

UNCLASSIFIED

AD 4181 02

DEFENSE DOCUMENTATION CENTER

FOR

SCIENTIFIC AND TECHNICAL INFORMATION

CAMERON STATION, ALEXANDRIA, VIRGINIA



UNCLASSIFIED

NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

64-3

418102

CATALOGED BY DDC
AS AD No. _____

418102

THE THEORY OF TRANSFINITE EFFECTIVITY

by

AZRIEL LÉVY

Applied Logic Branch

The Hebrew University of Jerusalem

Technical Report No. 12

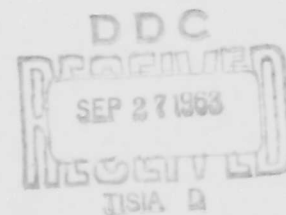
This report was prepared for the

U. S. OFFICE OF NAVAL RESEARCH, INFORMATION SYSTEMS BRANCH

under contract No. 62538-3510

Jerusalem, Israeli

July 1963



THE THEORY OF TRANSFINITE EFFECTIVITY

by

AZRIEL LÉVY

Applied Logic Branch

The Hebrew University of Jerusalem

Technical Report No. 12

This report was prepared for the

U. S. OFFICE OF NAVAL RESEARCH, INFORMATION SYSTEMS BRANCH

under contract No. 62558-3510

Jerusalem, Israel

July 1963

TABLE OF CONTENTS

1. Introduction	1
2. Recursive functionals	9
3. Computable functionals	15
4. The computability of recursive functionals for a regular Ω	22
5. The cardinality of the values of recursive functionals	64
6. Arithmetization of the notion of relative constructibility	68
7. Recursive functionals in complete models of set theory	80
8. The recursivity of the computable functionals for a regular Ω	83
9. The normal form, enumeration and hierarchy theorems for a regular Ω	100
10. The equivalence of recursivity and computability for a singular Ω	105
11. The equivalence of recursivity and Machover's notion of recursivity	115
Bibliography	122

THE THEORY OF TRANSFINITE EFFECTIVITY

by

Azriel Lévy

1. INTRODUCTION

Throughout this paper Ω will be a fixed infinite cardinal. Whenever we shall mention numbers or ordinal numbers we shall mean, unless otherwise stated or implied, ordinals $< \Omega$. Every ordinal is the set of all smaller ordinals. Whenever we shall mention functions or partial functions we shall mean, unless otherwise stated, functions or partial functions of n variables, $0 \leq n < \omega$, defined on Ω with values in Ω . Whenever we shall mention functionals we shall mean partial functions with values in Ω which depend on number variables (which vary over Ω) and function variables which vary over the set of functions.

Takeuti defined in [13] the notion of a recursive functional (recursive function in his terminology) of ordinal numbers. He imitated in the realm of the ordinal numbers that definition of the recursive functions of natural numbers which starts with the primitive recursive functions and extends them to the general recursive functions by adding the minimum operator¹. The reason why he considered recursive functionals rather than recursive functions was mainly not in order to achieve more generality but that in the transfinite case (i.e., for $\Omega > \omega$) this method of defining recursive functions must go via the definition of recursive functionals. In the finite case (i.e., where $\Omega = \omega$)

¹In [14] Takeuti and Kino define explicitly a notion of primitive recursive functions of ordinal numbers.

we have a recursion schema which makes the value of $f(x)$ depend on $f(x-1)$. The stronger recursion schema, which makes the value of $f(x)$ depend on the values of all $f(y)$'s, for $y < x$, can be derived from the schemata defining the primitive recursive functions, because the restriction of the function f to natural numbers less than x admits a simple representation by a single natural number. In the transfinite case not every number has a predecessor and therefore the only recursion schema available is the stronger recursion schema. In this case there is also no simple way of representing the restriction of a function f to a number x by a single number (and, unless we assume Gödel's axiom of constructibility, no constructive way of representing such a sequence is known at all); therefore the restriction of f to x has to enter the recursion schema as a function rather than a number.

Machover defined in [6], independently of Takeuti, another notion of recursive functions (and essentially also of recursive functionals) of ordinal numbers. He imitated the Gödel-Herbrand definition of the recursive functions of natural numbers, which uses a calculus of equations as given in [4].

The present work has a twofold aim. First we shall give a third approach to the notion of recursive functionals which will consist of an imitation of the notion of Turing computability. Second, we shall unify the three approaches by proving their equivalence (with some exceptions as will be

noted later).

The point of departure for defining the notion of a machine which computes recursive functions of ordinal numbers is Takeuti's or Machover's notion of a recursive function (where, unlike Machover we allow only a finite number of variables). Suppose we assign a clerk, with transfinite amount of time at his disposal (i.e., up to, but not including, Ω units of time) to actually compute the value of a given recursive function for a given number. He may have to write some numbers during his computation. The sequence of symbols needed to represent a number may be transfinite, so we shall allow him a transfinite amount of paper (i.e., up to, but not including, Ω finite sheets, which he can join to get a transfinite sheet) for the computation. By an analysis of Takeuti's or Machover's notion of a recursive function we get the surprising result that the clerk can do the computation in such a way that during each instant of the computation he will have only a finite number of ordinal numbers written on the sheet on which he computes. This fact enables the clerk to do the computation on a transfinite tape (of length Ω) in a relatively simple fashion. If the clerk would have infinitely many numbers on the tape then when during the computation he had to observe some number which he wrote down earlier he would have had trouble locating it.

Before our clerk starts computing the value of the function he is given instructions how to compute, which are written on a separate sheet, and the value of the argument, which is written on the sheet on which

he computes. We come now to consider the size of the instructions sheet. This is where a major difference from what we have in the finite case turns up. We agree first that the alphabet in which the instructions are written consists only of a finite number of symbols. For the sake of simplicity we discussed till now only computations of values of recursive functions, but similar remarks apply also to the computation of values of recursive functionals. Those functionals for the computations of which we can give instructions which can be written on a finite sheet will be called finitarily recursive functionals. Note that we do not demand that the computation should be done in finite time or that the clerk shall use only a finite amount of paper. Since from a finite alphabet we can make up at most $\aleph_0$ finite instructions, if $\aleph > \omega$ not even all the constant functionals are finitarily recursive. To remedy this we shall be a little more liberal in our allotment of paper for the instructions and in addition to the finite set of instructions we shall allow the instruction sheet to contain a finite list of ordinal numbers. The functionals we get in this way are called numerically recursive functionals. Finally we consider instruction sheets which contain an arbitrary set of instructions, as long as all the instructions together contain $< \aleph$ (distinct occurrences of) symbols. The functionals obtained this way are called general recursive functionals. If we would lift the restriction that the instruction sheet shall contain $< \aleph$ symbols then every function would become recursive, which would trivialize this notion. It will be shown that if the set of the instructions on the sheet is constructible in

the sense of Gödel [1] then the instructions can be summarized by a single ordinal number and a finite set of finite instructions and therefore the corresponding functional is numerically recursive. As a consequence we get that the notions of a general recursive functional coincides with the notion of a numerically recursive functional if Gödel's axiom of constructibility is assumed.

As was mentioned earlier a special rôle is played here by restrictions of functions to numbers or, what is equivalent, by sequences of numbers of length $< \aleph_1$. In the finite case these sequences can be easily represented by numbers, which is not the case in the transfinite case. For reasons that will become clear later we have to restrict our attention only to such bounded sequences, i.e., to sequences whose supremum is $< \aleph_1$ (this requirement is automatically fulfilled by each such sequence if $\aleph_1$ is a regular cardinal). We choose to represent these sequences by functions which are equal to 0 from some number on and we shall call them bounded eventually vanishing functions (b.e.v.f.'s). As the counterparts of the b.e.v.f.'s in the finite case are just the numbers we shall see that the b.e.v.f.'s behave like numbers in many respects. Till the end of this paragraph whenever we shall say functions we shall mean functionals in which the function variables range over the b.e.v.f.'s. Since we deal with b.e.v.f.'s we do not have to follow Machover [6] and consider recursive functions with any number $< \aleph_1$ of number variables since any number $< \aleph_1$ of number variables can be replaced by a single b.e.v.f. variable.

It is shown in this work that the notions of finitarily recursive functions for the three approaches (Takeuti's, Machover's and the author's) coincide, and the same is true for numerically recursive functions and general recursive functions. The numerically recursive functions are obtained from the finitarily recursive functions by substitution of fixed numbers for some of the number variables. The general recursive functions are obtained from the finitarily recursive functions by substitution of fixed b.e.v.f.'s for some of the function variables. Following Takeuti-Kino and Machover we get also a normal form theorem, an enumeration theorem and a hierarchy theorem.

The same results as mentioned for functions are obtained for functionals but only for a regular $\aleph$. For a singular $\aleph$ some of the theorems mentioned above are not true for functionals. The situation has not yet been cleared up in this case.

It is worth mentioning that, for $\aleph > \omega$, even though we said that the clerk has to finish the computation in $< \aleph$ time and use $< \aleph$ paper it follows from a result of Takeuti that as far as recursive functions are concerned these restrictions can be lifted without adding any new recursive functions. This is rather surprising because it goes contrary to what happens in the case $\aleph = \omega$. In the latter case, if $R(x,y)$ is a recursive predicate such that the predicate $(\exists y)R(x,y)$ is undecidable then if we lift the restriction that the computation should be finished in finite time then the predicate $(\exists y)R(x,y)$ becomes decidable (and its characteristic function recursive) since for any x the

clerk knows after ω steps whether there is such a y or not. What happens in the transfinite case when the clerk checking if there is a number y (i.e., $y < \aleph$) such that $R(x, y)$ is given more than $\aleph$ time and space? We shall now give an indication why in this case the predicate $(\exists y)R(x, y)$ does not become decidable. The clerk keeps track of the number y for which he is checking $R(x, y)$ by having y written, in some part of the sheet on which he computes. Suppose that he writes y by y consecutive bars $||| \dots$. When he finds out that $R(x, y)$ does not hold he adds a bar to y and starts checking $R(x, y)$ for the new y . Suppose now he was given a sheet larger than $\aleph$ and he was given more than $\aleph$ time. Suppose he found out that $R(x, y)$ does not hold for any $y < \aleph$. $\aleph$ is now written, by means of a sequence of $\aleph$ bars, in the part of the sheet devoted to y . He suspects that this is indeed $\aleph$ but how can he verify his suspicion? $\aleph$ is not written on the instruction sheet because the instruction sheet can contain only $< \aleph$ symbols. So, unable to confirm his suspicion, he continues to check $R(x, y)$ for y 's greater than or equal to $\aleph$. Thus either he will go on getting answers that $R(x, y)$ does not hold and at no instant will he be able to tell whether there is a number $y < \aleph$ such that $R(x, y)$ (and this will indeed happen always, as we shall see) or otherwise (as far as we know now) he finds out that $R(x, y)$ holds for some $y \geq \aleph$ and, not knowing that this y is $\geq \aleph$, will give us the false reply that there is a $y < \aleph$ for which $R(x, y)$ holds. This does not happen for $\aleph = \omega$ because the clerk can easily tell whether the y he has

is already ω or not by looking to see if it has a last bar or not.

In §5 we apply a central theorem of Takeuti to give an alternative proof of a generalization of theorem 12.2 of Gödel [1] (from which it follows immediately that the generalized continuum hypothesis holds in the model Δ). Even the more difficult theorem 4.10.3 of Hajnal [2] is obtained from Takeuti's theorem. The total length of the alternative proofs given here is not shorter than that of the original proofs since in order to prove those results from Takeuti's theorem we have to carry out an arithmetization of the notion of relative constructibility, but the new proofs are routine whereas the original ones are somewhat unexpected.

The special role played by the b.e.v.f.'s in the theory and the use of the notion of relative constructibility developed by Hajnal [2], Shoenfield [10] and the author [5] allows us to dispense with the axiom of constructibility used by Takeuti-Kino [14] and Machover [6]. In the present work all the proofs will be based only on the axiom groups A,B,C of Gödel [1] and we shall not use even the axioms of foundation and choice.

Some of the results of this work were announced in an abstract titled "Transfinite computability" which appeared in the Notices of the American Mathematical Society, vol. 10 (1963), p.286.

2. RECURSIVE FUNCTIONALS

We shall now give the definition of the notion of recursive functionals according to Takeuti [13], which we shall call finitarily recursive functionals. We shall present a version somewhat simpler than that in [13]². We are able to give this simpler version because we shall not deal with the notion of a primitive recursive functional. The function variables will vary only over partial functions of a single (number) variable. The more general case, of function variables ranging over the set of partial functions of several variables, is only a trivial generalization since, as we shall see, there is a finitarily recursive pairing function with finitarily recursive inverse functions and therefore functions of several variables can be reduced to functions of one variable. Unless otherwise mentioned we shall mean by a (partial) function a (partial) function of a single variable.

The symbols f_i, g_i, h_i will stand for variables which range over the set of partial functions, and x_i, y_i, z_i will stand for number variables. However, if given a function $F(x, y, z)$ which we shall wish

²The present version gives essentially a definition of the notion of a recursive functional for every ordinal $\Omega \geq 2$, not necessarily a cardinal.

to regard as a function of the single variable x , with y, z as parameters, we shall denote it with $\hat{x}F(x, y, z)$. If $g(x)$ is a partial function we shall denote with $g \upharpoonright y$ (g restricted to y) the partial

function defined by $(g \upharpoonright y)(x) = \begin{cases} g(x) & \text{if } x < y \\ 0 & \text{if } x \geq y \end{cases}$. $(\mu x)g(x)$ will

denote the least number α such that $g(\alpha) = 0$ and for every $\beta < \alpha$ $g(\beta)$ is defined. If there is no such number α ($< \Omega$) then $(\mu x)g(x)$ is undefined.

DEFINITION 1. $F(f_1, \dots, f_m, x_1, \dots, x_n)$, $0 \leq m < \omega$, $0 \leq n < \omega$ is a finitarily recursive functional (f.r.f.) if and only if one of the following holds.

$$(1) \quad F(f_1, \dots, f_m, x_1, \dots, x_n) = P^{(n, i)}(x_1, \dots, x_n) = x_i, \quad 1 \leq i \leq n.$$

$$(2) \quad F(f_1, \dots, f_m, x) = N(x) = \begin{cases} 0 & \text{if } x > 0 \\ 1 & \text{if } x = 0 \end{cases}$$

$$(3) \quad F(f_1, \dots, f_m, x, y) = \wedge(x, y) = \begin{cases} 0 & \text{if } x, y = 0 \\ 1 & \text{if } x > 0 \text{ or } y > 0 \end{cases}$$

$$(4) \quad F(f_1, \dots, f_m, x, y) = <(x, y) = \begin{cases} 0 & \text{if } x < y \\ 1 & \text{if } x \geq y \end{cases}$$

$$(5) \quad F(f_1, \dots, f_m, x) = f_i(x), \quad 1 \leq i \leq m$$

$$(6) \quad F(f_1, \dots, f_m, x_1, \dots, x_n) = G(f_1, \dots, f_m, H_1(f_1, \dots, x_n), \dots, H_1(f_1, \dots, x_n))$$

where $G(f_1, \dots, f_m, x_1, \dots, x_1)$, $H_1, \dots, H_1$ are f.r.f.'s.

$F(f_1, \dots, f_m, x_1, \dots, x_n)$ is defined only if all the terms on the right-hand side are defined.

$$(7) \quad F(f_1, \dots, f_m, x_1, \dots, x_n) = G(\hat{y}^F(f_1, \dots, f_m, y, x_1, \dots, x_n) 1_{x_1}, f_1, \dots, f_m, x_1, \dots, x_n)$$

where $G(f_0, f_1, \dots, f_m, x_1, \dots, x_n)$ is a f.r.f. In this case we say that

F is obtained from G by recursion on f_0 and x_1 .

$$(8) \quad F(f_1, \dots, f_m, x_1, \dots, x_n) = (\mu_{x_{n+1}})G(f_1, \dots, f_m, x_1, \dots, x_n, x_{n+1})$$

where $G(f_1, \dots, f_m, x_1, \dots, x_n, x_{n+1})$ is a f.r.f. Note that this is the only schema in which the value of F depends on $\mathfrak{Q}^3$.

Schema (6) permits substitution of values of f.r.f.'s for number variables. The following Lemma, which is, essentially, Proposition 2 of Takeuti [13], will permit substitution of f.r.f.'s for function variables.

³Every functional which is recursive according to Takeuti [13] is a f.r.f., as follows from Lemma 1, (22) and (28) of §6, and since $B_m(f, x)$ of Takeuti can be replaced by $(\mu y)[f(y) = 0 \vee (\forall z < x)(f(z) \neq 0)]$ - see the first part of §6. Every f.r.f. which is a total function, with the function variables varying over total functions only (this is the case considered by Takeuti) is recursive according to Takeuti, as follows from Proposition 2 of [13].

LEMMA 1. Let $F(f_1, \dots, f_m, x_1, \dots, x_n)$, $G_i(f_1, \dots, f_k, x_0, x_1, \dots, x_n)$,
 $1 \leq i \leq n$, be f.r.f.'s then

$$H(f_1, \dots, f_k, x_1, \dots, x_n) = F(\hat{x}_0 G_1(f_1, \dots, f_k, x_0, x_1, \dots, x_n), \dots, \hat{x}_0 G_m(f_1, \dots, f_k, x_0, x_1, \dots, x_n), x_1, \dots, x_n)$$

is a f.r.f.

PROOF. By induction on the construction of F . If F is obtained by one of the schemata (1) - (4) this is trivial.

$$(5) \quad H(f_1, \dots, f_k, x) = [\hat{x}_0 G_i(f_1, \dots, f_k, x_0, x)](x) = G_i(f_1, \dots, f_k, x, x).$$

$G_i(f_1, \dots, f_k, x, x)$ is obtained from the f.r.f. $G_i(f_1, \dots, f_k, x_0, x)$ by schemata (1) and (6), and hence is a f.r.f.

$$(6) \quad H(f_1, \dots, x_n) = G(\hat{x}_0 G_1(f_1, \dots, x_n), \dots, \hat{x}_0 G_m(f_1, \dots, x_n), H_1(\hat{x}_0 G_1(f_1, \dots, x_n), \dots, x_n), \dots, H_1(\hat{x}_0 G_1(f_1, \dots, x_n), \dots, x_n)).$$

By the induction hypothesis $G(\hat{x}_0 G_1(f_1, \dots, x_n), \dots, \hat{x}_0 G_m(f_1, \dots, x_n), y_1, \dots, y_n)$ is a f.r.f. (note that by schemata (1) and (6) every f.r.f., with number variables $x_1, \dots, x_n$ can be regarded also as a f.r.f. with additional number variables $y_1, \dots, y_r$) and also $H_i(\hat{x}_0 G_1(f_1, \dots, x_n), \dots, \hat{x}_0 G_m(f_1, \dots, x_n), x_1, \dots, x_n)$ are f.r.f.'s. By schema (6) (and schema (1)) $H(f_1, \dots, x_n)$ is a f.r.f.

If F is obtained by schema (7) or schema (8) then the Lemma follows immediately from the induction hypothesis.

It follows immediately from Definition 1 that there are only $\aleph_0$ f.r.f.'s. Thus, for $\aleph_0 > \omega$, not even all constant functionals are f.r.f.'s. The following notions of functionals are in certain respects nearer to the notion of a recursive functional in the finite case.

DEFINITION 2. The functionals obtained from the f.r.f.'s by substitution of numbers for some of their number variables are called numerically recursive functionals (n.r.f.'s). The functionals obtained from f.r.f.'s by substitution of bounded eventually vanishing functions (b.e.v.f.'s), (i.e., functions φ for which there is a number β such that $\varphi(\alpha) = 0$ for $\alpha \geq \beta$ and $\varphi(\alpha) < \beta$ for all α) for some of the function variables are called general recursive functionals (g.r.f.'s).

Obviously every f.r.f. is a n.r.f. Since each number α can be written as $(\mu x)N(\varphi_\alpha(x))$, where φ_α is a b.e.v.f. defined by

$$\varphi_\alpha(x) = \begin{cases} 0 & x \neq \alpha \\ 1 & x = \alpha \end{cases} \quad \text{every n.r.f. is a g.r.f.} \quad \text{If } \aleph_0 = \omega \text{ then,}$$

as one can easily see, the three notions coincide with the ordinary notion of recursive functional. If $\aleph_0 > \omega$ then there are n.r.f.'s which are not f.r.f.'s, since there are $\aleph_0$ n.r.f.'s and only $\aleph_0$ f.r.f.'s. It will follow from the results in §6 and §7 that the two notions of a n.r.f. and a g.r.f. coincide for $\aleph_0$ if and only if every b.e.v.f. (relative to $\aleph_0$) is constructible in the sense of Gödel [1]. Therefore it is not difficult to see that in the set theory

which consists of axiom groups A, B, C, D of Gödel [1] the notion of a n.r.f. and a g.r.f. coincide for every Ω if and only if Gödel's axiom of constructibility, $V = L$, holds⁴.

⁴See [5], 3.0.

3. COMPUTABLE FUNCTIONALS

We shall now develop a notion of Turing machines for the computation of functionals. Those machines will be similar to the ordinary Turing machines used in the finite case. We shall have two kinds of Turing machines, which will differ only in one respect, ordered Turing machines and discrete Turing machines.

A Turing machine will consist of the following items.

(a) A transfinite tape of length Ω , i.e., a tape divided to squares $\{p_\rho \mid \rho < \Omega\}$ which are ordered according to their indices and thus form an ordered set of order type Ω .

(b) An alphabet which is a set of symbols $S = \{s_\tau \mid \tau < \Lambda\}$, where $2 \leq \Lambda < \Omega$ and Λ is not a limit number. Every square on the tape always contains a single symbol. The symbol s_0 is called the blank symbol.

(c) A set of internal states $Q = \{q_\sigma \mid \sigma < \Gamma\}$, where $1 \leq \Gamma < \Omega$, and, in the case of an ordered machine, Γ is not a limit number.

At every moment the machine is said to be in one of the internal states. Among the states we distinguish two states: the initial state which, in the case of a discrete machine we denote with q_1 , and a terminal state which we denote with q_0 . $P_1, \dots, P_m$ are pairwise disjoint sets of (internal) states neither of which contains the terminal state.

(d) A function M_0 defined on $(Q - \{q_0\} - \bigcup_{i=1}^m P_i) \times S$ into $Q \times S \times 3$.

(e) A function M_1 defined on $\bigcup_{i=1}^m P_i$ into Q .

(f) A discrete machine has a function M_2 defined on the set $\mathcal{P}(Q - \{q_0\})$ of all subsets of $Q - \{p_0\}$ and into Q such that M_2 is constant outside a subset $P^* \subseteq \mathcal{P}(Q - \{q_0\})$ of cardinality $< \aleph$ (This condition is automatically fulfilled if $2^{\aleph} < \aleph$, and in particular if the axiom of choice holds and $\aleph$ is strongly inaccessible).

The way in which a given Turing machine is supposed to compute a given functional $F(f_1, \dots, f_m, x_1, \dots, x_n)$ is, roughly, as follows. Every number α is represented on the tape by a sequence $s_1^{1+\alpha} s_0$ (i.e., a sequence of $1+\alpha$ s_1 's and a single s_0), which we shall denote with $\underline{\alpha}$, preceded by s_0 . The machine starts with a tape on which $\underline{\alpha}_1, \dots, \underline{\alpha}_n$ are printed in an appropriate way and performs a computation during which it may ask for values of the partial functions $\varphi_1, \dots, \varphi_m$ for certain arguments. Those values are "fed" into it. If $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \beta$ then the machine finally stops with β printed in an appropriate place on the tape. If $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is undefined the machine never stops.

By a configuration we mean an ordered triple of an internal state, a printed tape (i.e., a function on $\{p_j \mid j < \aleph\}$ into S) and a square p_j on the tape. We say that the machine is in the configuration $\langle a, b, d \rangle$ if its internal state is a , its tape is b and it scans the square d . If $a \neq q_0$ the configuration is said to be active, if $a = q_0$ it is said to be inactive. An atomic act of a machine is to pass from an active

configuration c_0 to a configuration c_1 which is uniquely determined by three factors: The structure of the machine, m given partial functions $\varphi_1, \dots, \varphi_m$ and c_0 . Let c_0 be such that the machine scans a square p_j in which the symbol s_τ is printed and the internal state of the machine is q_σ ($\neq q_0$). c_1 is determined as follows. If $q_\sigma \notin \bigcup_{i=1}^m P_i$ then let $M_0(q_\sigma, s_\tau) = \langle q_\sigma, s_\tau, t \rangle$. The machine goes over to the internal state q_σ and prints s_τ in the square p_j (erasing s_τ). If $t = 0$ the machine "moves one square right" to scan the square p_{j+1} ; if $t = 1$ the machine stays scanning the same square p_j ; if $t = 2$ we have to consider three cases: (1) j is a successor number, then the machine "moves one square left" to scan the square p_{j-1} ; (2) j is not a successor number and for some $\gamma < j$ all the squares of the set $\{p_\xi \mid \gamma \leq \xi < j\}$ have the same symbol printed in them in c_0 , then the machine "moves left" to scan the square p_γ with the least γ as above; (3) if neither (1) nor (2) is the case, then the machine stays scanning p_j . If $q_\sigma \in P_i$, $1 \leq i \leq m$, then if p_j is the last square of a segment $s_0 \alpha$ of the tape in c_0 (i.e., there is a number λ such that $j = \lambda + 2 + \alpha$ and such that s_0 is printed in p_λ and p_j and s_1 is printed in p_j for $\lambda < \xi < j$) then this means that the machine wants to know $\varphi_i(\alpha)$. If $\varphi_i(\alpha) = \delta$ then δ is fed into the machine by an external agency which prints δ to the right of p_j (erasing whatever is printed in these squares), the machine stays scanning the same square p_j and its internal state becomes $M_1(q_\sigma)$.

If there is no α as above, or if there is such an α and $\varphi_1(\alpha)$ is undefined then c_1 is the same as c_0 .

During the computation the machine can go through a limit number of configurations. What will be the following configuration? Assume that the machine went through a sequence of configurations $\{c_\xi | \xi < \zeta\}$ where ζ is a limit number and let us denote with c_ζ the configuration of the machine which follows the ζ configurations. c_ζ depends only on the sequence $\{c_\xi | \xi < \zeta\}$ and, in the discrete case, also on the function M_2 . Let $c_\xi = \langle a_\xi, b_\xi, d_\xi \rangle$ for $\xi \leq \zeta$. Let us deal first with the discrete machine. We shall say that a state q_σ was frequented in the sequence $\{c_\xi | \xi < \zeta\}$ if there are arbitrarily large $\xi < \zeta$ such that $a_\xi = q_\sigma$. a_ζ is the value of M_2 at the set of all states frequented in the sequence $\{c_\xi | \xi < \zeta\}$. For the ordered machine let $a_\xi = q_{\sigma(\xi)}$ for $\xi \leq \zeta$, then we take $\sigma(\zeta) = \liminf_{\xi < \zeta} \sigma(\xi)$.

$\liminf_{\xi < \zeta} \sigma(\xi) < \Gamma$ since for an ordered machine Γ is not a limit number.

For $\xi \leq \zeta$ and $\lambda < \Omega$ let $s_{\tau(\xi, \lambda)}$ be the symbol in the square p_λ on the tape a_ξ . We put $\tau(\zeta, \lambda) = \liminf_{\xi < \zeta} \tau(\xi, \lambda)$; $\liminf_{\xi < \zeta} \tau(\xi, \lambda) < \Lambda$

since Λ is not a limit number. For $\xi \leq \zeta$ let $d_\xi = p_{\tau(\xi)}$. We put $p(\zeta) = \liminf_{\xi < \zeta} p(\xi)$. Since each configuration $c_{\xi+1}$ was obtained

from c_ξ by an atomic act during which the machine moved at most one square to the right, and for a limit number ξ c_ξ was obtained as indicated here, it is easy to prove by induction on ξ that $p(\xi) \leq p(0) + \xi$,

and hence $\liminf_{\xi < \zeta} f(\xi) \leq f(o) + \zeta < \Omega$.

