch along which the computation does not terminate. So it is not to be expected that $\Sigma$ could replace $\pi$ in 7.5. We shall soon see that the inclusion in 7.5 is, in fact, an equality.

Consider a formula of $\pi_1^{2*}$. The scope of the initial quantifier (F) contains only objects and operations which are recursive in F and the arguments $\mathfrak{u}$. The truth of the formula therefore only depends on the values of F for a denumerable set of arguments. Hence we may expect that the type 2 quantifier can be replaced by a type 1 quantifier. That this is indeed the case is shown by :

7.6 $$\pi_1^{2*}(2) \subseteq \pi_1^1(E_1 : 2) .$$

Let $\mathfrak{u}$ be a finite sequence of objects of types 1 and 2 ; this will remain fixed throughout the proof. Let a be a number which encodes a multiplet $a_0, \ldots, a_n$ of numbers, and also indicates a selection $\mathfrak{b}$ - perhaps arranged in a different order - from the sequence $\mathfrak{u}$. We write $\{z\}(a : \mathfrak{u})$ to denote $\{z\}(a_0, \ldots, a_n, \mathfrak{u})$.

Let a functional G be given ; we say that the function $\gamma$ *represents* G (with respect to recursions in $\mathfrak{u}$, G) if :

(i) $\{z\}(a : \mathfrak{u}, G)$ is not defined $\longrightarrow \gamma(2^z 3^a) = 0$ ;

(ii) $\{z\}(a : \mathfrak{u}, G) = w \longrightarrow \gamma(2^z 3^a) = w + 1$ .

Evidently there is an arithmetic formula $B(\mathfrak{u}_0, \mathfrak{u}, \gamma)$ (where $\mathfrak{u}_0$ is a string of numerical arguments), such that if $\gamma$ represents G, then

$$B(\mathfrak{u}_0, \mathfrak{u}, \gamma) \equiv (Q_1 \alpha_1 \upharpoonright [\mathfrak{u}, G])\, R(\mathfrak{u}_0, \mathfrak{u}, \alpha_1, \ldots, \alpha_n, G) .$$

Now we analyse the predicate '$\gamma$ represents some functional' ; the analysis will be by an induction on computations.

(a) If z refers to schemata S1-3, S4, S7, S9, then there are clauses which relate $\gamma(2^z 3^a)$ to certain other values of $\gamma$. These clauses are all recursive in $\gamma$.

(b) If z refers to an application of S 8 in which, say, a functional F (from the list $\mathfrak{u}$) is applied, then the clause is :

$$(t)\ (\gamma(2^{z_1} 3^{t*a}) = 0 \longrightarrow \gamma(2^z 3^a) = F(\lambda t .\ \gamma(2^{z_1} 3^{t*a}) \dot{-} 1) + 1) ,$$

where t* a encodes the result of prefixing t to the list of numerical arguments encoded by a. Here z and a determine which member of $\mathfrak{u}$ is to be taken as the above-mentioned F. Thus this clause is recursive in $\gamma$, $\mathfrak{u}$, and E.

(c) If z refers to an application of G by S8, then the value of $\gamma$ can be freely chosen subject to a consistency condition :

$$(y)\ (y \text{ refers to an application of G } \&\ (t)\ (\gamma(2^{z_1} 3^{t*a}) = \gamma(2^{y_1} 3^{t*a})) : \longrightarrow \gamma(2^z 3^a) = \gamma(2^y 3^a)) .$$

(d) If $\gamma$ satisfies the above clauses for all z, a, then it will determine the value of $\{z\}(a : \gamma, G)$ for some particular G whenever that computation is defined. Hence, as in remark 5 of section 2, we can determine recursively in z, a, $\gamma$, the computation description at any point on the tree for $\{z\}(a : \mathfrak{u}, G)$ which lies on or below a lowest non-terminating branch. Thus :

$$(z)\ (a)\ [\gamma(2^z 3^a) = 0 \equiv (E\rho)\ (\rho \text{ is a branch } \&\ (n)\ (\rho_n \text{ is not a tip on the tree for } \{z\}(a : \mathfrak{u}, G)))]$$

can be expressed recursively in $\gamma$, $E_1$. And this clause does ensure that $\gamma$ vanishes for all undefined computations. Thus '$\gamma$ represents some G with respect to recursions in $\mathfrak{u}$, G' can be expressed by a recursive predicate $Q(\gamma, \mathfrak{u}, E_1)$. And then the typical predicate of $\pi_1^{2*}$ is equivalent to

$$(\gamma)\ [Q(\gamma, \mathfrak{u}, E_1) \longrightarrow B(\mathfrak{u}_0, \mathfrak{u}, \gamma)]$$

This completes the proof of 7.6.