We shall say that a given machine computes the functional $F(f_1, \dots, f_m, x_1, \dots, x_n)$ if and only if for every m partial functions $\varphi_1, \dots, \varphi_m$ and for every n numbers $\alpha_1, \dots, \alpha_n$ whenever the machine starts in a configuration c_0 , in which it is in the initial state, the tape is $s_0 \alpha_1 \dots \alpha_n s_0^\Omega$ and the machine scans the last square of α_n (which is blank), and goes through a sequence of active configurations c_ξ according to the rules given above with $\varphi_1, \dots, \varphi_m$ as the given partial functions then the machine reaches an inactive configuration c_γ after $\gamma < \Omega$ acts and in that configuration the machine scans one of the squares of a sequence $s_0 \beta$ if and only if $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \beta$ (Thus if $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is undefined then the machine goes through Ω acts without getting into an inactive configuration - in which case we say that the computation diverges).

An ordered or discrete Turing machine for which Λ and Γ are finite is called finitary. It is easily seen that an ordered finitary machine acts exactly like the discrete machine which has the same structure and with the function M_2 defined by $M_2(P) =$ the member of P of least index, for every non-void set P of non-terminal states, and $M_2(0)$ is arbitrary. A Turing machine for which the sets $P_1, \dots, P_m$ and the functions M_0 and M_1 (regarded as sets and functions on the

indices of the q_σ 's and the s_τ 's) are constructible in the sense of Gödel [1], and, for a discrete machine, the function M_2 restricted to constructible subsets of $Q - \{q_0\}$ is constructible and M_2 is constant outside a subset of $\mathcal{P}(Q - \{q_0\})$ which can be mapped on an ordinal $< \Omega$ by a constructible one-one mapping, is called numerical. A functional which is computed by a finitary (ordered or discrete) Turing machine is said to be a finitarily computable functional (f.c.f.); a functional which is computed by a numerical Turing machine is said to be a numerically computable functional (n.c.f.); a functional which is computed by a Turing machine is said to be a general computable functional (g.c.f.).

We arrive now at the problem of the comparison of the respective notions of recursivity and computability. Furthermore, by distinguishing between ordered and discrete machines and by putting restrictions on the size of Λ and on some other details of the structure or the action of the machine we can get other notions of computability, which have again to be compared to recursivity and to each other. We shall give a full answer to most of these problems in the cases where either Ω is a regular cardinal or else Ω is any cardinal and the function variables of the functionals vary only over b.e.v.f.'s. In these cases we shall prove: Every (finitarily, numerically, general) computable functional is (f., n., g.) recursive. Every (f., n., g.) recursive functional is computable by both an ordered and a discrete (finitary, numerical, general) Turing machine for which $\Lambda = 3$, i.e., whose alphabet consists of the two symbols

s_1 and s_2 and the blank s_0 , and which satisfies the following additional conditions.

(a) If the machine starts in the initial state with the tape

$s_0^{\lambda+1} \alpha_1 \dots \alpha_n s_0^\Omega$, where $0 \leq \lambda < \Omega$, and it scans the last square of α_n then if $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \beta$ then after $< \Omega$ acts the machine arrives at an inactive configuration where the tape is $s_0^{\lambda+1} \alpha_1 \dots \alpha_n \beta s_0^\Omega$ and the machine scans the last square of β ; if $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is undefined then the computation diverges.

(b) In each configuration which occurs in the computation there is only a finite number of blocks of symbols on the tape (By a block of symbols we mean a segment of the tape of the form s_c^ξ , $0 < \xi \leq \Omega$, such that no segment of the tape which properly includes it is of the form s_c^η).

(c) Each P_i consists of at most one state.

(d) In each configuration $\langle a, b, d \rangle$ which occurs during the computation

for which the state a is in $\bigcup_{i=1}^m P_i$ the tape is of the form $\dots s_0 \alpha s_0^\Omega$ and the square d scanned is the last square of α .

As will be seen later we could even put the additional requirement that in each configuration of the computation there is at most one block of s_2 's on the tape. Furthermore we could even have $\Lambda = 2$, i.e., we could use only the symbol s_1 and the blank s_0 , if we would use a somewhat different notion of a Turing machine. The author does not know whether $\Lambda = 2$ is enough for the present notion of a Turing machine. Note that the proof in the finite case, that any finite number of symbols can be replaced by two since finitely many symbols can be represented by finite sequences of two symbols, cannot be applied here, since if we replace the s_2 's by sequences of s_0 's and s_1 's requirement (b) above will not be satisfied, but requirement (b) is essential to the computation of the recursive functionals.

4. THE COMPUTABILITY OF RECURSIVE FUNCTIONALS FOR A REGULAR Ω

The aim of this paragraph is to prove that for a regular Ω every (finitarily, numerically, general) recursive functional is (f.,n.,g.) computable. Before we start to prove it we have to observe some of the things a Turing machine can do.

Whenever we shall mention machines we shall mean, unless otherwise stated, finitary ordered Turing machines. We shall denote the symbols s_0, s_1, s_2 also by 0,1,2, respectively. Also the states will be denoted with numbers, but we shall not assume that λ denotes the state q_λ , so that the numbers which we shall use will not indicate the order of the states (according to their indices). Whenever we shall need to know the ordering of the states this ordering will be given explicitly. 1 will always denote the initial state and 0 the terminal state. By an instruction we shall understand a set of states with two states specified as the initial and terminal states and with a function like M_0 defined on ordered pairs consisting of a non-terminal state of the instruction and a symbol. Unless otherwise stated the set of the states of an instruction is ordered. The instructions will serve as building blocks for more complicated instructions and for machines. If A,B are instructions then the instruction A-B will be the instruction whose states will be the states of the instructions A,B (A and B are assumed to have no common states) with the terminal state of A identified with the initial state of B. The initial and terminal states of A-B will be the initial state of A and the terminal state of B, respectively.

The function M_0 of $A-B$ is the union of the corresponding functions of A and B . The order of the states has to be explicitly stated (so indeed the name $A-B$ will correspond to several different machines, but we shall always deal with only one of them which will be given explicitly). Some of the instructions will have two terminal states 0_1 and 0_2 . If A is such an instruction then by $A \begin{cases} -B \\ -C \end{cases}$ we shall mean the machine with the states of A, B, C , where the state 0_1 of A is identified with the initial state of B , the state 0_2 of A is identified with the initial state of C and whose initial state is the initial state of A . We shall develop additional notations for composition of instructions as we go along. We shall also check, while describing the instructions, that they satisfy the requirement (b) of §3. This requirement, as well as the way in which the instruction acts, will be checked

only under specified initial configurations. When we use the instructions to build machines we shall do it so that the instructions will be applied only when one of the prescribed configurations arises.

The function M_0 of an instruction will be given by a table which is like the multiplication table. The first component of the values of M_0 , which is s_0, s_1 or s_2 , will be denoted with E (read: Erase), P (read: Print) or P_2 (read: Print 2), respectively. This will be written only if the symbol in the square scanned has to be changed, otherwise it will be omitted. The third component of the value of M_0 will be denoted with R (read: Right) or L (read: Left) if it is 0 or 2, respectively, and will be omitted if it is 1. Thus P_n will

be read "Print (1) and enter state n", RO will be read "Move one square right and enter state 0", etc. The following instructions 1-18 will be finitary.

1) R (Right)

		symbol scanned		Unless otherwise stated the machine, when scanning the symbol 2 will react
		0	1	
instruction	state	1	RO	RO

as if it scanned the symbol 1. Thus the full table in this case is

	0	1	2	
1	RO	RO	RO	.

The instruction moves the machine one square to the right.

2) L (Left)

	0	1	The machine moves one square to the left if it is on a successor square. If it is on a limit square it moves left to the beginning of the rightmost block of symbols on the left of the square scanned, if there is such a block.
1	LO	LO	

3) P (Print)

	0	1	The machine prints if it scans a blank. It does not matter how the vacant space in the second column of the table is filled.
1	PO		

4) E (Erase)

	0	1	The machine erases if it scans a 1 (or a 2).
1		EO	

5) R to end (Right to end)

	0	1
1	R1	R2
2	0	R2

By numeral we shall mean α , for some number α , which is preceded by a block of 0's and with some of the 1's possibly replaced by 2's. The machine, when scanning a

non-blank square of a numeral or a blank square in front of a numeral (i.e., with no printed squares in between) moves to the last square of the numeral. Note that if the machine scans the printed squares during a limit number of configurations in the state 2 then the $\liminf$ of the set of states is again 2, and hence the machine stays in that state.

6) L to beg (Left to beginning)

	0	1
1	L1	L2
2	R3	L2
3	R3	0

The machine, if scanning a square of a numeral or a blank square closely behind one (by this we shall always mean a square to the right of the numeral with

no non-blank squares in between) moves to the first square of the numeral.

7) Check 1

	0	1
1	0_2	0_1

The machine, when scanning a square, enters the terminal state 0_1 if the square is printed and 0_2 if the

square is blank.

8) ER to end (Erase right to end)

	0	1
1	R1	ER2
2	0	ER2

The machine, when scanning a blank square in front of a numeral, or a non-blank square of a numeral, erases all the symbols of the numeral which are printed in

the square scanned or to the right of it and stops at the end of the numeral.

9) R to beg (Right to beginning)

	0	1
1	R2	R1
2	R2	0

The machine, when scanning a numeral or a blank square moves to the first square of the next numeral on the right.

10) Record 2

	0	1	2
1	0_2	$P_2 0_1$	R1

When the machine scans a square it moves right till it encounters the first square in which 2

is not printed (this may be also the square on which the machine started). If it encounters 1 it prints 2 on top of it and enters the terminal state 0_1 ; if it encounters 0 it enters the terminal state 0_2 .

11) Restore 1

	0	1	2
1	0	R1	PR1

The machine, when scanning a square of a numeral, replaces all the 2's in the square scanned and in

the squares of the numeral to the right of it by 1's and stops at the end of the numeral.

12) Copy r-th n.L. (Copy the r-th numeral on the left)

The machine, when scanning a free square (i.e., a blank square, such that the part of the tape to the right of it is blank) copies immediately to the right of it the r-th numeral to the left of it (where $1 \leq r$). It is assumed that there is such an r-th numeral and that it consists of 1's only. This instruction is built from simpler instructions as follows:

$$\begin{aligned}
 & \underline{R} - \underline{P} - (\underline{L} - \underline{L \text{ to beg}})^r - \underline{\text{Record 2}} - \{ * - \underline{L \text{ to beg}} - \\
 & - \underline{\text{Record 2}} \left\{ - (\underline{R \text{ to end}})^{r+1} - \underline{P} - (\underline{L \text{ to beg}} - \underline{L})^r - * \right. \\
 & \quad \left. - \underline{L \text{ to beg}} - \underline{\text{Restore 1}} - (\underline{R \text{ to end}})^r - C. \right.
 \end{aligned}$$

If A is an instruction then A^r is defined by $A^1 = A$, $A^{n+1} = A^n - A$. The stars denote identification of states, i.e., the star between Record 2 and L to beg denotes the terminal state O_1 of Record 2 which is the initial state of L to beg and this state is identified with the state denoted by the second star (which is the terminal state of the last L in $(\overline{L \text{ to beg}} - \overline{L})^r$). It is immaterial how M_1 is defined for the state O_2 of the first Record 2 and therefore nothing is written there. The 0 at the end means that the terminal state of the last R to end is the terminal state of the instruction Copy r-th n.L. Till now the order of the states of the various instructions did not matter; now we specify that state 3 of the overlined L to beg will be the first state in the ordering of the states (the order among the other states is immaterial). The machine acts as follows. It moves one step right, prints 1, passes left to the r-th numeral, changes its first symbol to 2 and checks the second square of that numeral. If that square contains 1 this 1 is changed to 2, then the machine moves right to the end of the copy (i.e., to the single 1 which is all we have yet of the copy) adds 1, returns again to the beginning of the copied numeral, looks for the first 1, if there is such, changes it to a 2, moves right to the end of the copy adds 1, and vice versa. After the machine goes through this cycle a limit number of times it finds itself in the 3-rd state of the overlined L to beg and it is scanning a blank square in front of the copied numeral. It will then move right to the beginning of the copied numeral, look for a 1 and vice versa. Finally, when the

machine checks the copied numeral and find no 1's the copying is completed and machine changes all the 2's of the copied numeral back to 1's, returns to the end of the copy and stops. We note that during the copying only three blocks of symbols were added to the tape and the machine did not move left of the 0 in front of the copied numeral if there is such a 0.

13) L to end (Left to end)

	0	1
1	L2	L1
2	L2	R3
3	0	R3

The machine, when scanning a non-blank square of a numeral or any blank square moves left to the end of the first numeral to the left (if there is such a numeral).

14) Check k-gap

	0	1
	1 R2	R1
$2 \leq i \leq k$	i R i+1	0_2
	k+1 L k+2	0_2
	k+2 L k+2	R k+3
	k+3 0_1	R k+3

The machine, when scanning a square of a numeral or a blank square checks whether the k squares to the right of the numeral, or to the right of the blank square scanned, respectively, are blank. If not all

of them are blank the machine stops at the first non-blank one in the terminal state 0_2 ; if all of them are blank the machine stops on the last square of the first numeral to their left, if there is such a numeral, in the state 0_1 .

15) Move in next n. (Move in next number)

The machine, when scanning the last square of a numeral or a blank square closely behind it with a single numeral, consisting of 1's only, on

its right moves the numeral on the right left so as to close the gap between the two numerals. When it ends the machine scans the last square of the moved numeral. The instruction is:

$$\begin{aligned} & \underline{R \text{ to end}} - \underline{\text{Copy 1-st n.L.}} - \underline{R} - \underline{P} - (\underline{L} - \underline{L \text{ to beg}})^2 - \underline{ER \text{ to end}} - \\ & - \underline{L \text{ to end}} - \underline{R} - * - \underline{P} - \underline{R} - \underline{R \text{ to end}} - \underline{R} - \underline{\text{Check 1}} \begin{cases} \underline{L} - \underline{L \text{ to beg}} - \underline{E} - \underline{L \text{ to end}} - * \\ \underline{L^2} - \underline{E} - \underline{L \text{ to end}} - \underline{L} - \underline{E} - 0. \end{cases} \end{aligned}$$

The order of the states is such that the state 3 of the overlined L to end precedes all the other states of the instructions between the stars and that the internal order of the states of Copy 1-st n.L. is unchanged. That the machine does indeed do what we claim it does follows clearly from its construction. We note that during the whole operation the machine does not move left of the numeral towards which the other numeral is being moved. Also not more than four additional blocks of symbols are present in any configuration of the operation.

16) Move next n. on top. (Move next numeral on top)

The machine, when scanning the last square of 0 (i.e., $s_1 s_0$) or a blank square closely behind it, with a single numeral consisting of 1's only on its right, moves the numeral on the right left so that its first 1 coincides with the 1 of the 0. At the end the machine scans the last square of the moved numeral. The instruction is

$$\begin{aligned} & \underline{R \text{ to end}} - \underline{\text{Copy 1-st n.L.}} - \underline{R} - \underline{P} - (\underline{L} - \underline{L \text{ to beg}})^2 - \underline{ER \text{ to end}} - \underline{L \text{ to end}} - * - \\ & - \underline{R \text{ to end}} - \underline{R} - \underline{\text{Check 1}} \begin{cases} \underline{L} - \underline{L \text{ to beg}} - \underline{E} - \underline{L \text{ to end}} - \underline{P} - \underline{R} - * \\ \underline{L^2} - \underline{E} - \underline{L \text{ to end}} - \underline{L} - \underline{E} - 0. \end{cases} \end{aligned}$$

The order of the states and all other comments are as in the case of

Move in next n.

17) End_k

The machine moves right till it encounters $k+2$ or more successive blanks in which case it stops on the last square of the last numeral preceding those blanks. The instruction is $* - \underline{\text{Check } k+1 - \text{gap}} \left\{ \begin{array}{l} - 0. \\ - \underline{\text{R to end}} - * \end{array} \right.$

The states of Check $k+1 - \text{gap}$ precede those of R to end.

18) Copy 1 to j across k - gap

Given two consecutive numerals we say that there is a gap between them if there are blank squares between them (the last square of the first numeral, which is blank, is not counted). The number of those blank squares is the order of the gap. A k - gap is a gap of order $\geq k$. If the last square of the $j - 1$ -st numeral to the right of the nearest $k - \text{gap}$ (on its left), $j \geq 2$, is free and the machine scans it then the machine copies immediately to the right of it the i -th numeral to the right of the nearest but one $k - \text{gap}$ (on its left). The instruction is $* - (\underline{\text{L to beg}} - \underline{\text{L}})^{j-1} - \textcircled{1} - \underline{\text{L to beg}} - \underline{\text{L to end}} - \underline{\text{Check } k\text{-gap}} \left\{ \begin{array}{l} - (\underline{\text{R to beg}})^i - \\ - \underline{\text{L}} - \textcircled{1} \end{array} \right.$
 $- \underline{\text{Record 2}} \left\{ \begin{array}{l} - \textcircled{2} - \underline{\text{R to end}} - \underline{\text{Check } k\text{-gap}} \left\{ \begin{array}{l} - (\underline{\text{R to end}})^{j-1} - \underline{\text{R}} - \\ - \textcircled{2} \end{array} \right. \\ - \underline{\text{Check 1}} \left\{ \underline{\text{R to end}} - \underline{\text{P}} - \underline{\text{L to beg}} - \underline{\text{L}} - * \right. \end{array} \right.$
 $\left. \begin{array}{l} - \underline{\text{L to beg}} - \underline{\text{Restore 1}} - \textcircled{3} - \underline{\text{Check } k\text{-gap}} \left\{ \begin{array}{l} - (\underline{\text{R to end}})^j - 0. \\ - \underline{\text{R to end}} - \textcircled{3} \end{array} \right. \end{array} \right.$

Numbers surrounded by circles denote identification of states, in the same way as stars do. However, these states are entered only a finite number of times between consecutive entries in the state denoted by the stars (but the machine can stay in each such state during an infinite number of consecutive configurations) since we assume that there are only finitely many blocks of symbols on the tape. The states are ordered so that the state 3 of the overlined L to end precedes all the states of the instructions between the stars. The way in which the instruction is carried out is quite clear from its description. We note that during the operation at most four blocks of symbols are added and that the machine does not move left of the numeral in front of the nearest but one k - gap on the left.

THEOREM 2. If Ω is regular then every finitarily recursive functional is computable by a finitary Turing machine with $\Lambda = 3$ such that requirements (a) - (d) of §3 are satisfied.

PROOF. We shall distinguish between two functionals which are the same but have different definitions. We shall define the recursion rank $\rho(F)$ of a f.r.f. F as follows.

$$\text{If } F \text{ is defined by schema } \left\{ \begin{array}{ll} (1) - (5) & \rho(F) = 0 \\ (6) & \rho(F) = \max(\rho(G), \rho(H_1), \dots, \rho(H_1)) \\ (7) & \rho(F) = \rho(G) + 1 \\ (8) & \rho(F) = \rho(G) \end{array} \right.$$

$\rho(F)$ is, obviously, a finite number.

For technical reasons we shall prove a somewhat stronger theorem, as we shall demand that in addition to requirements (a) - (d) also the following requirements (e) - (h) will be satisfied. A certain subset of the set of states is singled out and its members are called distinguished states.

(e) If the machine starts on a tape $\dots s_0 \alpha_1 \dots \alpha_n s_0^{\Omega}$ and it scans the last square of α_n it never moves during the computation left of the 0 written in front of α_1 and it does not change $\alpha_1 \dots \alpha_n$ and the 0 in front of α_1 , except for changing some of the 1's to 2's and then rechanging them to 1's.

(f) If $\rho(F) = k$ then in every configuration in which the machine is in a state in $\bigcup_{i=1}^m P_i$ or in a distinguished state then there are no $k+1$ - gaps on the tape to the right of the original $\alpha_1 \dots \alpha_n$.

(g) After ζ configurations, where ζ is a limit number, if the state of the machine was frequently in $\bigcup_{i=1}^m P_i$ (i.e., if the configurations were $\alpha_{\zeta} = \langle a_{\zeta}, b_{\zeta}, d_{\zeta} \rangle$, $\zeta < \zeta$ and $a_{\zeta} \in \bigcup_{i=1}^m P_i$ for arbitrarily large $\zeta < \zeta$) then in the following configuration the state of the machine is distinguished.

(h) For every finite $k \geq \rho(F)$ we can assume that the function M_0 of the machine is such that every distinguished state is an initial state of an instruction End_k of the machine.

We proceed now to prove the strengthened theorem by induction on

the definition of F . We shall consider eight cases according to the eight schemas for defining f.r.f.'s.

(1) $F(f_1, \dots, f_m, x_1, \dots, x_n) = x_1$. F is computed by the machine

Copy n-i+1-st n.L. All the P_i 's are void and there are no distinguished states. (a), (b), (c), (e) hold, as was noted; (d), (f), (g), (h) hold vacuously.

(2) $F(f_1, \dots, f_m, x) = \begin{cases} 0 & \text{if } x > 0 \\ 1 & \text{if } x = c \end{cases}$ F is computed by the machine:

	0	1
1	L2	
2	R3	L2
3	R3	R4
4	R5	R7
5	PR6	
6	PRO	
7	R6	R7

The P_i 's are void, there are no distinguished states. Requirements (a) - (h) obviously hold.

(3) $F(f_1, \dots, f_m, x, y) = \begin{cases} 0 & \text{if } x, y = 0 \\ 1 & \text{if } x > 0 \text{ or } y > 0 \end{cases}$ F is computed

by the machine

$$(\underline{L \text{ to } beg})^2 - \underline{R} - \underline{\text{Check 1}} \begin{cases} - \underline{R \text{ to } end} \\ - \underline{R \text{ to } beg} - \underline{R} - \underline{\text{Check 1}} \end{cases} \begin{cases} \underline{R \text{ to } end} - (\underline{R} - \underline{P})^2 - \underline{R} - 0. \\ - \underline{R} - \underline{P} - \underline{R} - 0. \end{cases}$$

The P_i 's are void; there are no distinguished states. Requirements (a) - (h) are obviously satisfied.

(4) $F(f_1, \dots, f_m, x, y) = \angle(x, y)$ is computed by the following machine, which we shall denote with $\angle$.

* - L to beg - L - L to beg -

$$\begin{aligned}
 & - \text{Record 2} \left\{ \begin{aligned} & - \text{R to beg} - \text{Record 2} \left\{ \begin{aligned} & - * \\ & - \text{L to beg} - \underline{\text{L}} - \text{L to beg} - \\ & - \text{Restore 1} - (\underline{\text{R}} - \underline{\text{P}})^2 - \underline{\text{R}} - 0. \end{aligned} \right. \\ & - \text{R to beg} - \text{Record 2} \left\{ \begin{aligned} & - \text{L to beg} - \underline{\text{L}} - \text{L to beg} - \\ & - \text{Restore 1} - \underline{\text{R}} - \underline{\text{P}} - \underline{\text{R}} - 0 \\ & - \text{L to beg} - \underline{\text{L}} - \text{L to beg} \\ & - \text{Restore 1} - (\underline{\text{R}} - \underline{\text{P}})^2 - \underline{\text{R}} - 0. \end{aligned} \right. \end{aligned} \right.
 \end{aligned}$$

The states are ordered so that the state 3 of the overlined L to beg precedes all other states of the instructions between the stars. All the P_i 's are void and there are no distinguished states. It is easily seen that requirements (a) - (h) are satisfied.

(5) $F(f_1, \dots, f_m, x) = f_i(x)$. This functional is computed by a machine for which $P_j = 0$ for $j \neq i$ and $P_i = \{1\}$ in addition to the state 1 the machine consists of an instruction R to end of which $M_1(1)$ is the initial state. We can write it schematically by $1 - M_1(1) - \text{R to end} - 0$. There are no distinguished states. Requirements (a) - (h) are obviously satisfied.

(6) $F(f_1, \dots, f_m, x_1, \dots, x_n) = G(f_1, \dots, f_m, H_1(f_1, \dots, x_n), \dots, H_l(f_1, \dots, x_n))$,

1 1. Let $\mathcal{G}$ be a finitary machine which computes G and $\mathcal{H}_i$ a finitary machine which computes H_i , $1 \leq i \leq l$. We assume that $\mathcal{G}, \mathcal{H}_i$, $1 \leq i \leq l$ satisfy requirements (a) - (g) and also requirement (h), where k is the same finite number for all of them, $k \geq \rho(F) \geq \rho(G), \rho(H_1), \dots, \rho(H_l)$. Then F is computed by the machine $\mathcal{F}'$, which, for $n > 0$, is given by

$$\begin{aligned}
& \mathcal{H}_1 - (\text{copy } n+1\text{-st n.L.})^n - \mathcal{H}_2 - (\text{Copy } n+1\text{-st n.L.})^n - \mathcal{H}_3 - \dots - \mathcal{H}_1 \\
& - \text{Copy } (n+1)(1-1)+1\text{-st n.L.} - \text{Copy } (n+1)(1-2)+2\text{-nd n.L.} - \\
& - \text{Copy } (n+1)(1-3)+3\text{-rd n.L.} - \dots - \text{Copy } (n+1)+1-1\text{-st n.L.} - \\
& - \text{Copy } 1\text{-th n.L.} - \mathcal{Q} - \underline{L \text{ to beg}} - \underline{L} - (\underline{L \text{ to beg}} - \text{ER to end})^{(n+1)(1-1)+1+1} \\
& - \text{Move in next n.} - 0.
\end{aligned}$$

and, for $n = 0$ is given by

$$\begin{aligned}
& \mathcal{H}_1 - \mathcal{H}_2 - \dots - \mathcal{H}_1 - \mathcal{Q} - \underline{L \text{ to beg}} - (\underline{L} - \underline{L \text{ to beg}})^1 - \underline{R} - \\
& - \text{Check 1} \left\{ - \underline{\text{ER to end}} - (\underline{\text{ER to end}})^{1-1} - \underline{\text{Move next n. on top}} - 0. \right.
\end{aligned}$$

The states of each instruction are ordered internally according to their original order. The P_i 's and M_1 are the unions of the respective P_i 's and M_1 of $\mathcal{Q}, \mathcal{H}_1, \dots, \mathcal{H}_1$; the distinguished states are those of $\mathcal{Q}, \mathcal{H}_1, \dots, \mathcal{H}_1$. It is easy to see that $\mathcal{F}'$ satisfies the requirements (a), (b), (d) - (h). This is shown by using the assumption that $\mathcal{Q}, \mathcal{H}_1, \dots, \mathcal{H}_1$ satisfy the same requirements.

As an example we shall check requirement (g). If ζ is a limit number as in (g) then, since the machine can enter states in the P_i 's only while carrying out one of the instructions $\mathcal{H}_1, \dots, \mathcal{H}_1, \mathcal{Q}$ and since these instructions are carried out in that order then, for some $\gamma < \zeta$,

all the atomic acts from the γ -th to the ζ -th were done while the machine was carrying out a particular instruction of the instructions $\mathcal{H}_1, \dots, \mathcal{H}_1, \mathcal{Q}$. Thus the ζ -th configuration is also a limit configuration with respect to that particular instruction. Hence,

by requirement (g) on that instruction, the machine is in a distinguished state of that instruction following the $\{$ configurations. But, by our definition of the distinguished states of $\mathcal{F}'$, this is also a distinguished state of $\mathcal{F}'$ and thus requirement (g) is satisfied.

However, $\mathcal{F}'$ does not necessarily satisfy requirement (c), since each set P_1 of $\mathcal{F}'$ is the union of the P_1 's of $\mathcal{H}_1, \dots, \mathcal{H}_l, \mathcal{J}$ and thus may contain up to $l+1$ members. Assume that P_1 has $r > 1$ members; we shall see that one can construct another machine $\mathcal{F}''$ which also computes F and satisfies (a), (b), (c) - (h) but in which P_1 has only one member and in which the other P_1 's have the same number of members as they have in $\mathcal{F}'$.

Let the set P_1 of $\mathcal{F}'$ consist of the states $t_1, \dots, t_r$. $\mathcal{F}''$ will be like $\mathcal{F}'$ except for the following differences: Its P_1 will consist of a single new state t ; another new state will serve as $M_1(t)$; $t_1, \dots, t_r$ will serve in $\mathcal{F}''$ as initial states of an instruction

$$\begin{array}{l}
 t_r - (\underline{R} - \underline{P})^r \\
 t_{r-1} - (\underline{R} - \underline{P})^{r-1} \\
 \hline
 t_1 - (\underline{R} - \underline{P})^1 \\
 \hline
 t_2 - (\underline{R} - \underline{P})^2 \\
 t_1 - \underline{R} - \underline{P}
 \end{array}
 \begin{array}{l}
 \diagup \\
 \diagup \\
 \hline
 \diagup \\
 \hline
 \diagup \\
 \diagup
 \end{array}
 \underline{R} - \underline{\text{Copy 2-nd n.L.}} - t - M_1(t) - \underline{L \text{ to beg}} - \underline{L}^3 -$$

$$\begin{array}{l}
 \begin{array}{c} \underline{L} - \underline{\text{Check 1}} \\ \dots \end{array} \left\{ \begin{array}{l} - (\underline{\text{ER to end}})^2 - \underline{\text{Move in next n.}} - \underline{L} - \underline{L \text{ to end}} - M_1(t_r) \\ - (\underline{\text{ER to end}})^2 - \underline{\text{Move in next n.}} - \underline{L} - \underline{L \text{ to end}} - M_1(t_{r-1}) \end{array} \right. \\
 \hline
 - \underline{\text{Check 1}} \left\{ \begin{array}{l} \underline{L} - \underline{\text{Check 1}} \left\{ \begin{array}{l} - (\underline{\text{ER to end}})^2 - \underline{\text{Move in next n.}} - \underline{L} - \underline{L \text{ to end}} - M_1(t_2) \\ - (\underline{\text{ER to end}})^2 - \underline{\text{Move in next n.}} - \underline{L} - \underline{L \text{ to end}} - M_1(t_1) \end{array} \right. \\ \hline \end{array} \right.
 \end{array}$$

By Copy 2-nd n.L. - t we mean that t is identified with the terminal state of Copy 2-nd n.L., etc. The order of the states in $\mathcal{J}'$ is the same as the order of the same states in $\mathcal{J}$; The states of Copy and Move are ordered internally according to their original order; all new states are added at the end.

To see that $\mathcal{J}'$ computes F and satisfies requirements (a), (b), (d) - (h) we note that the only difference between the computations of $\mathcal{J}$ and $\mathcal{J}'$ is that in $\mathcal{J}'$ the passage from the configuration in which the machine is in the state t_1 to the configuration in which the machine is in the state $T(t_1)$ and a new numeral β is printed on the tape involves some more computation. It is immediately seen that the outcome after each entry of the machines $\mathcal{J}$ and $\mathcal{J}'$ into a state in P_1 is the same for both machines. What happens after a limit number of such entries? The state of $\mathcal{J}'$ is like that of $\mathcal{J}$ since the new states of $\mathcal{J}'$ were added at the end. Also the tape is the same since everything that $\mathcal{J}'$ printed in addition to what $\mathcal{J}$ printed was immediately erased and the 1's changed to 2's during the copying were restored by the same instruction. The position of the machine $\mathcal{J}'$ (i.e., the square scanned by it) may, however, be left of that of the machine $\mathcal{J}$ after a limit number of entries as above.

But, by requirement (g) with respect to F $\mathcal{J}'$, and hence also $\mathcal{J}''$, will be in the same distinguished state, and by (h), (f) and (e) (with respect to $\mathcal{J}'$) will both move to the first free square on the tape, getting there in the same state. The actual checking that $\mathcal{J}''$ satisfies the requirements (a), (b), (d) - (h) is easy.

We keep changing $\mathcal{J}''$ in a similar fashion till each set P_1 has at most one member and thus we get a machine $\mathcal{J}$ which computes F and satisfies (a) - (h).

$$(7) \quad F(f_1, \dots, f_m, x_1, \dots, x_n) = G(\hat{y}F(f_1, \dots, f_m, y, x_2, \dots, x_n) \mid x_1, f_1, \dots, f_m, x_1, \dots, x_n).$$

Let $\mathcal{G}$ be a finitary machine which computes G and satisfies requirements (a) - (g) and also requirement (h), where k is some finite number $1, \dots, \infty$ $\rho(F) > \rho(G)$. If the set P_0 of $\mathcal{G}$ is void then the computation of $\mathcal{G}$ is independent on φ_0 and hence $G(f_0, f_1, \dots, f_m, x_1, \dots, x_n)$ does not depend on f_0 and thus $F(f_1, \dots, f_m, x_1, \dots, x_n) = G(f_0, f_1, \dots, f_m, x_1, \dots, x_n)$ and F is computed by a machine $\mathcal{J}$ which is exactly like $\mathcal{G}$ except that it has no set P_0 . Obviously $\mathcal{J}$ satisfies the requirements (a) - (h). From now on we shall assume that P_0 has a single member t (by requirement (c)).

Let $\mathcal{G}'$ be an instruction which is exactly like $\mathcal{G}$ except that it has two terminal states 0 (which will be denoted with $0'$) and t (which is no more in P_0), and two initial states 1 (to be denoted with $1'$) and $M_1(t)$. $\mathcal{G}'$ has no set P_0 . Let $k = \rho(F)$. The following machine $\mathcal{J}$ computes F .

$$\begin{aligned}
 & \left. \begin{aligned}
 \underline{R}^k - (\underline{R} - \underline{P} - \underline{R})^3 - (\underline{\text{Copy } n+3\text{-rd } n.L.})^n - * - 1' \\
 * - M_1(t)
 \end{aligned} \right\} \mathcal{G}' \quad \begin{aligned}
 0' \\
 t
 \end{aligned} \left\{ \begin{aligned}
 = \\
 =
 \end{aligned} \right. \\
 & = \underline{L \text{ to } beg} - (\underline{L} - \underline{L \text{ to } beg})^{n+1} - \underline{E} - \underline{R} - \underline{\text{Check 1}} \left\{ \begin{aligned}
 & - (\underline{L} - \underline{L \text{ to } beg})^2 - \\
 & - (\underline{ER \text{ to } end})^{n+3} - \\
 & - \underline{\text{Move in next } n} - \\
 & - \underline{L \text{ to } beg} - \underline{L} - * \\
 & - (\underline{L} - \underline{L \text{ to } beg})^2 - \\
 & - (\underline{ER \text{ to } end})^{n+2} - \\
 & - \underline{\text{Move in next } n} - 0.
 \end{aligned} \right. \\
 & \equiv \underline{R}^k - \underline{\text{Copy 1-st } n.L.} - \underline{\text{Copy 4 to 2 across } k\text{-gap}} - - \underline{L}^2 - \\
 & - \underline{\text{Check 1}} \left\{ \begin{aligned}
 & - (\underline{L} - \underline{L \text{ to } beg})^2 - (\underline{ER \text{ to } end})^3 - \underline{L \text{ to } end} - \underline{R} - \underline{P} - \underline{L} - * \\
 & - \underline{R} - \underline{E} - \underline{L} - \underline{\text{Copy 3 to 3 across } k\text{-gap}} - \underline{P} - \underline{R} - \underline{\text{Copy 3-rd } n.L.} - \\
 & - \underline{\text{Copy 5 to 5 across } k\text{-gap}} - \underline{\text{Copy 6 to 6 across } k\text{-gap}} - \dots - \\
 & - \underline{\text{Copy } n+3 \text{ to } n+3 \text{ across } k\text{-gap}} - *
 \end{aligned} \right.
 \end{aligned}$$

The two '='s are connected and the two '='s are connected. The two '*'s are identified and the two '*'s are identified. The states are ordered within each instruction in the original internal order and among the various instructions in the order in which they are written. $P_1, \dots, P_m$ and the distinguished states are as in $\mathcal{G}'$; M_1 is as in $\mathcal{G}'$ (except that it is defined only on $\bigcup_{i=1}^m P_i$).

The way in which the machine computes is as follows. When it

starts, scanning the last square of α_n in the tape

$\dots s_0 \alpha_1 \dots \alpha_n s_0^2$ it writes after a gap of order k $000\alpha_1 \dots \alpha_n$

and starts computing $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$, i.e.,

$G(\hat{y}F(\varphi_1, \dots, \varphi_m, y, \alpha_2, \dots, \alpha_n)1\alpha_1, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$. If during

the computation the machine enters the state t while scanning the last

square of a numeral β then $(\hat{y}F(\varphi_1, \dots, \varphi_m, y, \alpha_2, \dots, \alpha_n)1\alpha_1)(\beta)$

has to be supplied. The machine checks whether $\beta < \alpha_1$ or not.

If not then it gives the value 0 and continues the computation; if

$\beta < \alpha_1$ then to the right of a k -gap a sequence $\beta \alpha_1 1 \beta \alpha_2 \dots \alpha_n$

is printed and the machine enters the initial state $1'$ again and it

now computes $F(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n)$. The third number in the

sequence - 1 - tells us that the machine does not do now the main

computation, which is indicated by 0, but a primary subsidiary

computation. If during the computation of a primary subsidiary computation

we need to compute $F(\varphi_1, \dots, \varphi_m, \gamma, \alpha_2, \dots, \alpha_n)$ for $\gamma < \beta$ this

subsidiary computation is called secondary and the corresponding sequence

has 2 as its third numeral, and so on. If the machine $\mathcal{J}'$ enters

the state $0'$ it means that a computation ended and then the third

number of the sequence on which this computation was done is checked.

If it is not 0 then this was a subsidiary computation, the result is

moved in to the right place and the machine enters the state $M_1(t)$

and goes on performing the computation to which the computation just

ended was subsidiary. If, on the other hand, the third member of the

sequence is 0 this means that the main computation is finished. Then again all auxiliary numerals are erased and the result of the computation is moved left.

To show that $\mathcal{M}$ computes F and that the requirements (a) - (h) are satisfied we shall prove the following. Let us denote with g^* the part of $\mathcal{M}$ obtained by omitting the part $\underline{R}^k - (\underline{R} - \underline{P} - \underline{R})^3 - (\text{Copy } n+3\text{-rd n.L.})^n$ in the beginning, so that now 1' is the initial state. We claim that if g^* starts scanning the last square of α_n in the tape $\dots \underline{\tau}^s \alpha_1 \dots \leq_n^s \alpha$ then if $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \beta$ then g^* arrives at a configuration in which it scans the last square of β on the tape $\dots \underline{\tau} \beta$ and in which g^* is in the state $M_1(t)$ if $\nu > 0$ and 0 if $\nu = 0$. If $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is not defined then the computation of g^* diverges. We prove also that requirements analogous to (a) - (h) are satisfied in this case. We shall denote the corresponding requirements in our present case by (a') - (h'). The requirement (a') was stated above; the requirement (e') is as follows: g^* does not move left of $\underline{\tau}$; whenever g^* enters the 1's in $\underline{\tau}$ it is in some internal state q where the part of the table corresponding to that state is

	0	1
q	...	R_q

(so that g^* does not change those 1's and the

state in which g^* arrives at the last square of $\underline{\tau}$ depends on q but not on $\underline{\tau}$); it is obvious what requirements (b') - (d'), (f') - (h') are.

Assume that we proved already what we stated concerning $\mathcal{Q}^*$ then we shall see that $\mathcal{F}$ computes F and satisfies (a) - (h). When $\mathcal{F}$ starts on the tape $s_0^{\lambda+1} \alpha_1 \dots \alpha_n s_0^{\Omega}$ it prints on it additional symbols and arrives at the tape $s_0^{\lambda+1} \alpha_1 \dots \alpha_n s_0^k \underline{000} \alpha_1 \dots \alpha_n s_0^{\Omega}$ on which it scans the last square of the second α_n and then it starts acting as $\mathcal{Q}^*$. If $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \beta$ then, by what we assumed, the machine $\mathcal{Q}^*$, and hence $\mathcal{F}$, finally scans the last square of β on the tape $s_0^{\lambda+1} \alpha_1 \dots \alpha_n \beta s_0^{\Omega}$ in the terminal state. If $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is not defined then the computation of $\mathcal{Q}^*$, and hence of $\mathcal{F}$, diverges. That conditions (a) - (h) hold for $\mathcal{F}$ follows easily from the corresponding requirements (a') - (h') on $\mathcal{Q}^*$, when one notes that while acting as $\underline{R}^k - (\underline{R} - \underline{P} - \underline{R})^3 - (\text{Copy } n+3\text{-rd } n.L.)^n$ $\mathcal{F}$ does not enter states in the P_i 's or distinguished states.

We shall now prove what we claimed concerning $\mathcal{Q}^*$. This will be done by induction on α_1 . We shall compare the action of $\mathcal{Q}^*$, with the given partial functions $\varphi_1, \dots, \varphi_m$ on the tape $\tau s_0^k \underline{000} \alpha_1 \dots \alpha_n s_0^{\Omega}$ with the action of $\mathcal{Q}$ with the given partial functions $\hat{y}^F(\varphi_1, \dots, \varphi_m, y, \alpha_2, \dots, \alpha_n) 1 \alpha_1, \varphi_1, \dots, \varphi_m$ on the tape $s_0^{\lambda+1} \alpha_1 \dots \alpha_n s_0^{\Omega}$. By definition of $\mathcal{Q}^*$, and since $\mathcal{Q}$ does not go left of the blank square in front of α_1 , $\mathcal{Q}^*$ acts exactly like $\mathcal{Q}$ until it enters the state t for the first time. Let us see how $\mathcal{Q}^*$ continues. Since $\mathcal{Q}$ is assumed to satisfy (d), when it enters the state

t the tape is $\dots s_0 \gamma s_0^{\Omega}$ and $\mathcal{Q}$ scans the last square of γ .
 Therefore the tape of $\mathcal{Q}^*$ in the same time is also $\dots s_0 \gamma s_0^{\Omega}$
 and $\mathcal{Q}^*$ also scans the last square of γ in the state t . $\mathcal{Q}^*$
 now adds symbols to get the tape $\dots s_0 \gamma s_0^k \alpha_1 s_0^{\Omega}$. To see
 that the last numeral is indeed α_1 note that since $\beta(G) = k - 1$
 and $\mathcal{Q}$ satisfies requirement (f) and in $\mathcal{Q} \ t \in P_0$ there are no
 k -gaps between the gap succeeding $\underline{\gamma}$ and the gap succeeding γ and
 hence the fourth numeral to the right of the last k -gap is just α_1 .
 The next action of $\mathcal{Q}^*$ is to act as $<$ comparing γ and α_1 . If
 $\gamma \neq \alpha_1$ then $\mathcal{Q}^*$ erases the second γ and α_1 and changes the tape
 to $\dots \gamma \underline{0} s_0^{\Omega}$ and then it scans the last square of γ in the state
 $M_1(t)$. If $\gamma < \alpha_1$ we get the tape $\dots \gamma s_0^k \alpha_1 \gamma + 1 \alpha_2 \dots \alpha_n s_0^{\Omega}$
 which $\mathcal{Q}^*$ scans in the state $1'$. Since $\gamma < \alpha_1$ we can employ our
 induction hypothesis concerning $\mathcal{Q}^*$. If $F(\varphi_1, \dots, \varphi_m, \gamma, \alpha_2, \dots, \alpha_n) = \delta$
 then finally $\mathcal{Q}^*$ enters the state $M_1(t)$ while the tape is $\dots \gamma \delta s_0^{\Omega}$
 and $\mathcal{Q}^*$ scans the last square of γ . Nothing to the left of γ was
 changed by our induction hypothesis on $\mathcal{Q}^*$. Thus in both cases (i.e.,
 $\gamma \neq \alpha_1$ and $\gamma < \alpha_1$) if $(\hat{\gamma} F(\varphi_1, \dots, \varphi_m, \gamma, \alpha_2, \dots, \alpha_n) 1 \alpha_1) (\gamma)$
 is defined and equals δ what $\mathcal{Q}^*$ does is to get the tape
 $\dots \gamma \delta s_0^{\Omega}$ and scan the last square of γ in the state $M_1(t)$.
 When $\mathcal{Q}$ enters the state t , since $t \in P_0$, the same numeral is now
 added on the tape and $\mathcal{Q}$ enters the same state $M_1(t)$ while scanning
 the same square. If $\gamma < \alpha_1$ and $F(\varphi_1, \dots, \varphi_m, \gamma, \alpha_2, \dots, \alpha_n)$ is
 undefined then by the induction hypothesis the computation of $\mathcal{Q}^*$

diverges when it starts with the tape $\dots \gamma s_0^k \gamma \alpha_1 \gamma \alpha_2 \dots \alpha_n$.

Thus the computation of $\mathcal{T}$ diverges. Also the computation of

$G(\hat{\gamma}F(\varphi_1, \dots, \varphi_m, \gamma, \alpha_2, \dots, \alpha_n)1\alpha_1, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ by $\mathcal{Q}^*$

diverges (since $\hat{\gamma}F(\varphi_1, \dots, \varphi_m, \gamma, \alpha_2, \dots, \alpha_n)1\alpha_1$ is undefined for γ)

and hence $G(\hat{\gamma}F(\varphi_1, \dots, \varphi_m, \gamma, \alpha_2, \dots, \alpha_n)1\alpha_1, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$

is undefined, i.e., $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is undefined. If this

divergence does not happen then $\mathcal{Q}$ and $\mathcal{Q}^*$ operate alike again until

the internal state t is reached again in which case the same situation

arises as before. We can now easily show, by induction on the ordinal

number of the entrance in the state t , that in each such entrance either

the computation of $\mathcal{Q}^*$ with the corresponding tape $\dots \gamma s_0^{\mathcal{Q}}$ diverges

and then $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is not defined, or otherwise after

$\mathcal{Q}^*$ returns to the state $M_1(t)$ and the tape is $\dots \gamma s_0^{\mathcal{Q}}$ it

continues exactly like $\mathcal{Q}$. (till the state t is encountered again).

How do the configurations of $\mathcal{Q}$ and $\mathcal{Q}^*$ compare after a limit number

of entries of $\mathcal{Q}$ in the state t ? The tapes, from $\alpha_1 \dots \alpha_n$

and rightwards have the same sequence of symbols printed in them, because

the computation of $\mathcal{Q}^*$ is like that of $\mathcal{Q}$ only that some additional

subsidiary computations are carried out by $\mathcal{Q}^*$ in parts of the tape

which are blank in $\mathcal{Q}$ (by requirement (d) on $\mathcal{Q}$) and are erased

when terminated and hence do not effect the tape after a limit number of configurations. $\underline{\eta}, \underline{\tau}, \underline{\nu}, \underline{\alpha}_1, \dots, \underline{\alpha}_n$ are copied, but this does also not effect the tape eventually since the 1's are changed to 2's and then again to 1's, before the next entry in the state t . After a limit number of entries in the state t $\mathcal{Q}^*$ was frequently in at least one state of $\mathcal{Q}$, namely t , and $\mathcal{Q}$ and $\mathcal{Q}^*$ enter the same states of $\mathcal{Q}$ except for an exception which will be noted later and which will be seen not to effect the outcome. Since all the states of $\mathcal{Q}$ precede the additional states of $\mathcal{Q}^*$ in the order of the states in $\mathcal{Q}^*$ the $\liminf$ of the states in which $\mathcal{Q}^*$ was is the same as the $\liminf$ of the states in which $\mathcal{Q}$ was. Therefore $\mathcal{Q}$ and $\mathcal{Q}^*$ are in the same state following the limit number of entries in t . By requirement (g), which is satisfied by $\mathcal{Q}$, this state which $\mathcal{Q}$ and $\mathcal{Q}^*$ enter following the limit number of entries in t is a distinguished state and hence is in $\mathcal{Q}$ the initial state of an instruction $\underline{\text{End}}_l$, where $l \geq k$. By requirement (f) on $\mathcal{Q}$ the tape of $\mathcal{Q}$ has no k -gaps on it and therefore the tape of $\mathcal{Q}^*$, to the right of $\underline{\tau}$ has only the single k -gap s_o^k between $\underline{\tau}$ and $\underline{\eta}$. Since $\mathcal{Q}$ and $\mathcal{Q}^*$ obey now the same instruction $\underline{\text{End}}_l$, and the parts of the tapes to the right of the squares scanned contain no $k+1$ -gaps they both move to the first free square and get there in the same state without changing anything on the tape. It is possible that while carrying out the instruction $\underline{\text{End}}_l$ $\mathcal{Q}^*$ entered states of $\underline{\text{R to end}}$ (which is a part of $\underline{\text{End}}_l$) whereas $\mathcal{Q}$ did not, but this does

not effect the state of $\mathcal{Q}^*$ after a limit number of such cases since the states of Check l+1-gap precede those of R to end in End₁, and both $\mathcal{Q}$ and $\mathcal{Q}^*$ entered the states of Check l+1-gap. We do also notice that after $\mathcal{Q}$ went through a limit number $\{ < \Omega$ of atomic acts such that all the subsidiary computations done by $\mathcal{Q}^*$ at the same time did not diverge, the number of atomic acts done by $\mathcal{Q}^*$ is also $< \Omega$ since except for the acts done by $\mathcal{Q}$ $\mathcal{Q}^*$ carried out $< \Omega$ subsidiary computations of $< \Omega$ acts each and possibly additional moves after it entered distinguished states as above. Since Ω is regular the total number of atomic acts by $\mathcal{Q}^*$ is still $< \Omega$. When $\mathcal{Q}$ enters the terminal state 0 then by (a) the tape of $\mathcal{Q}$ is $s_0^{\lambda+1} \alpha_1 \dots \alpha_n \beta s_0^{\Omega}$ with $\mathcal{Q}$ scanning the last square of β , where $\beta = F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$, and then $\mathcal{Q}^*$ has the tape $\tau s_0^k \pi \sigma \gamma \alpha_1 \dots \alpha_n \beta s_0^{\Omega}$ and it scans the last square of β . By definition of $\mathcal{Q}^*$ the machine checks γ , and then it erases $\pi \sigma \gamma \alpha_1 \dots \alpha_n$ and moves β leftwards closing the gap between it and τ and it scans the last square of β on the tape $\dots \tau \beta s_0^{\Omega}$ in the state 0 or $M_1(t)$ according to whether $\gamma = 0$ or $\gamma > 0$.

We shall now see that $\mathcal{Q}^*$ satisfies requirements (a') - (h').

(a') holds as is already proved above.

(b') We have to prove that in each configuration there is only a finite number of blocks of symbols to the right of τ . While $\mathcal{Q}^*$ acts as $\mathcal{Q}$

(b') holds because $\mathcal{Q}$ satisfies (b). While $\mathcal{Q}^*$ carries out a subsidiary computation (b') holds because $\mathcal{Q}$ satisfies (b) and by the induction hypothesis, on α_1 , $\mathcal{Q}^*$ satisfies (b') within the subsidiary computation. During the additional actions of $\mathcal{Q}^*$, when it carries out the instructions $\leftarrow$, Copy, Move, etc., only a finite number of blocks of symbols is added, as noted where those instructions were introduced.

(c') holds by the construction of $\mathcal{Q}^*$.

(d') holds for $\mathcal{Q}^*$ because $\mathcal{Q}$ satisfies (d) and because (d') holds for the subsidiary computations of $\mathcal{Q}^*$ by the induction hypothesis.

(e') When $\mathcal{Q}^*$ operates as $\mathcal{Q}$ it does not go left of the s_0 in front of α_1 , by requirement (e) on $\mathcal{Q}$. When it enters the state t then while $\gamma, \alpha_1, \dots, \alpha_n$ are copied $\mathcal{Q}^*$ does not go left of σ .

When the instructions of the type Copy i to j across k-gap are carried out it is seen by looking at the definition of that instruction that $\mathcal{Q}^*$ does not go left of τ and when $\mathcal{Q}^*$ enters the 1's of τ it is in the state 3 of L to end. The line in the table corresponding to this

state is

0	1
3	R3

 . During a subsidiary computation when

$\mathcal{Q}^*$ is applied to the tape $\dots \gamma s_0^k \gamma \alpha_1 \gamma \alpha_2 \dots \alpha_n$ it does not go left of γ by the induction hypothesis, while γ itself cannot be left of α_n because $\mathcal{Q}$ satisfies requirements (d) and (e).

Finally, $\mathcal{Q}^*$ can enter the 1's of τ after a limit number of subsidiary computations, but then it is in a distinguished state of $\mathcal{Q}$ by requirement

(g) on $\mathcal{Q}$, and hence $\mathcal{Q}^*$ is in the initial state of an instruction End₁, which is the initial state of Check l+1-gap. The corresponding

line in the table is
$$\begin{array}{c|cc} & 0 & 1 \\ \hline 1 & \dots & R1 \end{array} .$$

(f') holds since $\mathcal{Q}$ satisfies requirement (f) and $\mathcal{Q}^*$ satisfies (f') for the subsidiary computations by the induction hypothesis.

(g') If ζ is a limit number such that during ζ consecutive

configurations $\mathcal{Q}^*$ was frequently in $\bigcup_{i=1}^m P_i$ then we consider

two cases. (i) For some $\eta < \zeta$ from the η -th configuration on $\mathcal{Q}^*$ was carrying out a single subsidiary computation, then, following the configurations $\mathcal{Q}^*$ is in a distinguished state since by the induction hypothesis (g') holds for the subsidiary computation. (ii) If (i) does not hold the $\mathcal{Q}^*$ was frequently in the states of

$\bigcup_{i=1}^m P_i$ during the main computation or, otherwise, $\mathcal{Q}^*$ carried out

frequently subsidiary computations, so that the machine $\mathcal{Q}$ was frequently in the state P_0 . Anyway the machine $\mathcal{Q}$, which we compare with $\mathcal{Q}^*$,

was frequently in $\bigcup_{i=0}^m P_i$ and hence, by (f), following the ζ acts

of $\mathcal{Q}^*$, $\mathcal{Q}$ is in a distinguished state, and hence $\mathcal{Q}^*$ is also in the same distinguished state.

(h') follows immediately from our definition of $\mathcal{Q}^*$.

$$(8) \quad F(f_1, \dots, f_m, x_1, \dots, x_n) = (\mu x_{n+1}) G(f_1, \dots, f_m, x_1, \dots, x_n, x_{n+1}).$$

Let $\mathcal{O}$ be a finitary machine which computes G and which satisfies requirements (a) - (g) and also requirement (h), where k is a finite number $k \geq \varphi(F) = \varphi(G)$. F is computed by the following machine $\mathcal{F}$.

$$\underline{R} - \underline{P} - \underline{R} - * - \mathcal{O} - \underline{L} \text{ to beg} - \underline{R} - \underline{\text{Check 1}} \left\{ \begin{array}{l} - \underline{L}^2 - \underline{P} - \underline{R} - \underline{\text{ER to end}} - \underline{\text{End}_k} - * \\ - \underline{L} - \underline{E} - \underline{L} - 0. \end{array} \right.$$

The states are ordered so that the first state is the initial state of $\underline{\text{End}_k}$ and the internal order of the states of $\mathcal{O}$ remains unchanged. The P_i 's are those of $\mathcal{O}$; the distinguished states are those of $\mathcal{O}$ with the addition of the initial state of $\underline{\text{End}_k}$.

The action of $\mathcal{O}$ is as follows. When $\mathcal{O}$ starts with the tape $s_0^{\lambda+1} \alpha_1 \dots \alpha_n s_0^{\mathcal{Q}}$ it adds a $\underline{0}$ to get $s_0^{\lambda+1} \alpha_1 \dots \alpha_n \underline{0} s_0^{\mathcal{Q}}$ and starts computing G for $\alpha_1, \dots, \alpha_n, 0$. If there is a result and it is not 0 then $\mathcal{F}$ adds another 1 to $\underline{0}$ and starts computing G for $\alpha_1, \dots, \alpha_n, 1$ and so on. After a limit number of computations like these, where all the computations resulted in numbers > 0 , the tape is $s_0^{\lambda+1} \alpha_1 \dots \alpha_n \gamma s_0^{\mathcal{Q}}$. The total number of atomic acts done by $\mathcal{F}$ is still $< \mathcal{Q}$, by the regularity of $\mathcal{Q}$. $\mathcal{F}$ moves now to the last square of $\mathcal{F}$ and starts computing G for $\alpha_1, \dots, \alpha_n, \gamma$. If for some $\delta < \mathcal{Q}$ the computation of G for $\alpha_1, \dots, \alpha_n, \delta$ results in 0 then $\mathcal{F}$ erases the $\underline{0}$ and stops in the last square of, δ in the tape $s_0^{\lambda+1} \alpha_1 \dots \alpha_n \delta s_0^{\mathcal{Q}}$. If for some $\delta < \mathcal{Q}$ $G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, \gamma)$ is defined and is not 0 for every $\gamma < \delta$ and $G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, \delta)$

is undefined then the computation of $\mathcal{F}$ diverges while it computes G for $\alpha_1, \dots, \alpha_n, \delta$. If for every δ $G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, \delta)$ is defined and is not 0 then the computation of $\mathcal{F}$ diverges since $\mathcal{F}$ goes through Ω computations without entering the terminal state.