7.7 $$D_{max}(z ; \mathfrak{u}, \mathfrak{S}) \text{ is complete for } \pi_1^1(E_1 : 2) .$$

For, since $E_1 \in [\mathfrak{S}]$, given any recursive $R(\alpha, \mathfrak{u}, E_1)$ we can find $z_1$ so that :

$$\{z_1\}(\alpha, u, \mathcal{E}) \simeq 0 \quad \text{if } R(\alpha, u, E_1),$$
$$\simeq \{z_1\}(\alpha, u, \mathcal{E}) \text{ if not.}$$

Then $(\lambda\alpha.\{z_1\}(\alpha, u, \mathcal{E}))$ is maximally defined if and only if $(\alpha)\, R(\alpha, u, E_1)$. Q.E.D.

This theorem gives an alternative proof of 6.6.

7.8 $$D_{max}[\mathcal{E}](2) = \pi_1^1(E_1 : 2) = \pi_1^{2*}(2) .$$

For 7.7 shows that $\pi_1^1(E_1 : 2) \subseteq D_{max}[\mathcal{E}](2)$ : and together with 7.5 and 7.6 this gives a circle of inclusions. This theorem gives the expected analogue of 4.5. Actually we could add '$= \pi_2^1(2)$' to 7.8 ; but this has only been proved (in 6.6) when the types of the arguments are $\leqslant 1$.

It was pointed out in section 6 that we cannot expect to obtain the analogue of 4.6. $D_{max}[\mathcal{E}]$ is certainly not coextensive with $(\Sigma_1^{2*}\restriction[\mathcal{E}])$. On the other hand, I believe that $D_{min}[\mathcal{E}]$ is coextensive with it. In seeking analogues to further theorems about the hyperarithmetic hierarchy, one must therefore distinguish between those occurrences (explicit of implicit) of $\pi_1^1$ which depend on 4.62., and those which depend on 4.5. The analogue for the former will be $D_{min}[\mathcal{E}]$ ; for the latter $D_{max}[\mathcal{E}]$ or, equivalently, $\pi_1^{2*}$.

## REFERENCES


A-K J.W. ADDISON and S.C. KLEENE - *A note on function quantification*, Proc. Amer. Math. Soc. 8 (1957) pp. 1 002-1 006.

KG K. GÖDEL - *Über eine bisher noch nicht benutzte Erweiterung des finiten Standpunktes*, Dialectica 12 (1958) pp. 210-287.

ROG1 R.O. GANDY - *Selection operators for recursive functionals*, in preparation.

ROG2 R.O. GANDY - *Proof of Mostowski's conjecture*, Bull. Acad. Pol. Sci. 8 (1960) pp. 571-574.

ROG3 R.O. GANDY - *The Analytic hierarchy and recursive functionals*, (Abstract), Monthly notices Amer. Math. Soc. June 1962.

GK1 G. KREISEL - *La Prédicativité*, Bull. Soc. Math. France 88 (1960) pp. 371-391.

GK2 G. KREISEL - *Set theoretic problems suggested by the notion of potential totality*, Infinitistic Methods, Warsaw 1961 pp. 103-140.

GK3 G. KREISEL - *Model theoretic invariants : applications to recursive and hyperarithmetic operations*, Proc. Theory of Models, Symposium held at Berkeley, July 1963.

SCK S.C. KLEENE - *Recursive functionals and Quantifiers of finite types* I, Trans. Amer. Math. Soc. 91 (1959) pp. 1-52.

AM1 A. MOSTOWSKI - *Formal systems of analysis based on an infinitistic rule of proof*, Infinitistic Methods, Warsaw 1961, pp. 141-166.

AM2 A. MOSTOWSKI - *An undecidable arithmetical statement*, Fund. Math. 36 (1949) pp. 143-164.

CS C. SPECTOR - *Hyperarithmetical quantifiers*, Fund. Math. 48 (1961) pp. 113-120.

JRS J.R. SHOENFIELD - *The form of the negation of a predicate*, Recursive Function Theory Proc. Symp. Pure. Math. 5 (1962) pp. 131-134.

TT T. TUGUÉ - *Predicates recursive in a type 2 object and Kleene hierarchies*, Comment. Math. Univ. St. Paul 8 (1959) pp. 97-117.