It is easy to prove that requirements (a) - (h) are satisfied by $\mathcal{F}$ using the assumption that they are satisfied by $\mathcal{G}$.

We shall now define an instruction Print α for every number α , which will instruct the machine to print α on the tape. Since there are only $\aleph_0$ finitary instructions and there are Ω numbers α the instruction Print α will not be finitary in general, but it will be always numerical.

19) Print α

This is an ordered instruction which prints a numeral α beginning in a free square which the machine scans. The machine stops in the last square of α . The machine is defined as follows.

		0	1
$1 \leq \pi < 2 + \alpha$	π	$PR\pi + 1$	
	$2 + \alpha$	0	

The states are ordered according to their indices. Since the ordinals and their addition are

absolute with respect to the class of the constructible sets it is easily seen that the function M_0 is constructible and hence the instruction Print α is numerical.

20) Print _d α

This is a discrete instruction doing the same thing as Print _o α above. We shall first define it in the case that $\alpha = \omega^\beta$, $\beta \geq 0$.

In this case it is given as follows.

		0	1
$1 \leq \eta < 2 + \alpha$	η	PR2	
$2 + \alpha$		0	

The function M_2 of the instruction is defined by

$$M_2(\gamma - \{0, 1\}) = \gamma \quad \text{for } 2 < \gamma \leq 2 + \beta$$

$$M_2(a) = 0^* \quad \text{otherwise.}$$

We use 0^* to denote the absolute terminal state, i.e., whenever the instruction Print _d α is a part of a machine the state 0^* of Print _d α is identified with the terminal state of the machine (whereas the state 0 of the instruction is in general identified with some other state of the machine). As is well known for every ordinal number $\zeta \geq 1$ there is a unique finite number $k > 0$ and a unique sequence $\gamma_1 \geq \gamma_2 \geq \dots \geq \gamma_k$ of ordinal numbers such that $\zeta = \sum_{i=1}^k \omega^{\gamma_i}$. We shall say that γ_k

is the index of ζ . We shall now prove by induction on $\zeta \leq \alpha$ that at the $1 + \zeta$ -th configuration (where the counting of the configurations starts with 0) the tape is $\dots \zeta s_0^\omega$, where ζ begins in the square which the machine scanned in the beginning, the machine scans the last square of ζ and the state of the machine is $2 +$ the index of ζ . For $\zeta = 0$ it is obvious. If $\zeta = \xi + 1$ then by the induction hypothesis in the $1 + \xi$ -th configuration the tape is $\dots \xi s_0^\omega$ and the machine is scanning the last square of ξ . Then the machine prints,

moves one square right and enters the state 2. Thus the tape becomes $\dots \underline{\xi} s_0^{\Omega}$, with the machine scanning the last square of $\underline{\xi}$ and the state is $2 + \text{the index of } \xi$, i.e., $2+0$. If ξ is a limit number then for every $\xi < \zeta$ the tape at the $1+\xi$ -th configuration is $\dots \underline{\xi} s_0^{\Omega}$, with the machine scanning the last square of ξ . Therefore at the $1+\zeta$ -th configuration the tape is $\dots \underline{\zeta} s_0^{\Omega}$ and the machine scans the last square of $\underline{\zeta}$. Let γ be the index of ζ , $\gamma \leq \beta$. Since $1+\zeta (= \zeta)$ is a limit of ordinals of index γ if and only if $\gamma < \gamma$ therefore, by the induction hypothesis, the set of the states frequented by the machine during the $1+\zeta$ configurations is $(2+\gamma) - \{0,1\}$. By the definition of M_2 the state of the machine in the $1+\zeta$ -th configuration is $2+\gamma$. By what just proved, the $1+\alpha$ -th configuration of the machine consists of the tape $\dots \underline{\alpha} s_0^{\Omega}$ in which the machine scans the last square of $\underline{\alpha}$ in the state $2+\beta$. In the next configuration the machine is in the state 0, but its position and tape remain unchanged.

Let α be now an arbitrary ordinal. If α is finite then we put $\text{Print}_d \alpha = (\text{Print}_d 0)^{\alpha+1} = 0$. If α is infinite then by the theorem quoted above there is a unique finite number $k > 0$ and a unique

sequence $\gamma_1 \geq \gamma_2 \geq \dots \geq \gamma_k$ of ordinals such that $\alpha = \sum_{i=1}^k \omega^{\gamma_i}$. We

define $\text{Print}_d \alpha$ by $\text{Print}_d \omega^{\gamma_1} - \text{Print}_d \omega^{\gamma_2} - \dots - \text{Print}_d \omega^{\gamma_k} = 0$,

where the function M_2 is defined like the M_2 of each instruction for sets of states of that instruction and 0^* otherwise.

It is again easy to see, by the absoluteness of (ordinal) powers, that

each instruction $\text{Print}_d \omega^?$ is numerical and hence each machine $\text{Print}_d \alpha$, which is a composition of finitely many such machines is numerical too.

THEOREM 3. If Ω is regular then every numerically recursive functional is computable by both an ordered and a discrete numerical Turing machine with $\Lambda = 3$ such that requirements (a) - (d) of §3 are satisfied.

PROOF. Every n.r.f. $F(f_1, \dots, f_m, x_1, \dots, x_n)$ is obtained from some f.r.f. $G(f_1, \dots, f_m, x_1, \dots, x_{n+k})$, $k \geq 0$, by substituting numbers $\alpha_1, \dots, \alpha_k$ for the variables $x_{i_1}, \dots, x_{i_k}$, respectively, where $1 \leq i_1 < i_2 < \dots < i_k \leq n+k$, and renaming the other variables, which we shall denote with $x_{j_1}, \dots, x_{j_n}$ by $x_1, \dots, x_n$, respectively. If $k = 0$ then F is finitarily recursive and hence satisfies the conclusion of Theorem 2 and, a fortiori, that of Theorem 3. If $k > 0$ let $H(f_1, \dots, f_m, x_1, \dots, x_{n+k})$ be function $G(f_1, \dots, f_m, x_1, \dots, x_{n+k})$ with $p^{(n+k, r)}$ substituted for x_{j_r} , $1 \leq r \leq n$, and $p^{(n+k, n+r)}$ substituted for x_{i_r} , $1 \leq r \leq k$. H is a f.r.f. by (6) and (1). By the definition of H we have $F(f_1, \dots, f_m, x_1, \dots, x_n) = H(f_1, \dots, f_m, x_1, \dots, x_n, \alpha_1, \dots, \alpha_k)$. Let $\mathcal{N}$ be a finitary Turing machine which computes H and satisfies (a) - (d). Let $\mathcal{F}$ be the machine given by

$$\begin{aligned} & \underline{R} - \underline{\text{Print}} \alpha_1 - \underline{R} - \underline{\text{Print}} \alpha_2 - \dots - \underline{R} - \underline{\text{Print}} \alpha_k - \mathcal{H} - \underline{L \text{ to beg}} - \\ & - (\underline{L} - \underline{L \text{ to beg}})^k - \underline{R} - \underline{\text{Check 1}} \left\{ - \underline{\text{ER to end}} - (\underline{\text{ER to end}})^{k-1} - \right. \end{aligned}$$

- Move next n. on top - 0.

The instructions $\underline{\text{Print}} \alpha_r$ are $\underline{\text{Print}}_0 \alpha_r$ for all $1 \leq r \leq k$ or $\underline{\text{Print}}_d \alpha_r$ for all $1 \leq r \leq k$. In the first case we order the states of $\mathcal{J}$ so that the order of the states of each instruction is unchanged and the states of different instructions are ordered in the same order in which the instructions are written, and then $\mathcal{J}$ is an ordered machine. In the second case we take the function M_2 to be the union of the various M_2 's, with M_2 defined as 0 for sets of states which contain states of more than one instruction (note that for a finitary ordered instruction M_2 of a non-void set of states is defined as its least member in the ordering), and then $\mathcal{J}$ is a discrete machine. In either case $P_1, \dots, P_n$ and the function M_1 are exactly those of $\mathcal{H}$. $\mathcal{J}$ is numerical since it is a composition of finitely many numerical instructions.

For every (bounded) eventually vanishing partial function φ we define the following instructions.

21) Output $_0 \varphi$

This is an ordered machine $\mathcal{J}$ such that if the tape is $\dots s_0 \alpha s_0^2$ and $\mathcal{J}$ scans the last square of α then if $\varphi(\alpha) = \beta$ $\mathcal{J}$ finally enters the terminal state while it scans the last square of β in the

tape $\dots s_{\alpha} \beta s_{\alpha}^{\Omega}$, and if $\varphi(\alpha)$ is undefined then the computation of $\mathcal{J}$ diverges.

Let γ be the least ordinal such that $\varphi(\alpha) = 0$ for every $\gamma \leq \alpha < \Omega$.

$\mathcal{J}$ will be the following machine.

$$\underline{L \text{ to beg}} - \underline{R} - \mathcal{H}_0^{(\gamma)} \begin{cases} \pi^* \underline{\text{Print}}_0 \varphi(\pi) - 0. & \text{If } \pi < \gamma \text{ and } \varphi(\pi) \text{ is defined.} \\ \pi^* \underline{\text{Diverge}} & \text{If } \pi < \gamma \text{ and } \varphi(\pi) \text{ is undefined} \\ \gamma^* \underline{\text{Print}}_0 0 - 0. & \end{cases}$$

where $\mathcal{H}_0^{(\gamma)}$ is the ordered machine given by the table

		0	1	
$0 \leq \pi < \gamma$	$1+\pi$	π^*	$R \ 1+\pi+1$	where $\pi^* = 1+\gamma+1+\pi$ for $0 \leq \pi < \gamma$. For $\pi < \gamma$ if $\varphi(\pi)$ is defined then π^*
	$1+\gamma$	γ^*	γ^*	

is the initial state of an instruction $\underline{\text{Print}}_0 \varphi(\pi)$; if $\varphi(\pi)$ is undefined then $\varphi(\pi)$ is an initial state of an instruction $\underline{\text{Diverge}}$

which is given by the table $\begin{array}{c|c} 0 & 1 \\ \hline 1 & 1 \end{array}$. γ^* is the initial

state of an instruction $\underline{\text{Print}}_0 0$. The order of the states is as follows. First come the states of $\underline{L \text{ to beg}}$, then those of $\underline{R}$, then the states of $\mathcal{H}_0^{(\gamma)}$ ordered by the natural order of the ordinals, then the non-initial and non-terminal states of the instructions $\underline{\text{Print}}_0 \varphi(\pi)$ ordered according to π , and for the same π according to their internal order, then come the non-initial states of $\underline{\text{Print}}_0 0$ in their internal order. It is easily seen that $\mathcal{J}$ has $< \Omega$ states and that it indeed performs what we claim it does.

We want now to construct a discrete machine Output_d φ like Output_o φ . It is quite clear that this can be done along the same lines as Output_o φ if we can construct a discrete machine $\mathcal{H}_d^{(\gamma)}$ which "reads" numerals the way $\mathcal{H}_o^{(\gamma)}$ does. We shall see that this can be done. First we shall construct for every number δ a discrete instruction which will find out the index of every ordinal $\alpha < \omega^{\delta+1}$, which is given by $\underline{\alpha}$, where we take the index of 0 to be -1.

22) Find index^(f)

	0	1	
$0 \leq \pi \leq \delta$	1	-1'	R2
	2+ π	π'	R2

$M_2(\gamma - \{0,1\}) = \gamma \quad 3 \leq \gamma \leq \delta$
 $M_2(a) = 0^*$ otherwise.

-1' and π' , $0 \leq \pi \leq \delta$ are terminal states. It is easy to see that if the machine starts its operation scanning the second square of $\underline{\alpha}$, where $\alpha < \omega^{\delta+1}$ it will end scanning the last square of $\underline{\alpha}$ in the state β' , where β is the index of α .

If the index of α is β then there is a least ordinal $\sigma \geq 0$ such that $\sigma + \omega^\beta = \alpha$. We shall now construct, for $\beta > 0$, a discrete machine which when it starts on the first square of a numeral $\underline{\alpha}$ such that the index of α is β it erases ω^β 1's at the end of $\underline{\alpha}$ so as to leave $\underline{\sigma}$ on the tape.

23) Erase ω^β at end ($\beta > 0$)

	0	1	
		R2	$M_2(\gamma - \{0,1\}) = \gamma \quad 3 \leq \gamma \leq 2+\beta$
$2 \leq \pi < 2+\beta$	1	ER2	$M_2(2+\beta+1 - \{0,1\} \cup \{1''\}) = 2+\beta+1$
or $\pi = 2+\beta+1$	π	ER2	$M_2(2+\beta+2 - \{0,1\} \cup \{1''\}) = 2+\beta+1$
	$2+\beta$	L1''	$M_2(a) = 0^* \quad \text{otherwise.}$
	1''	PR1''	ER2

The machine acts as follows. When it starts on the first square of α (where α is of index β) it moves right to the second square and erases ω^β 1's. Then it checks the next square on the right. If that square contains 1 then the machine moves left by a single move left and arrives at the first erased square since it was on a limit square. Then the machine prints in all the squares it erased, arrives at the first erased square since it was on a limit square. Then the machine prints in all the squares it erased, arrives at the first square it did not erase and starts erasing again, and vice versa. If the machine encounters a 0 immediately after it erased ω^β 1's it stops in that square in the terminal state.

24) Output $_d \varphi$

As we remarked we have to construct a machine $\mathcal{H}_d^{(r)}$ which "reads" numerals of ordinals $< \gamma$. We shall construct $\mathcal{H}_d^{(r)}$ which will do the same thing as $\underline{L}$ to beg - $\underline{R} - \mathcal{H}_o^{(r)}$. Let δ be the least ordinal such that $\gamma < \omega^\delta$. $\mathcal{H}_d^{(r)}$ will contain, for every ordinal $\tau < \alpha$ a state $\bar{\tau}$ which will be the initial state of an instruction

$$\bar{\tau} = \underline{L} \text{ to } \underline{\text{beg}} - \underline{R} - \underline{\text{Find index}}^{(\delta)} \begin{cases} -1' & \underline{\text{Fill in}} - \tau^* \\ 0' & \underline{L} - \underline{E} - \overline{1+\tau} \\ \pi' & \underline{\text{Erase } \omega^\pi} \text{ at end} - \overline{\omega^\pi + \tau} \end{cases} \quad 0 < \pi < \delta$$

where Fill in is the instruction

	0	1
1	PR1	EL2
2		EO

. $\mathcal{H}_d^{(\tau)}$ begins

with, i.e., its initial state is, the initial state of the instruction $\underline{R} - \underline{P} - \underline{R} - \underline{O}$. M_2 is the union of the M_2 's of the various instructions, with M_2 defined as O^* for sets of states which contain states of more than one instruction.

One can easily verify that if $\mathcal{H}_d^{(\tau)}$ starts on the last square of α in the tape $\dots s_0 \alpha s_0^R$, where $\alpha < \gamma$, it will end in the same square in the state α^* with the tape unchanged.

Output_d φ is the following machine.

$$\underline{\text{Print}}_d \left[\begin{array}{l} \underline{L} - \underline{L} \text{ to } \underline{\text{beg}} - (\underline{ER} \text{ to } \underline{\text{end}})^2 - \underline{L} \text{ to } \underline{\text{end}} - \underline{R} - \underline{P} - \underline{R} - \underline{O} \\ \underline{L} - \underline{L} \text{ to } \underline{\text{beg}} (\underline{ER} \text{ to } \underline{\text{end}})^2 - \underline{L} \text{ to } \underline{\text{end}} - \end{array} \right]$$

$$-\mathcal{H}_d^{(\tau)} \begin{cases} \pi^* & \underline{\text{Print}}_d \varphi(\pi) = 0 & \text{if } \pi < \gamma \text{ and } \varphi(\pi) \text{ is defined} \\ \pi^* & \underline{\text{Diverge}} & \text{if } \pi < \gamma \text{ and } \varphi(\pi) \text{ is undefined} \end{cases}$$

It is obvious what M_2 is. It is easily seen that Output_d φ does exactly the same thing as Output_o φ .

THEOREM 4. If $\mathcal{Q}$ is regular then every general recursive functional is computable by both an ordered and a discrete Turing machine with $\Lambda = 3$

and such that requirements (a) - (d) of §3 are satisfied.

PROOF. Every g.r.f. $F(f_1, \dots, f_m, x_1, \dots, x_n)$ is obtained from some f.r.f. $G(f_1, \dots, f_{m+k}, x_1, \dots, x_n)$ by substituting b.e.v.f.'s $\varphi_1, \dots, \varphi_k$ for $f_{m+1}, \dots, f_{m+k}$, respectively. We shall not even assume that $\varphi_1, \dots, \varphi_k$ are total functions. Let $\mathcal{O}$ be a finitary machine which computes G and satisfies requirements (a) - (h).

Let the set P_i of $\mathcal{O}$ be 0 or $\{t_i\}$. First we shall show how to construct an ordered machine $\mathcal{I}$ which computes F . Let $\mathcal{I}$ be like $\mathcal{O}$ except that it will not have sets P_i for $m < i \leq m+k$ and that k new instructions Output $\varphi_i - L - L$ to beg., $1 \leq i \leq k$, will be added. The initial state of Output φ_i will be t_{m+i} and its terminal state $M_1(t_{m+i})$ (if $P_{m+i} = 0$ the i -th instruction can be omitted). The order of the states of $\mathcal{I}$ will be the same as those of $\mathcal{O}$ with all the new states added at the end. States of different instructions will be ordered by the index i and the internal order of the states of each instruction will remain unchanged.

In order to see that $\mathcal{I}$ computes F we have to compare the action of $\mathcal{I}$, with the given partial functions $\varphi_1, \dots, \varphi_m$, when it starts scanning the last square of α_n on the tape $s_0^{\lambda+1} \alpha_1 \dots \alpha_n s_0^{\lambda}$ to the action of $\mathcal{O}$ with the given partial functions $\varphi_1, \dots, \varphi_m, \varphi_1, \dots, \varphi_k$ when it starts scanning the same square on the same tape. Obviously the only time the configurations can differ is when the machines enter one of the states t_{m+i} , $1 \leq i \leq k$, and following a limit number of such entries. When both machines enter a state t_{m+i} they are scanning the

last square of a numeral α on a tape $\dots s_0 \alpha s_0^Q$, by requirement (d) on $\mathcal{Q}$. If $\varphi_1(\alpha)$ is undefined then both machines diverge, each in its own way. If $\varphi_1(\alpha) = \beta$ then both machines transform the tape to $\dots s_0 \alpha \beta s_0^Q$ and return to the same square. We note that while $\mathcal{F}$ carries out Output $\varphi_1 - L - L$ to beg it does not move left of the s_0 in front of α and does not erase any symbol during its operation. Following a limit number of entries in the states t_{m+1} the machines get again in the same configuration - this is proved like the proof of the corresponding statement with respect to $\mathcal{Q}$ and $\mathcal{Q}^*$ in part (7) of the proof of Theorem 2. One can easily verify that requirements (a) - (h) are satisfied by $\mathcal{F}$, after one sees that analogous requirements are satisfied by Print α and Output φ .

A discrete machine $\mathcal{F}$ which does the same thing as the ordered machine $\mathcal{F}$ constructed above is constructed similarly, using Output φ_1 instead of Output φ_1 . However, we must pay attention to one additional point. When Output φ_1 is applied on the last square of α on the tape $\dots s_0 \alpha s_0^Q$ and $\varphi_1(\alpha) = \beta$ then it transforms the tape to $\dots s_0 \alpha \beta s_0^Q$ and returns to the same square, but during this operation some of the 1's in α are erased and then printed again. This does not effect the outcome of a single such operation, but if a 1 in α is erased and printed again a limit number of times the square becomes blank immediately after all these changes. Therefore we have to put an additional requirement on $\mathcal{Q}$.

If during the computation by $\mathcal{Q}$ of G we have a configuration in

which $\mathcal{Q}$ is in a state in one of the P_1 's and $\mathcal{Q}$ scans the last square of α on the tape $\dots s_0 \alpha s_0^Q$ then we shall say that $\mathcal{Q}$ marks the 1's in α . Each of the 1's in α is called a marked symbol and retains this attribute in later configurations until it is erased. If a new 1 is then printed in the same square it is not marked when it is printed, but it can become marked later as above. The additional requirement on $\mathcal{Q}$ which we need is:

- (i) During the computation the machine does not mark any symbol which is already marked.

If the machine $\mathcal{Q}$ indeed satisfies requirement (i) then if a symbol 1 is erased and printed a limit number of times in the same square by a limit number of actions of the instructions Output d φ_1 then when we compare the action of $\mathcal{F}$ to that of $\mathcal{Q}$ we see that $\mathcal{Q}$ also erased and printed the symbol a limit number of times in that square, because $\mathcal{Q}$ marked the symbol in that square a limit number of times. Therefore the tapes of $\mathcal{F}$ and $\mathcal{Q}$ will be the same following these actions.

To see that for every f.r.f. there is a finitary Turing machine which computes it and satisfies requirements (a) - (i) we check the proof of Theorem 2 and see that requirement (i) can be added. It is obvious in the cases (1) - (5). In cases (6), (7) it follows easily from the induction hypothesis (that the machines $\mathcal{Q}, \mathcal{N}_1, \dots, \mathcal{N}_l$ satisfy (i)). In case (8) we have to construct a machine somewhat different from the one we constructed in the proof of Theorem 2.

Let $F(f_1, \dots, f_n, x_1, \dots, x_n) = (\mu x_{n+1})G(f_1, \dots, f_n, x_1, \dots, x_n, x_{n+1})$.

Let $\mathcal{Q}$ be a finitary machine which computes G and satisfies (a) - (i).

Then F is computed by the following machine,

$$\begin{aligned} \underline{R} - \underline{P} - \underline{R} - * - (\underline{\text{Copy } n+1\text{-st } n.L.})^{n+1} - \mathcal{Q} - \underline{L} \text{ to } \underline{\text{beg}} - \underline{R} - \\ - \underline{\text{Check } 1} \left\{ \begin{array}{l} - \underline{L} - (\underline{L} - \underline{L} \text{ to } \underline{\text{beg}})^n - \underline{L} - \underline{P} - \underline{R} - (\underline{\text{ER to end}})^{n+1} - \underline{\text{End}}_k - * \\ - \underline{L} - (\underline{L} - \underline{L} \text{ to } \underline{\text{beg}})^n - (\underline{\text{ER to end}})^{n+1} - \underline{L} \text{ to } \underline{\text{end}} - 0. \end{array} \right. \end{aligned}$$

where all other details are as for the machine constructed in part (8) of the proof of Theorem 2, and the same comments apply. It is easily seen that $\mathcal{J}$ satisfies the requirements (a) - (i).

We shall now see to what extent the third symbol s_2 was needed in order to get theorems 2-4. s_2 was used in the two copying instructions Copy r-th n.L. and Copy i to j across k-gap. If a numeral has to be copied only a finite number of times we could do it by means of instructions which are like the copying instructions mentioned above, except that they use s_0 instead of s_2 . However, if we copy the same numeral a limit number of times we have to use s_2 's and not s_0 's, since if we use s_0 's the numeral copied will be erased after it is being copied a limit number of times. The only place in the proofs of Theorems 2-4 where s_2 's were used not within a copying instruction is in the construction of the machine $\mathcal{C}$ (in part (4) of the proof of Theorem 2). However, it can be seen that the direct use of s_2 's in that machine can be replaced by copying instructions. (The idea is that in order to compare x

and β you copy them again on the tape and compare them by erasing 1's of both at the same rate and checking which one runs out first). Thus the problem of replacing $\Lambda = 3$ by $\Lambda = 2$ in Theorems 2-4 reduces to the problem of copying numerals without using s_2 's. What seems to the author to be the simplest way to achieve the aim of reducing the number of symbols to two is to use a Turing machine with two heads, one reading head and one printing head, with the two heads moving independently of each other. Each atomic act will depend on the state of the machine and on the symbol in the square scanned by the reading head. The values of the function M_0 will be quadruples, because the movement, if any, of each of the heads has to be indicated. Symbols are printed and erased only in the square scanned by the printing head. It can be easily seen that such a machine can do everything that a Turing machine of §3 can do and, in addition, it can copy numerals without effecting at all the copied numeral. Therefore if we would use this more general notion of a Turing machine we could replace $\Lambda = 3$ by $\Lambda = 2$ in Theorems 2-4.

5. THE CARDINALITY OF THE VALUES OF RECURSIVE FUNCTIONALS

In this paragraph a theorem of Takeuti [13] will be reformulated to apply to functionals rather than to functions only.

Let $\varphi_1, \dots, \varphi_r$ be r given partial functions and let M be a set of numbers closed under the partial functions $F(\varphi_1, \dots, \varphi_r, x_1, \dots, x_n)$ where $n < \omega$ and F is any f.r.f. Let A denote the set of functions $F(\varphi_1, \dots, \varphi_r, x, \beta_1, \dots, \beta_k)$ where $k < \omega$, F is any f.r.f. and $\beta_1, \dots, \beta_k \in M$. M is obviously closed under the partial functions in A . Let τ be the one-one order preserving map which maps M on an ordinal μ . Let $f \in A$, we shall say that a partial function g corresponds to f if for every $\beta \in M$ $\tau(f(\beta)) = g(\tau(\beta))$.⁵ By this equality we mean that if $f(\beta)$ is defined then $g(\tau(\beta))$ is defined and the equality holds, but $g(\tau(\beta))$ can be defined without $f(\beta)$ being defined.

LEMMA 5 (Takeuti). Let $\varphi_1, \dots, \varphi_r, M, A$ and τ be as above. Let F be a f.r.f., let $g_1, \dots, g_m$ correspond to $f_1, \dots, f_m$, respectively, and let $\alpha_1, \dots, \alpha_n \in M$ then

$$\tau(F(f_1, \dots, f_m, \alpha_1, \dots, \alpha_n)) = F(g_1, \dots, g_m, \tau(\alpha_1), \dots, \tau(\alpha_n)).$$

⁵Under very similar circumstances Takeuti [13] says that g is an $f^{\tau\alpha}$.

We shall not prove this Lemma since it is essentially Theorem of Takeuti [13] and has almost exactly the same proof. Following Theorem 3 of Takeuti [13] we get

THEOREM 6 (Takeuti). Let $\omega < \omega_f < \Omega$. Let $G(f_1, \dots, f_m, x_1, \dots, x_n)$ be a f.r.f., let $\varphi_1, \dots, \varphi_m$ be partial functions such that for some $\beta < \omega_f$ $\varphi_i(\delta) = 0$ for $\delta \geq \beta$ and if $\delta < \beta$ and $\varphi_i(\delta)$ is defined then $\varphi_i(\delta) < \beta$, $1 \leq i \leq m$, and let $\alpha_1, \dots, \alpha_n < \omega_f$ then $G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) < \omega_f$ if $G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is defined.

PROOF. Let $\beta' = \max(\beta, \alpha_1, \dots, \alpha_n) + 1$. Let

$M = \{F(\varphi_1, \dots, \varphi_m, \delta_1, \dots, \delta_k) \mid k < \omega, F \text{ is a f.r.f., } \delta_1, \dots, \delta_k < \beta'\}$

M is obviously closed under the partial functions $F(\varphi_1, \dots, \varphi_m, x_1, \dots, x_k)$

where $k < \omega$ and F is a f.r.f. $\beta' \in M$ since $p^{(1,1)}$ is a f.r.f.,

and therefore $\tau(\alpha) = \alpha$ for $\alpha < \beta'$. $\varphi_1, \dots, \varphi_m \in A$ by schema (5);

we have for $\alpha < \beta$ $\tau(\varphi_i(\alpha)) = \varphi_i(\alpha) = \varphi_i(\tau(\alpha))$ and for $\alpha \in M$

$\alpha \geq \beta$ we have $\tau(\alpha) \geq \beta$ and hence $\tau(\varphi_i(\alpha)) = \tau(0) = 0 = \varphi_i(\tau(\alpha))$.

Thus each φ_i corresponds to itself and since also $\tau(\alpha_i) = \alpha_i$

for $1 \leq i \leq n$, as $\alpha_i < \beta'$, we get by Lemma 5

$G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \tau(G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)) < \mu$. Since

there are only $\aleph_0$ f.r.f.'s we have $|\mu| = |M| \leq \max(\aleph_0, |\beta'|) < \omega_f$.

where $|x|$ denotes the cardinality of the set x .

THEOREM 7. Let ω_f be a regular ordinal $< \Omega$. Let $G(f_1, \dots, f_m, x_1, \dots, x_n)$ be a f.r.f., let $\varphi_1, \dots, \varphi_m$ be partial recursive functions such that $\varphi_i(\delta) = 0$ for $\delta \geq \omega_f$ and $\varphi_i(\delta) < \omega_f$ for $\delta < \omega_f$, $1 \leq i \leq m$, and let $\alpha_1, \dots, \alpha_n < \omega_f$ be such that $G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is defined then there are arbitrarily large $\beta < \omega_f$ such that $G(\varphi_1 \upharpoonright \beta, \dots, \varphi_m \upharpoonright \beta, \alpha_1, \dots, \alpha_n)$ is defined and is $< \omega_f$.

PROOF. The proof of this theorem is similar to the proof of Theorem 6, but uses an additional idea of Hajnal [2].

Let $[a]$ denote the closure of the set a of numbers with respect to the functions $F(\varphi_1, \dots, \varphi_m, x_1, \dots, x_k)$, where $k < \omega$ and F is a f.r.f. Since there are only $\aleph_0$ f.r.f.'s $|[a]| \leq \max(\aleph_0, |a|)$. Let $a_0 = \max(\alpha_1, \dots, \alpha_n) + 1 + \zeta$, where ζ is an arbitrary ordinal $< \omega_f$, let $a_{n+1} = [U(a_n \cap \omega_f) \cup a_n]$, where $U(x)$ is the union set of x ,

i.e., the set of all the members of the members of x , and let

$M = \bigcup_{n=0}^{\infty} a_n$. It follows by induction that $|a_n| \leq \max(\aleph_0, |a_0|)$ and

hence $|M| < \omega_f$. It is also easily seen that since ω_f is regular

$M \cap \omega_f = \beta$, where $\beta < \omega_f$.⁶ By $M \cap \omega_f = \beta$ we have

$\tau(\alpha) = \alpha$ for $\alpha \in M$, $\alpha < \omega_f$, $\tau(\alpha) \geq \beta$ for $\alpha \in M$, $\alpha \geq \omega_f$.

⁶For detailed proofs of these statements see Hajnal [2], 4.8.8.

Therefore for $\alpha < \omega_f$, $\alpha \in M$, we have $\alpha < \beta$ and also

$$\begin{aligned} \varphi_1(\alpha) < \omega_f, \quad \varphi_1(\alpha) \in M, \quad \text{hence } \tau(\varphi_1(\alpha)) &= \varphi_1(\alpha) = (\varphi_1 \upharpoonright \beta)(\alpha) = \\ &= (\varphi_1 \upharpoonright \beta)(\tau(\alpha)); \quad \text{for } \alpha \geq \omega_f, \quad \alpha \in M, \quad \tau(\varphi_1(\alpha)) = \tau(0) = 0 = \\ &= (\varphi_1 \upharpoonright \beta)(\tau(\alpha)). \quad \text{Thus } \varphi_1 \upharpoonright \beta \text{ corresponds to } \varphi_1 \text{ and since} \end{aligned}$$

$$\tau(\alpha_i) = \alpha_i, \quad 1 \leq i \leq n, \quad \text{by } \alpha_1, \dots, \alpha_n \in M \text{ and } \alpha_1, \dots, \alpha_n < \omega_f,$$

we have, by Lemma 5,

$$\tau(G(\varphi_1, \dots, \varphi_n, \alpha_1, \dots, \alpha_n)) = G(\varphi_1 \upharpoonright \beta, \dots, \varphi_n \upharpoonright \beta, \alpha_1, \dots, \alpha_n).$$

Since $\zeta < \omega_f$, $\zeta \in M$ we get $\zeta \leq \beta$ and β can be taken arbitrarily large $< \omega_f$.

6. ARITHMETIZATION OF THE NOTION OF RELATIVE CONSTRUCTIBILITY

Takeuti used in [13] an arithmetization of the notion of constructibility developed by him in [11] and [12]. The same arithmetization was carried out by Machover in [6] for the purpose of arithmetizing his formalism of recursive functions. Here we shall need an arithmetization of the notion of relative constructibility developed by Hajnal [2], Shoenfield [10] and the author [5]. We shall do it using the methods of Takeuti and Machover. This arithmetization will be done, following and Takeuti and Machover, within the realm of finitary recursive functionals. We shall use here freely the terminology and notations of [5].

Let $k \subseteq F_k^{\Omega}$. Let 1 be the cocharacteristic function of the set $\{0d_k(u) | u \in k\}$ (by the cocharacteristic function of a set a we mean the function f such that $f(x) = 1$ for $x \in a$ and $f(x) = 0$ for $x \notin a$). We shall give an arithmetical definition of the relations $F_k(\alpha) \in F_k(\beta)$ and $F_k(\alpha) = F_k(\beta)$, for $\alpha, \beta < \Omega$, and we shall see that these relations are finitarily recursive in $1, \alpha, \beta$. We shall identify every relation with its characteristic function (i.e., the function whose value is 0 if the relation holds and 1 otherwise), and thus a relation is said to be f.r. if its characteristic function is f.r. We shall now list various f.r. functions and relations. We shall use rule (6) freely in showing that they are f.r.

$$(9) \quad \alpha < \beta, \quad \alpha > \beta$$

$$\alpha < \beta = <(\alpha, \beta) \quad \alpha > \beta = \beta < \alpha$$

(10) $\sim Q$, where Q is a f.r. predicate

$$\sim Q = N(Q)$$

(11) $Q \wedge R$ where Q and R are f.r. predicates

$$Q \wedge R = \wedge(Q, R)$$

(12) $\mathcal{C}(Q_1, \dots, Q_k)$, where $\mathcal{C}$ is any sentential connective and

$Q_1, \dots, Q_k$ are f.r. predicates.

Every sentential connective is obtained by iteration of $\wedge$ and $\sim$.

(13) $\alpha \geq \beta$

$$\alpha \geq \beta = \sim \alpha < \beta$$

(14) $\alpha \text{ eq } \beta \equiv \alpha \text{ equals } \beta$

$$\alpha \text{ eq } \beta = \alpha \leq \beta \wedge \beta \leq \alpha$$

We shall also write $\alpha = \beta$ for $\alpha \text{ eq } \beta$, but only in cases

where no confusion may arise.

(15) $(\forall \beta < \alpha) Q$, $(\exists \beta < \alpha) Q$, where Q is a f.r. predicate

$$(\forall \beta < \alpha) Q = \alpha \text{ eq } (\mu \beta) (\beta \text{ eq } \alpha \vee \sim Q)$$

$$(\exists \beta < \alpha) Q = \sim (\forall \beta < \alpha) \sim Q$$

(16) $\alpha + \beta$

$\alpha + \beta$ is obtained from $(\mu \delta) (\delta \geq \alpha \wedge (\forall \gamma < \beta) (\delta > f(\gamma)))$ by

recursion on f and β .

(17) $s(\alpha)$ = the successor of α

$$s(\alpha) = (\mu \beta) (\beta > \alpha)$$

(18) $\text{suc}(\alpha)$ = α is a successor

$$\text{suc}(\alpha) = (\exists \beta < \alpha) (s(\beta) \text{ eq } \alpha)$$

(19) $p(\alpha)$ = the predecessor of α if there is such and α otherwise.

$$p(\alpha) = (\mu \beta) (s(\beta) \geq \alpha)$$

$$(20) \quad 0$$

$$0 = (\mu\alpha)(\alpha = \alpha)$$

$$(21) \quad \text{cond}(\beta, \alpha) = \begin{cases} 0 & \text{if } \beta = 0 \\ \alpha & \text{if } \beta > 0 \end{cases}$$

$$\text{cond}(\beta, \alpha) = (\mu\gamma)(\beta = 0 \wedge \gamma = 0 \vee \beta > 0 \wedge \gamma = \alpha)$$

$$(22) \quad \max(\alpha, \beta) = \text{maximum of } \alpha \text{ and } \beta$$

$$\max(\alpha_1, \dots, \alpha_n) = \text{maximum of } \alpha_1, \dots, \alpha_n$$

$$\max(\alpha, \beta) = \text{cond}(\alpha < \beta, \alpha) + \text{cond}(\alpha \geq \beta, \beta)$$

$$\max(\alpha_1, \dots, \alpha_n) = \max(\alpha_1, \max(\alpha_2, (\dots \max(\alpha_{n-1}, \alpha_n) \dots)))$$

$$(23) \quad F = \begin{cases} G_0 & \text{if } Q_0 \\ \vdots & \\ \vdots & \\ G_{k-1} & \text{if } Q_{k-1} \\ G_k & \text{otherwise} \end{cases} \quad \begin{array}{l} \text{where } G_0, \dots, G_k \text{ are f.r.} \\ \text{functionals, } Q_0, \dots, Q_{k-1} \\ \text{are f.r. predicates so that} \\ \text{no two of them hold for the} \\ \text{same arguments. Note that } F \\ \text{is defined only for the values} \end{array}$$

for which all the G 's (and the Q 's) are defined.

$$F = \max(\text{cond}(\sim Q_0, G_0), \dots, \text{cond}(\sim Q_{k-1}, G_{k-1}), \text{cond}(Q_0 \vee \dots \vee Q_{k-1}, G_k))$$

$$(24) \quad \sup_{\beta < \gamma} F, \text{ where } F \text{ is a f.r.f. (note that if } \hat{\beta} F(\beta) \uparrow \gamma \text{ is not}$$

bounded by a number ($< \Omega$), which can happen only if Ω is singular,

then $\sup_{\beta < \gamma} F$ is undefined). $\sup_{\beta < \gamma} F = (\mu\gamma)(\forall\beta < \alpha)(\gamma \geq F)$, where γ is

not a variable of F .

$$(25) \quad \sum_{\beta < \alpha} F, \text{ where } F \text{ is a f.r.f.}$$

$$\sum_{\beta < \alpha} F \text{ is obtained from } G(f, \alpha) = \begin{cases} f(p(\alpha)) + F(\alpha) & \text{if } \text{suc}(\alpha) \\ \sup_{\beta < \alpha} f(\beta) & \text{otherwise} \end{cases}$$

by recursion on f and α .

$$(26) \quad \alpha \cdot \beta = \sum_{\gamma < \beta} \alpha$$

$$(27) \quad 1, 2, 3, \dots, \underline{n}, \dots, \text{ where } \underline{n} \text{ is the constant function } n. \\ 1 = s(0), \quad 2 = s(1), \dots, \underline{n} = s(s(\dots s(0) \dots)), \dots$$

$$(28) \quad P(\alpha, \beta). \text{ This is the pairing function defined by Gödel [1], 7.9.}$$

It is not difficult to prove that for every ordinal γ P restricted to $\omega^{\omega^\gamma} \times \omega^{\omega^\gamma}$ is a one-one mapping of this set onto ω^{ω^γ} , and hence P restricted to $\Omega \times \Omega$ is a one-one mapping of $\Omega \times \Omega$ onto Ω . We note that $P(\alpha, \beta) \geq \alpha, \beta$ for all ordinals α, β .

$$P(\alpha, \beta) = \sum_{\gamma < \max(\alpha, \beta)} (\gamma \cdot 2 + 1) + \beta + \alpha \cdot (\alpha > \beta).^7$$

$$(29) \quad (\forall \beta \leq \alpha) R(\beta), (\exists \beta \leq \alpha) R(\beta), \text{ where } R \text{ is a f.r. predicate}$$

$$(\forall \beta \leq \alpha) R(\beta) = (\forall \beta < \alpha) R(\beta) \wedge R(\alpha)$$

$$(\exists \beta \leq \alpha) R(\beta) = (\exists \beta < \alpha) R(\beta) \vee R(\alpha)$$

$$(30) \quad L_1(\alpha), L_2(\alpha), \text{ where } L_1 \text{ and } L_2 \text{ are the inverse functions of}$$

⁷This way of defining $P(\alpha, \beta)$ as a f.r.f. is, essentially, due to Machover [6].

$P(\alpha, \beta)$.

$$L_1(\alpha) = (\mu\beta)(\exists\gamma \leq \alpha)(P(\beta, \gamma) = \alpha)$$

$$L_2(\alpha) = (\mu\beta)(P(L_1(\alpha), \beta) = \alpha)$$

(31) $J^*(\alpha, \beta, \gamma)$, where J^* is the function defined in [5], p.147.

$$J^*(\alpha, \beta, \gamma) = 10 \cdot P(\beta, \gamma) + \alpha$$

(32) $K_0^*(\alpha), K_1^*(\alpha), K_2^*(\alpha)$, where K_0^*, K_1^*, K_2^* are the inverse functions

of J^* , where α in J^* is supposed to vary over numbers < 10 only.

$$K_0^*(\alpha) = (\mu\beta)(\exists\gamma \leq \alpha)(10 \cdot \gamma + \beta = \alpha)$$

$$K_1^*(\alpha) = (\mu\beta)(\exists\gamma \leq \alpha)(J^*(K_0^*(\alpha), \beta, \gamma) = \alpha)$$

$$K_2^*(\alpha) = (\mu\beta)(J^*(K_0^*(\alpha), K_1^*(\alpha), \beta) = \alpha)$$

We shall now define a predicate $F_n(1, \alpha)$ such that $F_n(1, P(\beta, \gamma)) = 0$

if and only if $\beta > \gamma$ and $F_k(\gamma) \in F_k(\beta)$ or $\beta \leq \gamma$ and $F_k(\gamma) = F_k(\beta)$.

We shall now define the following functionals in which we shall substitute

$\hat{F}_n(1, x)$ for f .

$$(33) \quad <(1, \beta, \gamma) = \beta > \gamma \wedge f(P(\beta, \gamma)) \text{ eq } 0$$

$$= (1, \beta, \gamma) = \beta \leq \gamma \wedge f(P(\beta, \gamma)) \text{ eq } 0 \vee \gamma \leq \beta \wedge f(P(\gamma, \beta)) \text{ eq } 0$$

$$\leq(1, \beta, \gamma) = (\exists \delta < \beta)(= (1, \delta, \gamma) \wedge <(1, \beta, \delta))$$

$$= (1, \beta; \{\gamma, \delta\}) = (\forall \lambda < \beta)(\leq(1, \beta, \lambda) \leftrightarrow = (1, \gamma, \lambda) \vee = (1, \delta, \lambda))$$

$$\wedge (\exists \lambda < \beta) = (1, \gamma, \lambda) \wedge (\exists \lambda < \beta) = (1, \delta, \lambda)$$

$$\leq(1, \beta; \{\gamma, \delta\}) = (\exists \lambda < \beta)(\leq(1, \beta, \lambda) \wedge = (1, \lambda; \{\gamma, \delta\}))$$

$$= (1, \beta; <\gamma, \delta>) = (\exists \lambda, \sigma < \beta)(= (1, \beta; \{\lambda, \sigma\}) \wedge = (1, \lambda; \{\gamma, \gamma\}))$$

$$\wedge = (1, \lambda; \{\gamma, \delta\}))$$

$$\begin{aligned} \ll(r, \beta; \langle \gamma, \delta \rangle) &= (\exists \lambda < \beta) (\ll(r, \beta, \lambda) \wedge = (r, \lambda, \langle \gamma, \delta \rangle)) \\ &= (r, \beta; \langle \gamma, \delta, \delta \rangle) = (\exists \lambda < \beta) (\ll(r, \beta, \lambda) \wedge = (r, \lambda; \langle \delta, \delta \rangle)) \\ \ll(r, \beta; \langle \gamma, \delta, \delta \rangle) &= (\exists \lambda < \beta) (\ll(r, \beta, \lambda) \wedge = (r, \lambda; \langle \gamma, \delta, \delta \rangle)) \end{aligned}$$

(34) $H_1(r, \alpha)$, $1 \leq i \leq 8$, $i=10$, $H_9(1, r, \alpha)$. $H_1(r, \alpha)$, $1 \leq i \leq 8$ is

a functional such that for $\beta \in \mathcal{W}(J_1^*)$, $\beta > \gamma$, $H_1(Fn, P(\beta, \gamma)) = 0$,

where Fn is the predicate mentioned above, if and only if

$F_k(\gamma) \in F_k(\beta)$. $H_9(1, r, \alpha)$ is similar only that it depends also on 1

(or k). $H_{10}(r, \alpha)$ is such that for $\beta \leq \gamma$ $H_{10}(Fn, P(\beta, \gamma)) = 0$ if

and only if $F_k(\beta) = F_k(\gamma)$.

$$H_1(r, \alpha) = (r, L_2(\alpha), K_1^*(L_1(\alpha))) \vee = (r, L_2(\alpha), K_2^*(L_1(\alpha)))$$

$$\begin{aligned} H_2(r, \alpha) &= \ll(r, K_1^*(L_1(\alpha)), L_2(\alpha)) \wedge (\exists \beta, \gamma < L_2(\alpha)) (\ll(r, \gamma, \beta) \\ &\quad \wedge = (r, L_2(\alpha); \langle \beta, \gamma \rangle)) \end{aligned}$$

$$H_3(r, \alpha) = \ll(r, K_1^*(L_1(\alpha)), L_2(\alpha)) \wedge \sim \ll(r, K_2^*(L_1(\alpha)), L_2(\alpha))$$

$$\begin{aligned} H_4(r, \alpha) &= \ll(r, K_1^*(L_1(\alpha)), L_2(\alpha)) \wedge (\exists \beta, \gamma < L_2(\alpha)) (\ll(r, L_2(\alpha); \langle \beta, \gamma \rangle) \\ &\quad \wedge \ll(r, K_2^*(L_1(\alpha)), \gamma)) \end{aligned}$$

$$H_5(r, \alpha) = \ll(r, K_1^*(L_1(\alpha)), L_2(\alpha)) \wedge (\exists \lambda < K_2^*(L_1(\alpha))) (\ll(r, K_2^*(L_1(\alpha)); \langle \lambda, L_2(\alpha) \rangle))$$

$$H_6(r, \alpha) = \ll(r, K_1^*(L_1(\alpha)), L_2(\alpha)) \wedge (\exists \beta, \gamma < K_1^*(L_1(\alpha)))$$

$$(\ll(r, K_1^*(L_1(\alpha)); \langle \beta, \gamma \rangle) \wedge = (r, L_2(\alpha); \langle \gamma, \beta \rangle))$$

$$H_7(r, \alpha) = \ll(r, K_1^*(L_1(\alpha)), L_2(\alpha)) \wedge (\exists \beta, \gamma, \delta < K_1^*(L_1(\alpha)))$$

$$(\ll(r, K_1^*(L_1(\alpha)); \langle \beta, \gamma, \delta \rangle) \wedge = (r, L_2(\alpha); \langle \gamma, \delta, \beta \rangle))$$

$$H_8(r, \alpha) = \ll(r, K_1^*(L_1(\alpha)), L_2(\alpha)) \wedge (\exists \beta, \gamma, \delta < K_1^*(L_1(\alpha)))$$

$$(\ll(r, K_1^*(L_1(\alpha)); \langle \beta, \gamma, \delta \rangle) \wedge = (r, L_2(\alpha); \langle \gamma, \beta, \delta \rangle))$$

$$H_9(1, f, \alpha) = \langle (f, K_1^*(L_1(\alpha)), L_2(\alpha)) \wedge (\exists \beta \leq L_2(\alpha)) (= (f, L_2(\alpha), \beta) \wedge 1(\beta) = 1)$$

$$H_{10}(f, \alpha) = (\forall \beta < L_2(\alpha)) (\langle (f, L_1(\alpha), \beta) \leftrightarrow \langle (f, L_2(\alpha), \beta))$$

$$(35) \quad F_n(1, \alpha)$$

This functional is obtained from

$$G(1, f, \alpha) = \begin{cases} 0 & \text{if } L_1(\alpha) > L_2(\alpha) \wedge K_0^*(L_1(\alpha)) = 0 \\ H_1(f, \alpha) & \text{if } L_1(\alpha) > L_2(\alpha) \wedge K_0^*(L_1(\alpha)) = \underline{1}, \quad 1 \leq i \leq 8 \\ H_9(1, f, \alpha) & \text{if } L_1(\alpha) > L_2(\alpha) \wedge K_0^*(L_1(\alpha)) = 9 \\ H_{10}(f, \alpha) & \text{if } L_1(\alpha) \leq L_2(\alpha) \end{cases}$$

by recursion on f and α . One can now prove, by comparing this definition of F_n with the definition of F_k in [5] that F_n is indeed the required functional. To verify this note that in

$G(1, f, \alpha)$ f is applied only to ordinals $< \alpha$ since, as follows

easily from the definitions, if for some ordinal β $K_0^*(\beta) \neq 0$ then $K_1^*(\beta), K_2^*(\beta) < \beta$.

$$(36) \quad \alpha \approx_1 \beta \equiv F_k(\alpha) = F_k(\beta)$$

$$\alpha \approx_1 \beta = (\hat{x} F_n(1, x), \alpha, \beta)$$

$$(37) \quad \alpha \varepsilon_1 \beta \equiv F_k(\alpha) \in F_k(\beta)$$

$$\alpha \varepsilon_1 \beta = \langle (\hat{x} F_n(1, x), \alpha, \beta) \rangle$$

Having arithmetized the notion of relative constructibility we shall now apply Theorems 6 and 7 to get an alternative, essentially simpler, proof of the main technical lemmas of Hajnal [2] and Shoenfield [10]. If we

use Takeuti's arithmetization of the notion of constructibility we can get, exactly as in the proof of the following Theorem 8, a proof of the main technical lemma of Gödel [1], i.e., [1] 12.2.

THEOREM 8. Let k be k -constructible, let $\text{Od}_k(k) < \omega_\gamma$, let $\lambda < \omega_\gamma$ and let u be a k -constructible subset of $F_k(\lambda)$; then $\text{Od}_k(u) < \omega_\gamma$.

PROOF. Without loss of generality we can assume that $\lambda \in \mathcal{U}(J_0^*)$

since otherwise for some $1 \leq i \leq 9$ $\lambda + i \in \mathcal{U}(J_0^*)$, $u \subseteq F_k(\lambda) \subseteq F_k^{\#} \lambda$

$\subseteq F_k^{\#}(\lambda + i) = F_k(\lambda + i)$, $\lambda + i < \omega_\gamma$. Let $\mathfrak{J}$ be any cardinal $> \omega_\gamma$.

Let l be the cocharacteristic function of $\{\text{Od}_k(x) \mid x \in k\}$, then in

analogy to [1] 9.52, $l(\delta) = 0$ for $\delta \geq \text{Od}_k(k)$. Let f be the

cocharacteristic function of $\{\sigma \mid \sigma < \lambda \wedge F_k(\sigma) \in u\}$, then $f(\delta) = 0$ for

$\delta \geq \lambda$. Since u is k -constructible $(\mu\alpha)(\forall\sigma < \lambda)(\sigma \varepsilon_1 \alpha \leftrightarrow f(\sigma) = 1)$

is defined, by the definition of f . $(\mu\alpha)(\forall\sigma < \lambda)(\sigma \varepsilon_1 \alpha \leftrightarrow f(\sigma) = 1)$

is a f.r.f. in l, f and λ and hence, by Theorem 6,

$(\mu\alpha)(\forall\sigma < \lambda)(\sigma \varepsilon_1 \alpha \leftrightarrow f(\sigma) = 1) < \omega_\gamma$. Therefore there is an ordinal

$\alpha < \omega_\gamma$ such that $F_k(\alpha) \cap F_k^{\#} \lambda = u$. Since $\lambda \in \mathcal{U}(J_0^*)$ $F_k^{\#} \lambda = F_k(\lambda)$,

and hence $u = F_k(\alpha) \cap F_k(\lambda)$, and therefore $u = F_k(J^*(3, \alpha, J^*(3, \alpha, \lambda)))$,

i.e., $\text{Od}_k(u) < J^*(3, \alpha, J^*(3, \alpha, \beta)) < \omega_\gamma$ (The last inequality follows

from the observation that $J^*(i, \alpha, \beta) < \omega^{\omega^\gamma}$ for every γ and

$\alpha, \beta < \omega^{\omega^\gamma}$).

THEOREM 9 (Hajnal). Let ω_f be a regular ordinal, let $k \subseteq F_k'' \omega_f$, let $\lambda < \omega_f$ and let u be a k -constructible subset of $F_k(\lambda)$; then for some $\beta < \omega_f$ $\text{Oa}_{k \cap F_k'' \beta}(u) < \omega_f$.

PROOF. Let l, f and λ be as in the proof of Theorem 8, then

$l(\delta) = 0$ for $\delta \geq \omega_f$, $f(\delta) = 0$ for $\delta \geq \lambda$. Since u is k -constructible $(\mu \alpha)(\forall \sigma < \lambda)(\sigma \varepsilon_1 \alpha \leftrightarrow f(\sigma) = 1)$ is defined. $(\mu \alpha)(\forall \sigma < \lambda)(\sigma \varepsilon_1 \alpha \leftrightarrow f(\sigma) = 1)$

is a f.r.f. in l, f and λ and hence, by Theorem 7, there exists a β ,

$\omega_f > \beta > \lambda$, such that $(\mu \alpha)(\forall \sigma < \lambda)(\sigma \varepsilon_{1\beta} \alpha \leftrightarrow (f \upharpoonright \beta)(\sigma) = 1)$ is

defined and is $< \omega_f$. Since $\beta > \lambda$, $f \upharpoonright \beta = f$ and hence

$(\mu \alpha)(\forall \sigma < \lambda)(\sigma \varepsilon_{1\beta} \alpha \leftrightarrow f(\sigma) = 1) < \omega_f$. It is easily seen, by induction

on τ , that $F_{k \cap F_k'' \beta}(\tau) = F_k(\tau)$ for $\tau < \beta$. Therefore $1 \upharpoonright \beta$

is the cocharacteristic function of $\{\text{Oa}_{k \cap F_k'' \beta}(x) \mid x \in k \cap F_k'' \beta\}$ and

f is the cocharacteristic function of $\{\sigma \mid \sigma < \lambda \wedge F_{k \cap F_k'' \beta}(\sigma) \in u\}$.

Since $(\mu \alpha)(\forall \sigma < \lambda)(\sigma \varepsilon_{1\beta} \alpha \leftrightarrow f(\sigma) = 1) < \omega_f$ there exists an ordinal

$\alpha < \omega_f$ such that $F_{k \cap F_k'' \beta}(\alpha) \cap F_{k \cap F_k'' \beta}'' \lambda = u$. Since $\lambda \in \mathcal{W}(J_0^*)$

$u = F_{k \cap F_k'' \beta}(\alpha) \cap F_{k \cap F_k'' \beta}(\lambda) = F_{k \cap F_k'' \beta}(J^*(3, \alpha, J^*(3, \alpha, \beta)))$ and

$J^*(3, \alpha, J^*(3, \alpha, \beta)) < \omega_f$.

LEMMA 10. Every ordinal $< \Omega$ has k -order $< \Omega$.

PROOF. Let φ be a formula of set theory containing no class variables. We shall denote with $\varphi^{(T)}$ the formula obtained from φ by relativizing all its quantifiers to the class T , i.e., by replacing each quantifier of the type $(\forall x)$ or $(\exists x)$ by $(\forall x)(x \in T \rightarrow \dots)$ or $(\exists x)(x \in T \wedge \dots)$, respectively. In the following we shall consider formulas which contain V, L_k, Od , etc., as formulas which contain no class variables, since those class constants can be eliminated without introducing class variables.

By [5] 2.9 $V = L_k \rightarrow Od_k(\beta) < \omega \vee |Od_k(\beta)| \leq |\beta|$. Hence

$$(V = L_k \cap L_k)^{(L_k)} \rightarrow (Od_{k \cap L_k}(\beta) < \omega)^{(L_k)} \vee (|Od_{k \cap L_k}(\beta)| \leq |\beta|)^{(L_k)},$$

since L_k is a complete model of set theory, in the sense of Sheperdson [8].

As shown in [5] p. 148 $(V = L_k \cap L_k)^{(L_k)}$ is provable, hence we get

$(Od_{k \cap L_k}(\beta) < \omega)^{(L_k)} \vee (|Od_{k \cap L_k}(\beta)| \leq |\beta|)^{(L_k)}$. In analogy to [1] 11.6

we obtain that F_k is absolute and hence Od_k is absolute. Therefore, since $k \cap L_k$ is a k -constructible set by [5] 1.5 and since $\leq$ and ω are absolute by [8] 2.318 and 2.322 (L_k obviously satisfies assumption 2.312 (b) of [8] since we can take for v the member of U of least k -order) we get $|Od_{k \cap L_k}(\beta)|_{(L_k)} < \omega \vee |Od_{k \cap L_k}(\beta)|_{(L_k)} \leq |\beta|_{(L_k)}$, where $|\gamma|_{(L_k)}$ means the cardinality of γ in L_k , i.e., the least ordinal on which γ can be mapped by a one-one mapping which is a member

of L_k . Obviously, for each γ $|\gamma|_{(L_k)}| = |\gamma|$ therefore, by taking cardinalities in $|\text{Od}_k \cap L_k(\beta)|_{(L_k)} < \omega \vee |\text{Od}_k \cap L_k(\beta)|_{(L_k)} \leq |\beta|_{(L_k)}$, we get $\text{Od}_k \cap L_k(\beta) < \omega \vee |\text{Od}_k \cap L_k(\beta)| \leq |\beta|$. Finally we note that one can easily prove by induction on γ that $F_k(\gamma) = F_{k \cap L_k}(\gamma)$ and hence $\text{Od}_k \cap L_k(\beta) = \text{Od}_k(\beta)$. Thus we get $\text{Od}_k(\beta) < \omega \vee |\text{Od}_k(\beta)| \leq |\beta|$ from which we get $\beta < \Omega \rightarrow \text{Od}_k(\beta) < \Omega$.

LEMMA 11. Let $k \in F_k''\Omega$, let g be a k -constructible bounded subset of $\Omega \times \Omega$, i.e., for some $\delta < \Omega$ $g \subseteq \delta \times \delta$ (which holds, in particular if Ω is regular and $|g| < \Omega$), then $\text{Od}_k(g) < \Omega$.

PROOF. Let $g \subseteq \delta \times \delta$, $\delta < \Omega$. By Lemma 10 $\text{Od}_k(\delta) = \wp < \Omega$. For $\langle \beta, \gamma \rangle \in \delta \times \delta$, $\beta, \gamma < \delta$ and hence, in analogy to [1] 9.52, $\text{Od}_k(\beta)$, $\text{Od}_k(\gamma) < \wp$ and since J^* is monotonic in all variables $\text{Od}_k(\langle \beta, \gamma \rangle) \leq J^*(1, J^*(1, \text{Od}_k(\beta), \text{Od}_k(\beta)), J^*(1, \text{Od}_k(\beta), \text{Od}_k(\gamma))) < J^*(1, J^*(1, \wp, \wp), J^*(1, \wp, \wp)) < \Omega$. Let λ be an ordinal, $\lambda \in \mathcal{U}(J_0^*)$, $J^*(1, J^*(1, \wp, \wp), J^*(1, \wp, \wp)) \leq \lambda < \Omega$. $g \subseteq \delta \times \delta \subseteq F_k''\lambda = F_k(\lambda)$. By Theorem 8 we get $\text{Od}_k(g) < \Omega$.

$$(38) \quad \text{or}_1(\alpha) = \text{Od}_k(\alpha)$$

$\text{or}_1(\alpha)$ is obtained from the functional

$$(\mu\beta) [(\forall \gamma < \alpha)(\exists \delta \varepsilon_1 \beta) \wedge (\forall \delta < \beta)(\delta \varepsilon_1 \beta \rightarrow (\exists \gamma < \alpha)(\exists \delta = \gamma))]$$

by recursion on f and α . $or_1(\alpha)$ is defined (for 1 as above) for every $\alpha < \Omega$ by Lemma 10.

(39) $ord_1(\alpha) \equiv F_k(\alpha)$ is an ordinal

$ord_1(\alpha) = (\exists \beta \leq \alpha)(or_1(\beta) \approx \alpha)$ (note that $Od_k(\alpha) \geq \alpha$ since $Od_k(\alpha)$ is a monotonic function of α).

(40) $val_1(\alpha) = F_k(\alpha)$ if $F_k(\alpha)$ is an ordinal (and 0 otherwise).

$val_1(\alpha) = (\mu \beta)(or_1(\beta) \approx_1 \alpha \vee \sim ord_1(\alpha))$

(41) $od_1(\alpha) = Od_k(F_k(\alpha))$

$od_1(\alpha) = (\mu \beta)(\beta \approx_1(\alpha))$

(42) $[\alpha, \beta] =$ an ordinal γ such that $F_k(\gamma) = \langle F_k(\alpha), F_k(\beta) \rangle$

$[\alpha, \beta] = J^*(1, J^*(1, \alpha, \alpha), J^*(1, \alpha, \beta))$.

7. RECURSIVE FUNCTIONALS IN COMPLETE MODELS OF SET THEORY

We can develop a system of notation for the f.r.f.'s. We actually gave names to the functionals given by schemata (1) - (5). We wrote also the functionals given by schemata (6) and (8) explicitly in terms of the given functionals. In the case of schema (7), where F is defined by

$$F(f_1, \dots, f_m, x_1, \dots, x_n) = G(\hat{y}F(f_1, \dots, f_m, y, x_2, \dots, x_n) \uparrow x_1, f_1, \dots, f_m, x_1, \dots, x_n)$$

let us denote

$$F(f_1, \dots, f_m, x_1, \dots, x_n) = \text{Rec}_{f_0, y}(G(f_0, f_1, \dots, f_m, y, x_2, \dots, x_n), x_1).$$

We can Gödelize this system of notation and therefore we can assume that the names of the functionals are natural numbers, and the simple operations on names we use, like substitution, are simple arithmetical operations. What is the connection between the notations and the functions? It is given actually in Definition 1. It is a theorem of set theory that to each such notation corresponds a function as laid down in Definition 1.

THEOREM 12. Assume we are given a complete model of set theory in the sense of Shepherdson [8] whose universal class is M ($V_M = M$ in the notation of [8]) and $\mathcal{Q} \in M$. If $\mathcal{W}$ is a notation for a f.r.f., $F_{\mathcal{W}}$ the corresponding functional and $F_{\mathcal{W}}^{(M)}$ the corresponding functional in the model then

$$F_{\mathcal{W}}^{(M)}(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = F_{\mathcal{W}}(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) \text{ for } \varphi_1, \dots, \varphi_m \in M.$$

PROOF. By [8] 2.322 the notion of a natural number is absolute with respect to M . One can easily see that the (finite) arithmetical operations are absolute with respect to M and hence the operations on the notations for functionals which we use are absolute. We shall prove the theorem by induction on the length of the notation.

(1) $\mathcal{W}$ is $P^{(n,i)}(x_1, \dots, x_n)$, then $F_{\mathcal{W}}$ is given by

$F_{\mathcal{W}}(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \alpha_i$. The relativization of this statement

to the model M is $F_{\mathcal{W}}^{(M)}(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \alpha_i$. Hence

$F_{\mathcal{W}}^{(M)}(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = F_{\mathcal{W}}(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$, and both

sides are defined.

(2) - (5). $\mathcal{W}$ is $N(x)$, $\wedge(x, y)$, $<(x, y)$ or $f_i(x)$. The proof is easy.

(6) $\mathcal{W}$ is $G(f_1, \dots, f_m, H_1(f_1, \dots, f_m, x_1, \dots, x_n), \dots, H_l(f_1, \dots, f_m, x_1, \dots, x_n))$

where $G(f_1, \dots, x_1)$, $H_1(f_1, \dots, x_n), \dots, H_l(f_1, \dots, x_n)$ are shorter notations.

What we have to prove follows easily from the induction hypothesis concerning these shorter notations.

(7) $\mathcal{W}$ is $\text{Rec}_{f_0, y}^{(G(f_0, f_1, \dots, f_m, y, x_2, \dots, x_n), x_1)}$. We shall write $\wp$

for $G(f_0, f_1, \dots, f_m, y, x_2, \dots, x_n)$. We shall prove the theorem for this case

by induction on α_1 . Our induction hypothesis is, therefore,

$F_{\mathcal{W}}^{(M)}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) = F_{\mathcal{W}}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n)$, for all

$\beta < \alpha$, and both sides are defined or undefined alike.

$$F_w^{(M)}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) = F_p^{(M)}(\hat{\beta}_w^{(M)}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) \upharpoonright \alpha_1, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n),$$

where $\hat{\beta}_w^{(M)}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) \upharpoonright \alpha_1$ is a restriction in the sense of the model to α_1 of a partial function in M . But the operation of restriction of a partial function is easily seen to be absolute, hence

$\hat{\beta}_w^{(M)}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) \upharpoonright \alpha_1 \in M$ and the restriction is the ordinary restriction. By the induction hypothesis concerning $\wp$ we get

$$\begin{aligned} & F_p^{(M)}(\hat{\beta}_w^{(M)}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) \upharpoonright \alpha_1, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) \\ &= F_p(\hat{\beta}_w^{(M)}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) \upharpoonright \alpha_1, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) \\ & \text{which, by induction hypothesis on } \alpha_1, \\ &= F_p(\hat{\beta}_w(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) \upharpoonright \alpha_1, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) \\ &= F_w(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n). \end{aligned}$$

Since all these equalities have also the meaning that if one side is defined so is the other, this case of the theorem is proved.

(8) w is $(\mu^{x_{n+1}}) G(f_1, \dots, f_m, x_1, \dots, x_n, x_{n+1})$. The theorem follows easily from the induction hypothesis concerning $G(f_1, \dots, x_{n+1})$.

8. THE RECURSIVITY OF THE COMPUTABLE FUNCTIONALS FOR A REGULAR Ω .

Throughout this paragraph we shall assume that Ω is regular, but we shall always note when the assumption of regularity will be used.

We shall now arithmetize the action of a Turing machine. In the following we shall write $\langle \alpha, \beta \rangle^*$ for $P(\alpha, \beta)$ and $\langle \alpha, \beta, \gamma \rangle^*$ for $P(\alpha, P(\beta, \gamma))$. Assume we are given a particular ordered Turing machine and particular functions $\varphi_1, \dots, \varphi_m$. We define

$$k_0 = \{ \langle 0, \langle \gamma, \delta, \lambda \rangle^*, \langle \alpha, \beta \rangle^* \rangle \mid M_0(q_\alpha, s_\beta) = \langle q_\gamma, s_\delta, \lambda \rangle \vee (\alpha = 0 \vee \alpha \in \bigcup_{i=1}^m P_i) \wedge \beta < \Lambda \wedge \gamma = \delta = \lambda = 0 \}$$

$$k_1 = \{ \langle i, \alpha \rangle^* \mid q_\alpha \in P_i \}, \quad 1 \leq i \leq m$$

$$k_{m+1} = \{ \langle m+1, \beta, \alpha \rangle^* \mid M_1(\alpha) = \beta \vee \alpha < \Gamma \wedge \alpha \notin \bigcup_{i=1}^m P_i \wedge \beta = 0 \}$$

$$k_{m+1+i} = \{ \langle m+1+i, \beta, \alpha \rangle^* \mid \beta = \varphi_i(\alpha) \}, \quad 1 \leq i \leq m$$

$$k = \bigcup_{i=0}^{2m+1} k_i$$

k is a set of ordinal numbers. Let 1 be the cocharacteristic function of $\{O\alpha_k(u) \mid u \in k\}$. By Lemma 10 $k \leq F_k''\Omega$, hence 1 is a function on Ω into $\{0, 1\}$.

LEMMA 13. (a) $F_{k \cap F_k''\beta}(\alpha) = F_k(\alpha)$ for $\alpha < \beta$. (b) If 1 is as above then $1 \upharpoonright \beta$ is the cocharacteristic function of

$$\{O\alpha_{k \cap F_k''\beta}(u) \mid u \in k \cap F_k''\beta\}.$$

PROOF. (a) is proved by induction on α . (b) follows immediately from (a).

We shall now list further f.r.f.'s. We shall note what their values are only for some of the values of the arguments. We shall be interested only in the case where r is the cocharacteristic function of a set

$\{Od_s(u) \mid u \in s\}$ for an s such that $s \in F_s^{\Omega}$.

(43) $sqo_r(\alpha) \equiv F_s(\alpha)$ is a bounded (by an ordinal $< \Omega$) sequence of ordinals $< \Omega$ of length $< \Omega$.

$$\begin{aligned} sqo_r(\alpha) = & (\forall \beta < \alpha) (\exists \gamma, \delta < \beta) (\beta \varepsilon_r \alpha \rightarrow \beta \approx_r [\gamma, \delta]) \\ & \wedge (\forall \beta, \gamma, \delta < \alpha) ([\gamma, \beta] \varepsilon_r \alpha \wedge [\delta, \beta] \varepsilon_r \alpha \rightarrow \gamma \approx_r \delta) \\ & \wedge ord_r(\beta) \wedge ord_r(\gamma) \wedge ([\gamma, \beta] \varepsilon_r \alpha \wedge \delta \varepsilon_r \beta \rightarrow \\ & (\exists \zeta < \alpha) ([\zeta, \delta] \varepsilon_r \alpha)) \end{aligned}$$

(44) $Lh_r(\alpha) =$ the length of the sequence $F_s(\alpha)$ if $sqo_r(\alpha)$.

$$Lh_r(\alpha) = \sum_{\beta < \alpha} \sim [od_r(\beta) = \beta \wedge (\exists \gamma < \alpha) ([\gamma, \beta] \varepsilon_r \alpha)]$$

(45) $\alpha_{r,\beta} = [F_s(\alpha)](\beta)$ if $sqo_r(\alpha)$

$$\alpha_{r,\beta} = val_r((\mu \gamma) ([or_r(\gamma), or_r(\beta)] \varepsilon_r \alpha \vee \beta \geq Lh_r(\gamma)))$$

(46) $\beta_r(\gamma) =$ the s -order of the sequence obtained from $F_s(\beta)$ by replacing its γ -th member by ρ , if $sqo_r(\beta) \wedge \gamma < Lh_r(\beta)$.

$$\beta_r(\gamma) = (\mu \alpha) [(\forall \delta < Lh_r(\beta)) (\delta \neq \gamma \rightarrow \alpha_{r,\delta} = \beta_{r,\delta}) \wedge \alpha_{r,\gamma} = \rho]$$

If $sqo_r(\beta)$ then $\beta_r(\gamma)$ is defined since the sequence

$$F_s(\beta) = \{ \langle \gamma, [F_s(\beta)](\gamma) \rangle \} \cup \{ \langle \gamma, \rho \rangle \} \quad (\text{or } F_s(\beta) \cup \{ \langle \gamma, \rho \rangle \}, \text{ if } \gamma > Lh_r(\beta))$$

is obviously s -constructible and by Lemma 11 its s -order is $< \Omega$.

(47) β_r^+ = the s -order of the sequence obtained from $F_s(\beta)$ by adding a 0 at the end, if $\text{sqo}_r(\beta)$.

$$\beta_r^+ = (\mu \alpha)(\text{Lh}_r(\alpha) = \text{Lh}_r(\beta) + 1 \wedge (\forall \gamma < \text{Lh}_r(\alpha))(\alpha_{r,\beta} = \beta_{r,\gamma})).$$

When we shall now refer to the Turing machine we shall assume that the symbols are the ordinals $< \Lambda < \Omega$, the states are the ordinals $< \Gamma < \Omega$ and the states are ordered by the relation $<$. Every tape will be denoted by a sequence g of length $< \Omega$ (which is not unique) such that if s_τ is printed in p_γ and $\tau \neq 0$ then $g(\gamma) = \tau$ and if s_0 is printed in p_γ then $g(\gamma) = 0$ or $\gamma \geq$ the length of g . Without loss of generality we can assume that the initial state is 1, since if the initial state is $\sigma \neq 1$ we can replace each state λ by $1 + \lambda$ in the functions M_0, M_1 and in the sets $P_1, \dots, P_m$, and add 1 as a new initial state with $M_0(1, \tau) = \langle 1 + \sigma, \tau, 1 \rangle$. It is immediately seen that the first act of the machine is to pass from the state 1 to the state $1 + \sigma$ without any additional changes and that the machine never returns to the state 1 during the computation.

(48) $L_r(\beta, \gamma)$ = the index of the square on the tape $F_s(\beta)$ to which the machine will move from the square p_γ when the machine carries out the instruction $\underline{L}$

$$L_r(\beta, \gamma) = \begin{cases} p(\gamma) & \text{if } \text{suc}(\gamma) \\ (\mu \xi)(\forall \eta < \gamma)(\eta > \xi \rightarrow \beta_{r,\eta} = \beta_{r,\xi}) & \text{if } \sim \text{suc}(\gamma) \wedge (\exists \xi < \gamma) \\ \gamma & \text{otherwise} \end{cases}$$

(49) $W_r(\alpha, \beta, \gamma) =$ the least δ such that δ is the s -order of a tape obtained from $F_s(\alpha)$ by writing γ to the right of the square $p\beta$.

The tape obtained from $F_s(\alpha)$ by writing γ to the right of the square $p\beta$ can be shown to be s -constructible (by applying the absoluteness results in [8] 2.2 and similar methods). Hence, by Lemma 11 there is always such a δ .

$$W_r(\alpha, \beta, \gamma) = (\mu \delta) [\text{sqo}_r(\delta) \wedge \text{Lh}_r(\delta) = \max(\text{Lh}_r(\alpha), \beta+1+\gamma) \\ \wedge (\forall \lambda \leq \text{Lh}_r(\delta)) ((\beta < \lambda < \beta+1+\gamma \rightarrow \delta_{r,\lambda} = 1) \\ \wedge (\lambda \leq \beta \vee \lambda \geq \beta+1+\gamma \rightarrow \delta_{r,\lambda} = \alpha_{r,\lambda}))]$$

(50) $\varphi_1(\alpha)_1 = \varphi_1(\alpha)$, where φ_1 is as above

$$\varphi_1(\alpha)_1 = (\mu \beta) (1(\text{or}_1(\langle \underline{m+1+i}, \beta, \alpha \rangle^*)) = 1)$$

(51) $(\alpha \in P_1)_1 \equiv \alpha \in P_1$

$$(\alpha \in P_1)_1 = 1(\text{or}_1(\langle \underline{1}, \alpha \rangle^*)) \text{eq } 1$$

(52) $M_1(\alpha)_1 = M_1(\alpha)$ if $\alpha \in \bigcup_{i=1}^m P_i$, and 0 if $\alpha \in \bigcap_{i=1}^m P_i$.

$$M_1(\alpha)_1 = (\mu \beta) (1(\text{or}_1(\langle \underline{m+1}, \beta, \alpha \rangle^*)) = 1)$$

(53) $\text{Rd}_r(\gamma, \delta) =$ the α for which the tape $F_s(\gamma)$ is $\dots s_0 \underline{\alpha} \dots$, where the last square of $\underline{\alpha}$ is p_δ , and 0 otherwise.

$$\text{Rd}_r(\gamma, \delta) = (\mu \alpha) | (\exists \lambda < \delta) (\delta = \lambda + 2 + \alpha \wedge \gamma_{r,\lambda+1} = 0 \\ \wedge (\forall \tau < \delta) (\tau > \lambda + 1 \rightarrow \delta_{r,\tau} = 1) \wedge \gamma_{r,\lambda+1} = 0)$$

$$\begin{aligned} V \sim (\exists \alpha \leq \delta)(\exists \lambda < \delta)(\delta = \lambda + 2 + \alpha \wedge \gamma_{r, \lambda+1} = 0 \\ \wedge (\forall \tau < \delta)(\tau > \lambda + 1 \rightarrow \gamma_{r, \tau} = 1) \wedge \gamma_{r, \lambda+1} = 0) \end{aligned}$$

(54) $M_{oi,1,r}(\beta, \gamma, \delta)$ = the i -th component of the value of M_0 for the state β and the symbol $[F_s(\gamma)](\delta)$, $i = 1, 2, 3$, if M_0 is defined for this state, and 0 if M_0 is not defined for the state β .

$$\begin{aligned} M_{0,1,r}(\beta, \gamma, \delta) &= (\mu \zeta)(1(\text{or}_1(\langle 0, \zeta, \langle \beta, \gamma_{r, \delta} \rangle^* \rangle^*) = 1)) \\ M_{01,1,r}(\beta, \gamma, \delta) &= (\mu \eta)(\exists \sigma, \tau \leq M_{0,1,r}(\beta, \gamma, \delta))(M_{0,1,r}(\beta, \gamma, \delta) = \langle \gamma, \sigma, \tau \rangle^*) \\ M_{02,1,r}(\beta, \gamma, \delta) &= (\mu \eta)(\exists \sigma, \tau \leq M_{0,1,r}(\beta, \gamma, \delta))(M_{0,1,r}(\beta, \gamma, \delta) = \langle \sigma, \eta, \tau \rangle^*) \\ M_{03,1,r}(\beta, \gamma, \delta) &= (\mu \eta)(\exists \sigma, \tau \leq M_{0,1,r}(\beta, \gamma, \delta))(M_{0,1,r}(\beta, \gamma, \delta) = \langle \sigma, \tau, \eta \rangle^*) \end{aligned}$$

The configuration where the state is ξ , the tape is $F_s(\gamma)$ and the square scanned is p_ξ will be denoted with $\langle \xi, \gamma, \zeta \rangle_r^*$.

(55) $T_{1,r}(\beta, \gamma, \delta)$ = the configuration which follows the configuration $\langle \beta, \gamma, \delta \rangle_r^*$, where an inactive configuration is supposed to be followed by itself.

Note that each of the five functionals used in the definition of $T_{1,r}(\beta, \gamma, \delta)$, which follows, is defined (for l and r as above) for every $\beta < \Gamma, \gamma, \delta$ if $\text{sqo}_r(\gamma)$.

$$\begin{aligned}
 T_{1,r}(\beta, \gamma, \delta) = & \left\{ \begin{aligned}
 & \langle M_{01,1,r}(\beta, \gamma, \delta), \gamma_r^{+}(M_{02,1,r}(\beta, \gamma, \delta)), \delta+1 \rangle^* \\
 & \text{if } \beta \neq 0 \wedge \bigwedge_{i=1}^m \sim(\beta \in P_i)_1 \wedge M_{03,1,r}(\beta, \gamma, \delta) = 0 \\
 & \langle M_{01,1,r}(\beta, \gamma, \delta), \gamma_r(M_{02,1,r}(\beta, \gamma, \delta)), \delta \rangle^* \\
 & \text{if } \beta \neq 0 \wedge \bigwedge_{i=1}^m \sim(\beta \in P_i)_1 \wedge M_{03,1,r}(\beta, \gamma, \delta) = 1 \\
 & \langle M_{01,1,r}(\beta, \gamma, \delta), \gamma_r(M_{02,1,r}(\beta, \gamma, \delta)), L_r(\gamma, \delta) \rangle^* \\
 & \text{if } \beta \neq 0 \wedge \bigwedge_{i=1}^m \sim(\beta \in P_i)_1 \wedge M_{03,1,r}(\beta, \gamma, \delta) = 2 \\
 & \langle M_1(\beta)_1, W_r(\gamma, \delta, \varphi_i(Rd_r(\gamma, \delta))_r), \delta \rangle^* \\
 & \text{if } (\beta \in P_i)_1 \wedge (\exists \alpha \leq \delta)(\exists \lambda < \delta)(\delta = \lambda + 2 + \alpha \wedge \gamma_{r,\lambda+1} = 0 \\
 & \wedge (\forall \tau < \delta)(\tau > \lambda + 1 \rightarrow \gamma_{r,\tau} = 1) \wedge \gamma_{r,\delta} = 0) \\
 & i = 1, \dots, m \\
 & \langle \beta, \gamma, \delta \rangle^* \qquad \text{otherwise}
 \end{aligned} \right.
 \end{aligned}$$

In passing from the configuration $\langle \beta, \gamma, \delta \rangle_r^*$ to the configuration $T_{1,r}(\beta, \gamma, \delta)_r$ we use some information stored in l . The information used for each given $\langle \beta, \gamma, \delta \rangle_r^*$ is contained already in some restriction of l . The functions are predicates $M_{0i,1,r}(\beta, \gamma, \delta)$, $i = 1, 2, 3$, $(\beta \in P_i)_1$, $1 \leq i \leq m$,

$\mathbb{M}_1(\beta)_1$ and $\varphi_1(\text{Rd}_r(\gamma, \delta))_r$ use only information contained in the restriction of 1 to

$$\text{or}_1(\langle 0, (\mu \zeta)(1(\text{or}_1(\langle 0, \zeta, \langle \beta, \gamma_r, \delta \rangle^*)^*) = 1), \langle \beta, \gamma_r, \delta \rangle^*)^* \rangle + 1,$$

$$\text{or}_1(\langle \underline{1}, \beta \rangle^*) + 1, \text{or}_1(\langle \underline{m+1}, (\mu \lambda)(1(\text{or}_1(\langle \underline{m+1}, \lambda, \beta \rangle^*)) = 1), \beta \rangle^*) + 1,$$

$$\text{or}_1(\langle \underline{m+1+i}, (\mu \lambda)(1(\text{or}_1(\langle \underline{m+1+i}, \lambda, \text{Rd}_r(\gamma, \delta) \rangle^*)) = 1), \text{Rd}_r(\gamma, \delta) \rangle^*) + 1,$$

respectively. To verify this statement note that the function $P(\sigma, \tau)$ is monotonous in both arguments and that, as follows from Lemma 13, the value of $\text{or}_1(\sigma)$ depends only on the restriction of 1 to $\text{or}_1(\sigma)$. Thus the value of the functional $\text{In}_{1,r}(\beta, \gamma, \delta)$ which we shall now define is such that the restriction of 1 to $\text{In}_{1,r}(\beta, \gamma, \delta)$ contains already all the information used in the passage from $\langle \beta, \gamma, \delta \rangle_r^*$ to $\tau_{1,r}(\beta, \gamma, \delta)_r$.

$$(56) \quad \text{In}_{1,r}(\beta, \gamma, \delta) = \max(\text{or}_1(\langle 0, (\mu \zeta)(1(\text{or}_1(\langle 0, \zeta, \langle \beta, \gamma_r, \delta \rangle^*)^*) = 1), \langle \beta, \gamma_r, \delta \rangle^*)^*), \text{or}_1(\langle 1, \beta \rangle^*), \dots, \text{or}_1(\langle \underline{m}, \beta \rangle^*), \\ \text{or}_1(\langle \underline{m+1}, (\mu \lambda)(1(\text{or}_1(\langle \underline{m+1}, \lambda, \beta \rangle^*)) = 1), \beta \rangle^*), \\ \text{or}_1(\langle \underline{m+2}, (\mu \lambda)(1(\text{or}_1(\langle \underline{m+2}, \lambda, \text{Rd}_r(\gamma, \delta) \rangle^*)) = 1), \text{Rd}_r(\gamma, \delta) \rangle^*), \dots, \\ \text{or}_1(\langle \underline{2m+1}, (\mu \lambda)(1(\text{or}_1(\langle \underline{2m+1}, \lambda, \text{Rd}_r(\gamma, \delta) \rangle^*)) = 1), \text{Rd}_r(\gamma, \delta) \rangle^*) + 1$$

The exact meaning of what was said above concerning $\text{In}_{1,r}(\beta, \gamma, \delta)$ is that for $\lambda \geq \text{In}_{1,r}(\beta, \gamma, \delta)$ $\tau_{1\lambda,r}(\beta, \gamma, \delta) = \tau_{1,r}(\beta, \gamma, \delta)$. It is also easily seen that for $\lambda \geq \text{In}_{1,r}(\beta, \gamma, \delta)$ $\text{In}_{1\lambda,r}(\beta, \gamma, \delta) = \text{In}_{1,r}(\beta, \gamma, \delta)$.

(57) $T_{1,r}(\alpha)$ = the ordinal τ such that τ_r is the configuration which follows the configuration α_r .

$$T_{1,r}(\alpha) = T_{1,r}(L_1(\alpha), L_1(L_2(\alpha)), L_2(L_2(\alpha))).$$

$$(58) \quad \text{In}_{1,r}(\alpha) = \text{In}_{1,r}(L_1(\alpha), L_1(L_2(\alpha)), L_2(L_2(\alpha))).$$

$$(59) \quad \liminf_{\gamma < \beta} f(\gamma) = (\mu\delta)(\forall\lambda < \beta)(\exists\gamma < \beta)(\gamma \geq \lambda \wedge f(\gamma) \leq \delta)$$

(60) $T_1^\beta(\alpha) = \langle \lambda, \rho \rangle^*$, where $\rho_{1\lambda}$ is the β -th configuration of the machine when it starts with the 0-th configuration α_{10} .

$T_1^\beta(\alpha)$ is obtained from

$$G(1, f, ,) = \begin{cases} \langle 0, \alpha \rangle^* & \text{if } \beta = 0 \\ \langle \max(L_1(f(p(\beta))), \text{In}_{1,1} L_1(f(p(\beta))) L_2(f(p(\beta))), \\ T_{1,1} L_1(f(p(\beta))) (L_2(f(p(\beta)))) \rangle^* & \text{if } \text{nu}(\beta) \\ \langle \sup_{\gamma < \beta} L_1(f(\gamma)), \liminf_{\gamma < \beta} L_1(L_2(f(\gamma))), \\ (\mu\delta) [\text{sq}_{0,1} \sup_{\gamma < \beta} L_1(f(\gamma))(\delta) \\ \wedge \text{Lh}_{1,1} \sup_{\gamma < \beta} L_1(f(\gamma))(\delta) = \\ (\sup_{\gamma < \beta} \text{Lh}_{1,1} L_1(f(\gamma)) L_1(L_2(L_2(f(\gamma)))) + 1 \\ \wedge (\forall \tau < \text{Lh}_{1,1} \sup_{\gamma < \beta} L_1(f(\gamma))(\delta)) (\delta_{1,1} \sup_{\gamma < \beta} L_1(f(\gamma)), \tau = \\ \liminf_{\gamma < \beta} L_1(L_2(L_2(f(\gamma))))_{1,1} \sup_{\gamma < \beta} L_1(f(\gamma)), \tau], \\ \liminf_{\gamma < \beta} L_2(L_2(L_2(f(\gamma)))) \rangle^* \rangle^* & \text{otherwise} \end{cases}$$

by recursion on f and β .

We shall now prove, by induction on β , that, for l as above and for α such that α_{110} is a configuration, $T_1^\beta(\alpha)$ is defined, that

$$L_1(T_1^\lambda(\alpha)) \leq L_1(T_1^\beta(\alpha)) \quad \text{for } \lambda < \beta \quad \text{and} \quad T_{11\sigma}^\beta(\alpha) = T_1^\beta(\alpha) \quad \text{for}$$

$$\sigma \geq L_1(T_1^\beta(\alpha)) \quad (\text{The latter holds since in } T_1^\beta(\alpha) \text{ } l \text{ was used only for}$$

arguments $< L_1^\beta(T_1(\alpha))$). By the induction hypothesis $\hat{f}_{T_1^f(\alpha)} 1\beta$ is a

total function, $L_1(T_1^\lambda(\alpha)) \leq L_1(T_1^\gamma(\alpha))$ for $\lambda < \gamma < \beta$ and

$$\hat{f}_{T_{11\sigma}^f(\alpha)} 1\beta = \hat{f}_{T_1^f(\alpha)} 1\beta \quad \text{for } \sigma \text{ such that } \sigma \geq L_1(T_1^f(\alpha)) \text{ for every } \gamma < \beta.$$

$$\text{For } \beta = 0 \quad T_1^0(\alpha) = \langle 0, \alpha \rangle^* = T_{11\sigma}^0(\alpha). \quad \text{for } \beta = \delta + 1$$

$$T_1^{\delta+1}(\alpha) = \langle \max(L_1(f(\delta)), \text{In}_{1,11L_1(f_1(\delta))}(L_2(f(\delta)))), T_{1,11L_1(f(\delta))}(L_2(f(\delta))) \rangle^*$$

where $f = \hat{f}_{T_1^f(\alpha)} 1(\delta+1)$. Since all the terms on the right hand side are

defined, by the induction hypothesis and since $T_{1,r}$, $\text{In}_{1,r}$ are defined

for the right arguments, $T_1^{\delta+1}(\alpha)$ is defined.

$$L_1(T_1^{\delta+1}(\alpha)) \geq L_1(f(\delta)) = L_1(T_1^\delta(\alpha)) \geq L_1(T_1^\lambda(\alpha)) \quad \text{for } \lambda \leq \delta. \quad \text{Let } \sigma \geq L_1(T_1^{\delta+1}(\alpha)).$$

$$\sigma \geq L_1(T_1^\delta(\alpha)) \quad \text{for } \gamma \leq \delta \quad \text{and hence } \hat{f}_{T_{11\sigma}^\delta(\alpha)} 1(\delta+1) = \hat{f}_{T_1^\delta(\alpha)} 1(\delta+1),$$

i.e., replacing l by 11σ does not effect f . Since

$$L_1(T_1^{\delta+1}(\alpha)) \geq L_1(f,(\delta)) \quad \text{we have } (11\sigma) 1 L_1(f(\delta)) = 11 L_1(f(\delta)), \quad \text{we get}$$

$$\text{In}_{1,1} L_1(f(\delta))(L_2(f(\delta))) = \text{In}_{1,(1\sigma)} 1 L_1(f(\delta))(L_2(f(\delta)))$$

$$T_{1,1} L_1(f(\delta))(L_2(f(\delta))) = T_{1,(1\sigma)} 1 L_1(f(\delta))(L_2(f(\delta))). \quad \text{How does}$$

substitution of 1σ for the first 1 in the right hand sides effect them? It does not effect them at all, by the remark which follows the

definition of $\text{In}_{1,r}(\beta, \gamma, \delta)$ and since, by definition of $T_r^{\delta+1}(\alpha)$

$$\sigma \geq L_1(T_1^{\delta+1}(\alpha)) \geq \text{In}_{1,1} L_1(f(\delta))(L_2(f(\delta))). \quad \text{If } \beta \text{ is a limit number then}$$

$$T_1^\beta(\alpha) = \langle \sup_{\gamma < \beta} L_1(f(\gamma)), \langle \liminf_{\gamma < \beta} L_1(L_2(f(\gamma))), (\mu\delta) [\text{sq}_{0,1} \sup_{\gamma < \beta} L_1(f(\gamma))(\delta) \rangle \rangle$$

$$\wedge \text{Lh}_{1,1} \sup_{\gamma < \beta} L_1(f(\gamma))(\delta) = (\sup_{\gamma < \beta} \text{Lh}_{1,1} L_1(f(\gamma)) L_1(L_2(L_2(f(\gamma)))) + 1$$

$$\wedge (\forall \tau < \text{Lh}_{1,1} \sup_{\gamma < \beta} L_1(f(\gamma))(\delta)) (\delta \text{ } 1 \sup_{\gamma < \beta} L_1(f(\gamma)), \tau =$$

$$\liminf_{\gamma < \beta} L_1(L_2(L_2(f(\gamma)))) \text{ } 1 \sup_{\gamma < \beta} L_1(f(\gamma)), \tau \rangle], \liminf_{\gamma < \beta} L_2(L_2(L_2(f(\gamma)))) \rangle^* \rangle^*,$$

where $f = \hat{\gamma} T_1^\gamma(\alpha) 1 \beta$. To check whether $T_1^\beta(\alpha)$ is defined all we have to do is to check whether there exists a δ as in the square brackets following $(\mu\delta)$.

The functional $\liminf_{\gamma < \beta} L_1(L_2(L_2(f(\gamma))))_{1,c}$ is obviously finitarily recursive,

(where f is $\hat{\gamma} T_1^\gamma(\alpha) 1 \beta$). $1 \sup_{\gamma < \beta} L_1(f(\gamma))$ is a $k \cap F_k$ " $\sup L_1(f(\gamma))$ -

constructible function, by Lemma 13(b) and the absoluteness of the function

Od_k with respect to the model Δ_k . Therefore, by Theorem 12,

$\hat{c} \liminf_{\gamma < \beta} L_1(L_2(L_2(f(\gamma))))_{1 \uparrow \sup_{\gamma < \beta} L_1(f(\gamma))}, \tau$ is a $k \cap F_k^{\sup_{\gamma < \beta} L_1(f(\gamma))}$ -

constructible function. By the regularity of Ω the sequence

$$\{ \langle \tau, \liminf_{\gamma < \beta} L_1(L_2(L_2(f(\gamma))))_{1 \uparrow \sup_{\gamma < \beta} L_1(f(\gamma))}, \tau \rangle \mid$$

$\tau < \sup_{\gamma < \beta} L_{1 \uparrow L_1(f(\gamma))} L_1(L_2(L_2(f(\gamma)))) + 1 \}$ is a $k \cap F_k^{\sup_{\gamma < \beta} L_1(f(\gamma))}$ -

constructible bounded subset of $\Omega \times \Omega$, hence, by Lemma 11, the

$k \cap F_k^{\sup_{\gamma < \beta} L_1(f(\gamma))}$ - order of this sequence is $< \Omega$. If we denote the

$k \cap F_k^{\sup_{\gamma < \beta} L_1(f(\gamma))}$ - order of this sequence with δ we can easily see that

this δ satisfies what is written inside the square brackets following $(\mu \delta)$ above, if we note that by Lemma 13(b) $1 \uparrow \sup_{\gamma < \beta} L_1(f(\gamma))$ is the cocharacteristic function of $\{ \text{Ord}_{k \cap F_k^{\sup_{\gamma < \beta} L_1(f(\gamma))}}(x) \mid x \in k \cap F_k^{\sup_{\gamma < \beta} L_1(f(\gamma))} \}$. The rest

of the induction step follows easily and is similar to what we did in the

case where $\beta = \delta + 1$.

(61) $I_1(\alpha_1, \dots, \alpha_n) \circ I_1(\alpha_1, \dots, \alpha_n)_{1 \uparrow 0}$ is the configuration of the

machine when it is in the initial state 1, the tape is $s_0 \alpha_1 \alpha_2 \dots \alpha_n s_0^{\Omega}$

and the machine scans the last square of α_n .

$$I(\alpha_1, \dots, \alpha_n) = \left\langle 1, (\mu \delta) [Lh_0(\delta) = 2 + \alpha_1 + 2 + \alpha_2 + \dots + 2 + \alpha_n + 1 \right. \\ \wedge (\forall \tau < Lh_0(\delta)) ((\delta_{0,\tau} = 0 \vee \delta_{0,\tau} = 1) \wedge (\delta_{0,\tau} = 0 \\ \leftrightarrow \tau = 0 \vee \tau = 2 + \alpha_1 \vee \tau = 2 + \alpha_1 + 2 + \alpha_2 \vee \dots \\ \vee \tau = 2 + \alpha_1 + 2 + \alpha_2 + \dots + 2 + \alpha_n))], 2 + \alpha_1 + 2 + \alpha_2 + \dots + 2 + \alpha_n \rangle^*$$

By Theorem 12 the function

$$f(\alpha_1, \dots, \alpha_n, \tau) = \begin{cases} 0 & \text{if } \tau = 0 \vee \tau = 2 + \alpha_1 \vee \tau = 2 + \alpha_1 + 2 + \alpha_2 \vee \dots \\ & \vee \tau = 2 + \alpha_1 + 2 + \alpha_2 + \dots + 2 + \alpha_n \\ 1 & \text{otherwise} \end{cases}$$

is 0-constructible and hence it follows from Lemma 11 that there is a $\mathcal{J}$ as required in the definition of $I_1(\alpha_1, \dots, \alpha_n)$ and thus $I(\alpha_1, \dots, \alpha_n)$ is defined for all numbers $\alpha_1, \dots, \alpha_n$.

$$(62) \quad En_1(\alpha_1, \dots, \alpha_n) = \text{The value of the functional } T_1^\beta(I(\alpha_1, \dots, \alpha_n))$$

for β such that the machine enters the terminal state 0 for the first time at the β -th configuration when computing the value of the function, or functional, for $\alpha_1, \dots, \alpha_n$, and undefined if the computation diverges

$$En_1(\alpha_1, \dots, \alpha_n) = T_1^{(\mu\beta)(L_1(L_2(T_1^\beta(I(\alpha_1, \dots, \alpha_n)))))(I(\alpha_1, \dots, \alpha_n))}$$

$$(63) \quad Rt_1(\alpha_1, \dots, \alpha_n) = \text{the number computed by the machine for the } n\text{-tuple } \alpha_1, \dots, \alpha_n \text{ if the computation results in a number and undefined otherwise.}$$

Without loss of generality we can assume that the machine scans the last square of the numeral of the result of the computation, since we can always

add to the machine the instruction given by $\begin{array}{c|c} 0 & 1 \\ \hline 1 & R1 \end{array}$.

$$Rt_1(\alpha_1, \dots, \alpha_n) = Rd_{1 \uparrow L_1(En_1(\alpha_1, \dots, \alpha_n))} (L_1(L_2(L_2(En_1(\alpha_1, \dots, \alpha_n))))), \\ L_2(L_2(L_2(En_1(\alpha_1, \dots, \alpha_n))))).$$

Let us denote with k^* the characteristic function of k (k is a set of ordinals)

$$(64) \quad on_{k^*}(\alpha) = Od_k(\alpha)$$

$$\text{Let } l^*(f, k^*, \alpha, \tau) = \begin{cases} 1 & \text{if } (\exists \beta < \alpha)(k^*(\beta) = 1 \wedge f(\beta) = \tau). \\ 0 & \text{otherwise} \end{cases}$$

$on_{k^*}(\alpha)$ is obtained from $or_{\hat{c}l^*(f, k^*, \alpha, \tau)}(\alpha)$, which is a f.r.f. by

Lemma 1, by recursion on f and α . The proof that $on_{k^*}(\alpha) = Od_k(\alpha)$

proceeds by induction on α . Assume $on_{k^*}(\beta) = Od_k(\beta)$ for all $\beta < \alpha$.

$\hat{c}l^*(\hat{\beta} on_{k^*}(\beta) \uparrow \alpha, k^*, \alpha, \tau)$ is therefore the cocharacteristic function of $\{Od_k(\beta) \mid \beta < \alpha \wedge \beta \in k\}$. Since $Od_k(\beta)$ is a strictly monotonic function of β

(compare [1] 9.52) we have $\{Od_k(\beta) \mid \beta < \alpha \wedge \beta \in k\} = \{Od_k(\beta) \mid \beta \in k\} \cap Od_k(\alpha)$,

hence $\hat{c}l^*(\hat{\beta} on_{k^*}(\beta) \uparrow \alpha, k^*, \alpha, \tau) = 1 \uparrow Od_k(\alpha)$. Thus

$$on_{k^*}(\alpha) = or_{\hat{c}l^*(\hat{\beta} on_{k^*}(\beta) \uparrow \alpha, k^*, \alpha, \tau)}(\alpha) = or_{1 \uparrow Od_k(\alpha)}(\alpha).$$

By Lemma 13(b) and (38) $or_1 1 Od_k(\alpha)(\alpha) = Od_{k \cap F_k} Od_k(\alpha)(\alpha)$. By Lemma

13(a) $F_k \cap F_k Od_k(\alpha)(\beta) = F_k(\beta)$ for $\beta \leq Od_k(\alpha)$ and hence

$Od_{k \cap F_k} Od_k(\alpha)(\alpha) = Od_k(\alpha)$. Thus we get $on_{k*}(\alpha) = Od_k(\alpha)$.

$$(65) \quad l_{k*}(\alpha) = l(\alpha)$$

$$l_{k*}(\alpha) = \begin{cases} 1 & \text{if } (\exists \beta \leq \alpha)(k^*(\beta) = 1 \wedge on_{k*}(\beta) = \alpha) \\ 0 & \text{otherwise.} \end{cases}$$

That $l_{k*}(\alpha) = l(\alpha)$ follows easily from $on_{k*}(\alpha) = Od_k(\alpha)$ and from $\alpha \leq Od_k(\alpha)$, which is a direct result of the monotonicity of $Od_k(\alpha)$.

Let st be the cocharacteristic function of $\bigcup_{i=0}^{m+1} k_i$. Since

$$\bigcup_{i=0}^{m+1} k_i \subseteq \max(\langle 0, \langle \Gamma, \Lambda, 2 \rangle^*, \langle \Gamma, \Lambda \rangle^* \rangle^*, \langle m, \Gamma \rangle^*, \langle m+1, \Gamma, \Gamma \rangle^*)$$

and since $m, \Gamma, \Lambda < \mathcal{O}$ the right hand side is an ordinal $< \mathcal{O}$ and therefore st is a b.e.v.f. We can call st the structure function of the machine since st contains exactly all the information on the structure of the machine, i.e., on the functions M_0, M_1 and the sets $P_1, \dots, P_m$.

$$(66) \quad k^*(st, \varphi_1, \dots, \varphi_m,) = k^*(\alpha)$$

$$k^*(st, \varphi_1, \dots, \varphi_m,) = \begin{cases} 1 & \text{if } st(\alpha) = 1 \vee \bigvee_{i=1}^m (\exists \beta, \gamma \leq \alpha)(\alpha = \langle m+1+i, \gamma, \beta \rangle^* \\ & \wedge \varphi_i(\beta) = \gamma) \\ 0 & \text{otherwise} \end{cases}$$

THEOREM 14. If Ω is regular then every general computable functional (where $f_1, \dots, f_m$ vary over total functions only) is a general recursive functional.

PROOF. If we are given an ordered machine computing the functional $F(f_1, \dots, f_m, x_1, \dots, x_n)$ then, by what was shown,

$$F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \text{Rt}_{1_{k^*(st, \varphi_1, \dots, \varphi_m)}}(\alpha_1, \dots, \alpha_n) \quad (\text{where}$$

$$1 = \hat{\alpha} 1(\alpha) \text{ and } k^*(st, \varphi_1, \dots, \varphi_m) = \hat{\alpha} k^*(st, \varphi_1, \dots, \varphi_m, \alpha).)$$

The right-hand side is a f.r.f. in $st, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n$ and hence is a g.r.f. in

$\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n$. For a discrete machine we have to repeat appropriately

the discussion in the present paragraph (from (60) and on). The main point

is the following. M_2 is defined on the set $\wp(\Gamma - \{0\})$ of all subsets of $\Gamma - \{0\}$, but is constant outside a subset of $\wp(\Gamma - \{0\})$ of cardinality

$< \Omega$. Let such a subset of $\wp(\Gamma - \{0\})$ be of the same cardinality as $\Phi < \Omega$,

then its members can be indexed by ordinals $< \Omega$. M_2 can therefore be

given by a subset U of $\Phi \times \Gamma \times \Gamma$ and an ordinal $\Psi < \Omega$ such that for

$$a \in \Gamma - \{0\} \quad M_2(a) = \beta \quad \text{if and only if} \quad (\exists \gamma < \Phi)(\forall \delta < \Gamma)(\delta \in a \leftrightarrow \langle \gamma, \delta, \beta \rangle \in U)$$

$$\vee \sim (\exists \gamma < \Phi)(\forall \delta < \Gamma)(\delta \in a \leftrightarrow \langle \gamma, \delta, \beta \rangle \in U) \wedge \beta = \Psi.$$

Thus all the information concerning M_2 can easily be incorporated in the structure function.

THEOREM 15. If Ω is regular then every numerically computable functional (where $f_1, \dots, f_m$ vary over total functions only) is a numerically recursive functional.

PROOF. If we are given an ordered numerical machine computing

$F(f_1, \dots, f_m, x_1, \dots, x_n)$ this means that the functions M_0, M_1 and the sets $P_1, \dots, P_m$ of the machine are constructible and hence, as can be

easily seen, the set $t = \bigcup_{i=0}^{m+1} k_i$, of which the structure function of the

machine is the cocharacteristic function, is constructible. As follows

immediately from [8] 2.329 every constructible set is K -constructible

for every class K . Hence t is 0 -constructible, and by Lemma 8 and 10

$$\text{Od}_0(t) = \tau < \Omega. \text{ Put } \text{st}(\alpha, \tau) = \begin{cases} 1 & \text{if } \text{Od}_0 \xi_0 \tau \\ 0 & \text{otherwise} \end{cases}$$

Thus $F(\varphi_1, \dots, \varphi_n, \alpha_1, \dots, \alpha_n) = \text{Rt}_{1_{K^*}}(\bigwedge \text{st}(\alpha, \tau), \varphi_1, \dots, \varphi_n)(\alpha_1, \dots, \alpha_n)$.

The right hand side is a f.r.f. in $\varphi_1, \dots, \varphi_n, \alpha_1, \dots, \alpha_n$, and hence,

since the machine determines a fixed τ , F is a n.r.f. in $\varphi_1, \dots, \varphi_n, \alpha_1, \dots, \alpha_n$.

If the machine is discrete we proceed similarly, noting that the function mapping the subset of $\mathcal{P}(\Gamma - \{0\})$ on Φ can be taken to be constructible (by the definition of a numerical discrete machine).

THEOREM 16. If Ω is regular then every finitarily computable functional (where $f_1, \dots, f_n$ vary over total functions only) is a finitarily recursive functional.

PROOF. If we are given a finitary machine computing $F(f_1, \dots, f_m, x_1, \dots, x_n)$ the set t of the proof of Theorem 15 can easily be seen to be a finite set

of finite ordinals, for an ordered or a discrete machine. It is also easily shown that every member of $R(\omega)$ (where $R(\alpha) = \bigcup_{\beta < \alpha} \wp(R(\beta))$, $\wp(x)$ is the power set of x) is 0-constructible of finite 0-order. Hence τ is a finite ordinal and we can substitute for τ in

$$F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \text{Rt}_{1, k^*}(\hat{\alpha} \text{ st}(\alpha, \tau), \varphi_1, \dots, \varphi_m)(\alpha_1, \dots, \alpha_n)$$

the constant function τ , which is f.r., and hence F is a f.r.f.

Thus we get that for a regular $\mathcal{R}$ the notions of a (f., n., g.) recursive functional coincide with the notion of a (f., n., g.) computable functional, as long as $f_1, \dots, f_m$ vary over total functions only. The case where $f_1, \dots, f_m$ are allowed to vary also over partial functions is not interesting since one does not see how to define a natural notion of a recursive functional in that case. Here we considered that case only for technical reasons (so as to get Lemma 1 without restrictions). For the rest of this paper we shall assume tacitly, unless otherwise stated, that in every functional or predicate $f_1, \dots, f_m$ (or $\varphi_1, \dots, \varphi_m$) vary over total functions only.

9. THE NORMAL FORM, ENUMERATION AND HIERARCHY THEOREMS FOR A REGULAR Ω .

THEOREM 17 (the normal form theorem). Let Ω be regular. (a) There exists a f.r.f. $G_{m,n}(f_0, f_1, \dots, f_m, x_1, \dots, x_n)$ such that for every g.r.f.

$F(f_1, \dots, f_m, x_1, \dots, x_n)$ there is a b.e.v.f. φ such that

$F(f_1, \dots, f_m, x_1, \dots, x_n) = G(\varphi, f_1, \dots, f_m, x_1, \dots, x_n)$. If F is a n.r.f. φ can

be taken to be finite (i.e., there is a $\beta < \omega$ such that $\varphi(\gamma) < \beta$ for every γ

and $\varphi(\gamma) = 0$ for $\gamma \geq \beta$). (b) There exists a f.r.f. $H_{m,n}(f_1, \dots, f_m, x_0, x_1, \dots, x_n)$

such that for every n.r.f. $F(f_1, \dots, f_m, x_1, \dots, x_n)$ there is an ordinal $\alpha < \Omega$

such that $F(f_1, \dots, f_m, x_1, \dots, x_n) = H_{m,n}(f_1, \dots, f_m, \alpha, x_0, x_1, \dots, x_n)$. If F

is a f.r.f. α can be chosen to be finite.

PROOF. (a) Take $G_{m,n}(f_0, f_1, \dots, f_m, x_1, \dots, x_n) = \text{Rt}_{1, k^*(f_0, f_1, \dots, f_m)}(x_1, \dots, x_n)$

For a given g.r.f. F there is an ordered machine computing it, by Theorem 4.

We take φ to be the structure function of that machine, then we get the

theorem. If F is a n.r.f. then, by Theorem 3, it is computed by an ordered

numerical machine and the structure function of such a machine is constructible.

For a f.r.f. F the proof is similar.

(b) Take $H_{m,n}(f_1, \dots, f_m, x_0, x_1, \dots, x_n) = \text{Rt}_{1, k^*(\hat{y} \text{ st}(y, x_0), f_1, \dots, f_m)}(x_1, \dots, x_n)$

(b) follows by Theorems 2, 3 and the proofs of Theorems 15 and 16.

By a (finitarily, numerically, general) recursive predicate $R(f_1, \dots, f_m, x_1, \dots, x_n)$ we mean a predicate of the form $F(f_1, \dots, f_m, x_1, \dots, x_n) = 0$, where F is a (finitarily, numerically, general) total recursive functional (i.e., F is defined for all functions $f_1, \dots, f_m$ and numbers $x_1, \dots, x_n$).

THEOREM 18 (the enumeration theorem). Let Ω be regular. (a) There exists a finitarily recursive predicate $S_{m,n}(f_0, f_1, \dots, f_m, x_1, \dots, x_n, y)$ such that for every general recursive predicate $R(f_1, \dots, f_m, x_1, \dots, x_n, y)$ there is a b.e.v.f. φ such that

$$(\exists y) R(f_1, \dots, f_m, x_1, \dots, x_n, y) \equiv (\exists y) S_{m,n}(\varphi, f_1, \dots, f_m, x_1, \dots, x_n, y). \text{ If}$$

R is numerically (resp., finitarily) recursive then φ can be chosen to be constructible (resp., finite). (b) There exists a f.r. predicate

$$T_{m,n}(f_1, \dots, f_m, x_0, x_1, \dots, x_n, y) \text{ such that for every n.r. predicate}$$

$R(f_1, \dots, f_m, x_1, \dots, x_n)$ there is an ordinal $\alpha < \Omega$ such that

$$(\exists y) R(f_1, \dots, f_m, x_1, \dots, x_n, y) \equiv (\exists y) T_{m,n}(f_1, \dots, f_m, \alpha, x_1, \dots, x_n, y).$$

If R is f.r. then α can be chosen to be finite.

PROOF. (a) We take for $S_{m,n}(f_0, f_1, \dots, f_m, x_1, \dots, x_n, y)$ the predicate

$$T_{1,k}^y(f_0, f_1, \dots, f_m) I(x_1, \dots, x_n) = 0 \text{ and for } \varphi \text{ the structure function of}$$

an ordered machine computing $(\mu y) R(f_1, \dots, f_m, x_1, \dots, x_n, y)$. The rest of the

proof is left to the reader. (b) We take for $T_{m,n}(f_1, \dots, f_m, x_0, x_1, \dots, x_n, y)$

the predicate $T_{1k}^y(\hat{z} \text{ st}(z, x_0), f_1, \dots, f_m) (I(x_1, \dots, x_n)) = 0$, and for α we

take $Od_0(t)$, where $t = \bigcup_{i=0}^{m+1} k_i$ of an ordered machine which computes

$$(\mu y) R(f_1, \dots, f_m, x_1, \dots, x_n, y).$$

THEOREM 19 (the hierarchy theorem). Let Ω be regular. (a) The predicate

$(\forall y_1)(\exists y_2)(\forall y_3) \dots (\dots y_k) \sim^k S_{m,n+k-1}(f_1, f_1, f_2, \dots, f_m, x_1, \dots, x_n, y_1, \dots, y_k)$,
 $k \geq 1, m \geq 1$ (where $\sim^k$ denotes k successive negation signs) is not of the
 form $(\exists y_1)(\forall y_2)(\exists y_3) \dots (\dots y_k) R(f_1, \dots, f_m, x_1, \dots, x_n, y_1, \dots, y_k)$, where
 R is any g.r. predicate.

(b) The predicate $(\forall y_1)(\exists y_2)(\forall y_3) \dots (\dots y_k) \sim^k T_{m,n+k-1}(f_1, \dots, f_m, x_1, x_1, x_2, \dots, x_n,$
 $y_1, \dots, y_k)$,

$k \geq 1, n \geq 1$ is not of the form

$(\exists y_1)(\forall y_2)(\exists y_3) \dots (\dots y_k) R(f_1, \dots, f_m, x_1, \dots, x_n, y_1, \dots, y_k)$, where R is
 any n.r. predicate.

The proof of Theorem 19 is left to the reader. Note that unless we
 assume that all b.e.v.f.'s are constructible we do not know how to prove
 that there is a predicate $(\forall y_1)(\exists y_2) \dots (\dots y_k) P(x_1, \dots, x_n, y_1, \dots, y_k)$,

where P is a g.r. predicate in which there are no function variables,
 such that there is no g.r. predicate R with

$$(\exists y_1)(\forall y_2) \dots (\dots y_k) R(x_1, \dots, x_n, y_1, \dots, y_k) \equiv$$

$$(\forall y_1)(\exists y_2) \dots (\dots y_k) P(x_1, \dots, x_n, y_1, \dots, y_k).$$

Except for the above mentioned analogue of the arithmetical hierarchy we have, obviously, also an analogue of the analytical hierarchy. However, here we have also an intermediate hierarchy where we consider recursive predicates quantified by quantifiers which range over b.e.v.f.'s. If we assume the axiom of constructibility then this intermediate hierarchy coincides with the analogue of the arithmetical hierarchy.

As an illustration of the problem of locating predicates in the hierarchy we shall consider the predicate $\text{card}(\alpha) \equiv \alpha \text{ is a cardinal}$.

$$\text{card}(\alpha) = (\forall \varphi)(\forall \beta < \alpha) [(\exists \gamma < \alpha)(\varphi(\gamma) \geq \beta) \vee (\exists \gamma, \delta < \alpha)(\gamma \neq \delta \wedge \varphi(\gamma) = \varphi(\delta))],$$

where φ varies over b.e.v.f.'s. If $\Omega > \omega$, then $\text{card}(\alpha)$ is not a f.r. predicate since if it were a f.r. predicate then

$$f(\beta) = (\mu \alpha)(\alpha > \beta \wedge \text{card}(\alpha)) \text{ were a f.r. partial function, but } f(\omega) = \omega_1,$$

contradicting Theorem 6. If $\Omega = \omega_{\gamma+1}$ then let φ be the b.e.v.f.

$$\text{given by } \varphi(\lambda) = \begin{cases} \omega_\lambda & \text{for } \lambda \leq \nu \\ 0 & \text{otherwise} \end{cases} \quad \text{and then } \text{card}(\alpha) \equiv (\exists \beta \leq \alpha)(\varphi(\beta) = \alpha),$$

and therefore $\text{card}(\alpha)$ is a g.r. predicate. If $\Omega = \omega_\nu$, where ν is a limit number then $\text{card}(\alpha)$ is not even a g.r. predicate, since if

$\text{card}(\alpha) \equiv F(\Psi, \alpha) = 0$, where Ψ is a b.e.v.f., then let $\tau < \nu$ be such that $\Psi(\lambda) < \omega_\tau$ for every $\lambda < \omega_\tau$ and $\Psi(\lambda) = 0$ for $\lambda \geq \omega_\tau$.

$f(\beta) = (\mu \alpha)(\alpha > \beta \wedge F(\Psi, \alpha) = 0)$. On one hand $f(\omega_\tau) = \omega_{\tau+1}$, but, on the other hand, by Theorem 6, $f(\omega_\tau) < \omega_{\tau+1}$, contradiction.

A related predicate which we consider is $\text{concard}(\alpha) \equiv \alpha$ is a constructible cardinal (i.e., there is no constructible one-one mapping of α on an ordinal $< \alpha$).

$$\begin{aligned} \text{concard}(\alpha) \equiv & (\forall \lambda)(\forall \beta < \alpha)[(\exists \gamma \leq \alpha)(\forall \delta < \alpha)([\text{or}_0(\delta), \text{or}_0(\gamma)] \varepsilon_0 \lambda \rightarrow \delta \geq \beta) \\ & \vee (\exists \gamma, \delta, \zeta < \alpha)(\gamma \neq \delta \wedge [\text{or}_0(\zeta), \text{or}_0(\gamma)] \varepsilon_0 \lambda \wedge [\text{or}_0(\zeta), \text{or}_0(\delta)] \varepsilon_0 \lambda) \\ & \vee (\exists \gamma < \alpha)(\forall \delta < \alpha)(\sim [\text{or}_0(\delta), \text{or}_0(\gamma)] \varepsilon_0 \lambda)]. \end{aligned}$$

This is indeed the required predicate since the notions of constructibility and 0-constructibility coincide (see [8] 2.329 and [5] p. 148) and since if there is a one-one constructible mapping of α on $\beta < \alpha$ then there is such a mapping of 0-order $< \Omega$, by Theorem 6. As above we can see, using the relativization of Theorem 6 to the class L_0 that $\text{concard}(\alpha)$ is not a f.r. predicate.⁸

⁸Theorems 5 and 6 of Takeuti [13] can be presented as follows. If Theorem 6 (of the present paper) would hold for some regular $\omega_\gamma > \omega_0$ also for the arithmetical function $f(\beta) = (\mu \alpha)(\alpha > \beta \wedge \text{concard}(\alpha))$ (i.e., we would have $f(\beta) < \omega_\gamma$ for $\beta < \omega_\gamma$), then ω_γ is obviously a limit cardinal in the model Δ of [1]. Since ω_γ is regular also in Δ (because if there is no sequence of length $< \omega_\gamma$ of ordinals $< \omega_\gamma$ whose limit is ω_γ there is, a fortiori, no such constructible sequence) and hence, by [8] p. 189, ω_γ is strongly inaccessible in Δ . As shown, essentially, by Mendelson [7], pp. 297-299, $F''\omega_\gamma$ is a model of set theory and we get therefore that set theory is consistent.

10. THE EQUIVALENCE OF RECURSIVITY AND COMPUTABILITY FOR A SINGULAR $\aleph$.

A remark of Takeuti on the bottom of p. 123 in [13] can be written as follows.

THEOREM 20 (Takeuti). Let $\aleph_0, \aleph$ be two cardinals such that $\omega < \aleph_0 < \aleph$.

Let $F_{\aleph}(f_1, \dots, f_m, x_1, \dots, x_n)$ and $F_{\aleph_0}(f_1, \dots, f_m, x_1, \dots, x_n)$ be two f.r.f.'s

which are given by the same definition (i.e., which are built up in the same way from schemata (1) - (8)); $F_{\aleph}$ is a functional for the numbers $< \aleph$ and $F_{\aleph_0}$ is a functional for the numbers $< \aleph_0$. For all partial functions

$\varphi_1, \dots, \varphi_m$ on $\aleph$ which are bounded by $\beta_1, \dots, \beta_m < \aleph_0$ respectively,

i.e., $\varphi_i(\alpha) < \beta_i$ for $\alpha < \aleph$, $\varphi_i(\alpha) = 0$ for $\alpha \geq \beta_i$, $1 \leq i \leq m$,

and for all $\alpha_1, \dots, \alpha_n < \aleph_0$ $F_{\aleph}(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) =$

$F_{\aleph_0}(\varphi_1^{\aleph_0}, \dots, \varphi_m^{\aleph_0}, \alpha_1, \dots, \alpha_n)$ (where $\varphi_i^{\aleph_0}$ is φ_i restricted to $\aleph_0$)

in the sense that if one side is undefined so is the other.

PROOF. By induction on the definition of F . If F is defined by one of the schemata (1) - (5) this is trivial. If F is defined by schema (6)

this follows easily from the induction hypothesis by Theorem 6. If F is

defined by schema (7) we prove the theorem by induction on $\alpha_1 < \aleph_0$. We have

$$F_{\aleph}(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = G_{\aleph}(\hat{F}_{\aleph}(\varphi_1, \dots, \varphi_m, \alpha_2, \dots, \alpha_n) \upharpoonright \alpha_1, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n).$$

By the induction hypothesis on α_1 we have

$$\hat{y}_{F_{\Omega}}(\varphi_1, \dots, \varphi_m, y, \alpha_2, \dots, \alpha_n) 1_{\alpha_1} = \hat{y}_{F_{\Omega_0}}(\varphi_1^{\Omega_0}, \dots, \varphi_m^{\Omega_0}, y, \alpha_2, \dots, \alpha_n) 1_{\alpha_1}.$$

Since $\alpha_1 < \Omega$ this function (of y) vanishes for $y \geq \alpha_1$. Let ψ be the function such that $\psi(\gamma)$ is the γ -th member of $\{\beta \mid \beta < \alpha_1 \text{ and } F_{\Omega}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) \text{ is defined}\}$ if there is such a member and 0 otherwise. Obviously $\psi(\delta) < \alpha_1$ for every $\delta < \Omega$, $\psi(\delta) = 0$ for

$$\delta \geq \alpha_1. \text{ By the definition of } \psi \sup_{\delta < \alpha_1} F_{\Omega}(\varphi_1, \dots, \varphi_m, \psi(\delta), \alpha_2, \dots, \alpha_n)$$

is defined and is equal to $\sup\{F_{\Omega}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) \mid \beta < \alpha, \text{ and } F_{\Omega}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) \text{ is defined}\}$; let us denote it with τ .

Since $\sup_{\gamma < \alpha_1} F(f_1, \dots, f_m, g(\gamma), x_2, \dots, x_n)$ is a f.o.f. we have, by Theorem 6,

$$\tau < \Omega_0. \text{ Thus } \hat{\beta}_{F_{\Omega}}(\varphi_1, \dots, \varphi_m, \beta, \alpha_2, \dots, \alpha_n) 1_{\alpha_1} \text{ is bounded by}$$

$\max(\alpha_1, \tau+1) < \Omega_0$. Therefore, by the induction hypothesis on G

$$G_{\Omega}(\hat{y}_{F_{\Omega}}(\varphi_1, \dots, \varphi_m, y, \alpha_2, \dots, \alpha_n) 1_{\alpha_1}, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$$

$$= G_{\Omega_0}(\hat{y}_{F_{\Omega_0}}(\varphi_1^{\Omega_0}, \dots, \varphi_m^{\Omega_0}, y, \alpha_2, \dots, \alpha_n) 1_{\alpha_1}, \varphi_1^{\Omega_0}, \dots, \varphi_m^{\Omega_0}, \alpha_1, \dots, \alpha_n)$$

$$= F_{\Omega_0}(\varphi_1^{\Omega_0}, \dots, \varphi_m^{\Omega_0}, \alpha_1, \dots, \alpha_n). \text{ These equalities go both ways so that}$$

if one side is defined the other one is defined too.

If F is defined by schema (8) then if

$$(\mu^{x_{n+1}}) G_{\Omega_0}(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, x_{n+1}) \text{ is defined then let its value}$$

be $\alpha_{n+1} < \Omega_0$ and, as follows easily from the induction hypothesis on G ,

$$(\mu^{x_{n+1}}) G_{\Omega}(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, x_{n+1}) = \alpha_{n+1}. \text{ If, on the other hand,}$$

$(\mu^{x_{n+1}})G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, x_{n+1})$ is defined then, by Theorem 6, its value α_{n+1} is $< \Omega_0$ and hence, using again the induction hypothesis on G , we get $(\mu^{x_{n+1}})G_0(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \alpha_{n+1}$.

THEOREM 21. Every (finitarily, numerically, general) recursive functional is (finitarily, numerically, general) computable, if we restrict the function variables $f_1, \dots, f_m$ to b.e.v.f.'s; and the functional is indeed computed by the machine assigned to do it in the proofs of Theorems 2-4.

PROOF. Let us go again through the proofs of Theorems 2-4 and see what goes wrong if Ω is not regular. We lift, temporarily, the restriction that $f_1, \dots, f_m$ are b.e.v.f.'s. The number of the states of the various machines is always $< \Omega$ as can be seen in their definitions, since by our definition of a g.r.f. the φ_i 's involved are b.e.v.f.'s. The only trouble is that for functionals defined by schemata (7) and (8) the computation may diverge for lack of time, not because the functional is not defined. For example, if the machine computes $(\mu^{x_{n+1}})G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, x_{n+1})$ the value of which is β , in order to obtain β in the computation the machine must actually go through all computations $G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, \gamma)$ for all $\gamma < \beta$. If Ω is not regular it is possible that each of these computations uses $< \Omega$ steps but that the total number of steps involved in all these computations is Ω . So the remedy on this case will be to allow more time, i.e., to allow a larger number of steps. How many steps shall we allow? If the cardinal Ω^+ which follows Ω is regular then Ω^+

steps are enough, but since we do not assume the axiom of choice we do not know whether Ω^+ is regular. However, we shall show that Ω^+ steps will indeed be enough. To do it we shall show that if $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \beta$ the computation will end in $< \Omega^+$ steps, and as shown in the proofs of Theorems 2-4 the result will indeed be β .

Since we allow now Ω or more steps we must notice that we may be confronted with a new situation in which the machine has moved right on the tape and passed its end. In this case we shall say that the tape fell off the machine and the computation diverged. How does the extension of the time effect the computation of $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ when it is undefined? Checking again the proofs of Theorems 2-4 we see that one case in which the computation diverges is where F is given by schema (8) and is undefined because there is no $\beta < \Omega$ such that $G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, \beta) = 0$, even though $G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, \beta)$ is defined for every $\beta < \Omega$. Since now the machine is given more time, after computing $G(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, \beta)$ for every $\beta < \Omega$ the tape will be $s_0^{\lambda+1} \alpha_1 \dots \alpha_n s_1^{\Omega}$. The machine will be scanning the s_0 in front of α_1 or any square to the right of it. Its state will be the initial state of End_k , $k \geq 0$. So the machine will start going right on the tape looking for $k+1$ -gaps. Since there are no $k+1$ -gaps on the tape to the right of α_1 the tape will fall off the machine after Ω more steps and the computation diverges. In all other cases where $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is undefined the computation diverges because

some subsidiary computation diverges or because the configuration remains the same for the rest of the computation; the same reasons will cause divergence even now that the time was extended.

Suppose now that $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \beta$ and consider the machine which was constructed in one of the proofs of Theorems 2-4 to compute F . By the proofs of Theorems 2-4 the machine eventually stops and gives the result β . Assume that the machine stopped at the ν -th step and let $\aleph^*$ be the least cardinal greater than ν and $\aleph$. We arithmetize the action of the machine as we did in §8, except that $\aleph$ is now replaced by $\aleph^*$ (We can also assume that the length of the tape is now $\aleph^*$ because from the proofs of Theorems 2-4 it can be easily seen that since the computation does not diverge no more than the first $\aleph$ squares of the tape will be used anyway). The only place where the regularity of

was assumed in the arithmetization of the action of a Turing machine in §8 was in (60) where

$$\sup_{\gamma < \beta} Lh_{1 \uparrow L_1}(f(\gamma))^{L_1(L_2(L_2(f(\gamma))))} =$$

$$\sup_{\gamma < \beta} Lh_{1 \uparrow L_1}(T_1^\gamma(\alpha))^{(L_1(L_2(L_2(T_1^\gamma(\alpha)))))} \quad \text{and} \quad \sup_{\gamma < \beta} L_1(f(\gamma)) = \sup_{\gamma < \beta} L_1(T_1^\gamma(\alpha))$$

were supposed to be defined. As seen in (60), (55) and (49)

$$Lh_{1 \uparrow L_1}(T_1^{\delta+1}(\alpha))^{(L_1(L_2(L_2(T_1^{\delta+1}(\alpha)))))} \quad \text{is obtained from}$$

$$Lh_{1 \uparrow L_1}(T_1^\delta(\alpha))^{(L_1(L_2(L_2(T_1^\delta(\alpha)))))} \quad \text{by addition of a number which is 0, 1 or}$$

the value of one of the φ_i 's for some argument. Also for a limit number δ

$$Lh_{11}L_1(T_1^\delta(\alpha))^{(L_1(L_2(L_2(T_1^\delta(\alpha)))))} = \sup_{\gamma < \beta} Lh_{11}L_1(T_1^\gamma(\alpha))^{(L_1(L_2(L_2(T_1^\gamma(\alpha)))))} + 1.$$

Since in our case the values of the φ_i 's are $< \mathfrak{O}$ we can prove easily, by induction, that for $\gamma < \mathfrak{O}^*$ we have

$$Lh_{11}L_1(T_1^\gamma(\alpha))^{(L_1(L_2(L_2(T_1^\gamma(\alpha)))))} \leq \mathfrak{O} \cdot \gamma + 1 < \mathfrak{O}^* \quad \text{and therefore}$$

$$\sup_{\gamma < \beta} Lh_{11}L_1(T_1^\gamma(\alpha))^{(L_1(L_2(L_2(T_1^\gamma(\alpha)))))} \quad \text{is defined and is} \leq \mathfrak{O} \cdot \beta.$$

It is immediately seen in (60) that $\sup_{\gamma < \beta} L_1(T_1^\gamma(\alpha)) =$

$$= \sup_{\gamma < \beta} In_{1,1}L_1(T_1^\gamma(\alpha))^{(L_2(T_1^\gamma(\alpha)))} \quad (\text{since the first component of } T_1^\gamma(\alpha))$$

grows only when $\text{suc}(\gamma)$, except for passing to the supremum at limit ordinals γ). As remarked above our machine operates only in the first

squares of the tape and hence if $L_2(T_1^\gamma(\alpha)) = \langle \xi, \gamma, \zeta \rangle^*$ then $\xi < \Gamma$,

$\eta_{11}L_1(T_1^\gamma(\alpha)) < \Lambda$ and $\zeta < \mathfrak{O}$. This implies, as can be easily seen in (56),

that $In_{11}L_1(T_1^\gamma(\alpha))^{(\xi, \gamma, \zeta)} < \mathfrak{O}$ and therefore $\sup_{\gamma < \beta} In_{1,1}L_1(T_1^\gamma(\alpha))^{(L_2(T_1^\gamma(\alpha)))}$

is defined and is $\leq \mathfrak{O}$. Thus in this case the arithmetization is correct,

and hence, since $\nu < \mathfrak{O}^*$, $\nu = (\mu\beta)^{(L_1(L_2(T_1^\beta)_{k^*(st, \varphi_1, \dots, \varphi_m)}^{(I(\alpha_1, \dots, \alpha_n))})))$.

Since $\varphi_i(\lambda)$, $st(\lambda) < \mathfrak{O}$ for $\lambda < \mathfrak{O}^*$, $\varphi_i(\lambda) = st(\lambda) = 0$ for $\lambda \geq \mathfrak{O}$,

we get, by Theorem 6, $\nu = (\mu^\beta)(L_1(L_2(T_{1k}^\beta(st, \varphi_1, \dots, \varphi_m)^{(I(\alpha_1, \dots, \alpha_n))}))) < \Omega^+$

and hence the computation will be finished in $< \Omega^+$ steps. If we take up again the case where $\varphi_1, \dots, \varphi_n$ are b.e.v.f.'s (with respect to Ω) then, since st is anyway a b.e.v.f., we get, by Theorem 6, that the computation ends in $< \Omega$ steps.

THEOREM 22. Every (finitarily, numerically, general) computable functional is (finitarily, numerically, general) recursive, provided we restrict the function variables $f_1, \dots, f_m$ to b.e.v.f.'s.

PROOF. The proof proceeds via the arithmetization of the action of a Turing machine like the proofs of Theorems 14-16, except that we have to use different arguments where the regularity of Ω was used in the arithmetization. As was already noted in the proof of Theorem 21 the only place in §8 where the regularity of Ω was used in (60), where

$$\sup_{\gamma < \beta} Lh_{11} L_1(T_1^\gamma(\alpha))^{(L_1(L_2(L_2(T_1^\gamma(\alpha)))))} \quad \text{and} \quad \sup_{\gamma < \beta} L_1(T_1^\gamma(\alpha)) \quad \text{were}$$

supposed to be defined. In the present case $\varphi_1, \dots, \varphi_m$ are b.e.v.f.'s so there is an ordinal number $\sigma < \Omega$ which is greater than all the values of the φ_i 's. As in the proof of Theorem 21 we get

$$Lh_{11} L_1(T_1^\gamma(\alpha))^{(L_1(L_2(L_2(T_1^\gamma(\alpha))))} \leq \sigma \cdot \gamma + 1 < \Omega \quad \text{and hence}$$

$$\sup_{\gamma < \beta} Lh_{11} L_1(T_1^\gamma(\alpha))^{(L_1(L_2(L_2(T_1^\gamma(\alpha))))} \quad \text{is defined and} \leq \sigma \cdot \beta. \quad \text{Also}$$

as in the proof of Theorem 21 we have

$\sup_{\gamma < \beta} L_1(T_1^\gamma(\alpha)) = \sup_{\gamma < \beta} \text{In}_{1,1} L_1(T_1^\gamma(\alpha)) (L_2(T_1^\gamma(\alpha)))$. If $L_2(T_1^\gamma(\alpha))$
 $= \langle \xi, \gamma, \zeta \rangle^*$ then $\zeta \leq L_2(L_2(\alpha)) + \gamma$ (since in the 0-th configuration
 α_0 , the machine scanned the square $P_{L_2(L_2(\alpha))}$ and during the γ steps
it moved at most one square right at each step). Looking at (56) and noting
that $\text{Rd}_r(\gamma, \zeta) \leq \zeta$ we see that $\text{In}_{1,1} L_1(T_1^\gamma(\alpha)) (\xi, \gamma, \zeta) \leq \max(\langle 0, \langle \Gamma, \Lambda, 2 \rangle^*,$
 $\langle \Gamma, \Lambda \rangle^* \rangle^*, \langle \mathfrak{m}, \Gamma \rangle^*, \langle \mathfrak{m}+1, \Gamma, \Gamma \rangle^*, \langle 2\mathfrak{m}+1, \sigma, \zeta \rangle^*) + 1 \leq \max(\langle 0, \langle \Gamma, \Lambda, 2 \rangle^*,$
 $\langle \Gamma, \Lambda \rangle^* \rangle^*, \langle \mathfrak{m}, \Gamma \rangle^*, \langle \mathfrak{m}+1, \Gamma, \Gamma \rangle^*, \langle 2\mathfrak{m}+1, \sigma, L_2(L_2(\alpha)) + \gamma \rangle^*) + 1$. Since
 $\langle 2\mathfrak{m}+1, \sigma, L_2(L_2(\alpha)) + \gamma \rangle^*$ is monotonic in γ we get that

$$\begin{aligned}
 \sup_{\gamma < \beta} (L_1(T_1(\alpha))) \text{ is defined and is } &\leq \max(\langle 0, \langle \Gamma, \Lambda, 2 \rangle^*, \langle \Gamma, \Lambda \rangle^* \rangle^*, \\
 \langle \mathfrak{m}, \Gamma \rangle^*, \langle \mathfrak{m}+1, \Gamma, \Gamma \rangle^*, \langle 2\mathfrak{m}+1, \sigma, &L_2(L_2(\alpha)) + \beta \rangle^*) + 1. ^9
 \end{aligned}$$

⁹Let $\mathcal{F}$ be a machine which computes a functional $F_{\mathcal{Q}}(f_1, \dots, f_{\mathfrak{m}}, x_1, \dots, x_n)$
of ordinal numbers $< \mathcal{Q}$, where $\mathcal{Q} > \omega$ and $f_1, \dots, f_{\mathfrak{m}}$ vary over b.e.v.f.'s.
If we replace the original tape of $\mathcal{F}$ whose length is $\mathcal{Q}$ by a new tape of
length $\mathcal{Q}^* > \mathcal{Q}$ and allow $\mathcal{F} < \mathcal{Q}^*$ steps in each computation then $\mathcal{F}$ will
still compute the same functional (as far as $f_1, \dots, f_{\mathfrak{m}}$ are b.e.v.f.'s
with respect to $\mathcal{Q}$ and $x_1, \dots, x_n < \mathcal{Q}$). To prove it, note that by the proofs
of Theorems 14 and 22

$F_{\mathcal{Q}}(f_1, \dots, f_{\mathfrak{m}}, x_1, \dots, x_n) = \text{Rt}_{1k^*(\text{st}, f_1, \dots, f_{\mathfrak{m}})}(x_1, \dots, x_n)$. If $\mathcal{Q}$ is replaced
by $\mathcal{Q}^*$ then the function computed by the machine is still given by the
right-hand side, which denotes now a functional on numbers $< \mathcal{Q}^*$, and

It is immediately seen that Theorems 17-19 hold also for a singular $\aleph$ only that now the function variables f_i range over b.e.v.f.'s only.

What we can say in the singular case about the functionals if we allow the f_i 's to vary over arbitrary functions is as follows. In the proof of Theorem 21 we saw that every recursive functional is computable in $< \aleph^+$ steps, and a machine which computes it is the one assigned for this task in the proofs of Theorems 2-4. On the other hand, there are recursive functionals which are not computable (in $< \aleph$ steps). E.g., we shall see that the functional $(\mu x)(\exists y < f(x))(f(x) = 2y+1)$ is not computable. Since $\aleph$ is singular there is a sequence α_λ of length $\gamma < \aleph$ such that $\lim_{\lambda < \gamma} \alpha_\lambda$ is $\aleph$. Let φ be defined by

$$\varphi(\lambda) = \begin{cases} 2\alpha_\lambda & \text{for } \lambda < \gamma \\ 1 & \text{otherwise} \end{cases}$$

Obviously $(\mu x)(\exists y < \varphi(x))(\varphi(x) = 2y+1) = \gamma$. Assume that the machine computes this functional and assume that the answer γ was obtained in $\beta < \aleph$ steps. Then $p_{\beta+1}$ (and the squares to the right of it) were not scanned by the machine during the computation. Let σ be an ordinal, $\sigma < \gamma$, such that $\alpha_\sigma > \beta+1$. Let ψ be the function defined by $\psi(\lambda) = \varphi(\lambda)$

hence the machine computes the function which is, according to the notation in Theorem 20, $F_{\aleph}(f_1, \dots, f_m, x_1, \dots, x_n)$. What we claimed follows now immediately from Theorem 20.

for $\lambda < \Omega$, $\lambda \neq \sigma$, $\psi(\sigma) = \varphi(\sigma)+1$. Now, on one hand $(\mu x)(\exists y < \psi(x))(\psi(x) = 2y+1) = \sigma$; but on the other hand $\mathcal{F}$ will give the answer δ for ψ too since $\mathcal{F}$ will not distinguish between φ and ψ , because even if during the computations $\varphi(\sigma)$ and $\psi(\sigma)$ were printed on the tape $\varphi(\sigma)$ and $\psi(\sigma)$ differ only in a square to the right of $p_{\delta+1}$ which was not scanned by $\mathcal{F}$.

We shall now notice a peculiar property of the recursive functionals.

Let α_ν , $\nu < \gamma$, be a sequence as above. Let $\varphi(\lambda) = \begin{cases} 1+\alpha_\lambda & \text{if } \lambda < \gamma \\ 0 & \text{otherwise} \end{cases}$.

Every quantifier $(\exists x)$ can be replaced by $(\exists y < (\mu z) \varphi(z))(\exists x < \varphi(y))$.

If Ω is regular or if $f_1, \dots, f_m$ are b.e.v.f.'s then Theorem 19 is true

also if we take for some of the f_i 's (other than f_1 for part (b))

fixed functions. In the case of a singular Ω this will not be true

any more if we substitute φ for one of the f_i 's.

We used unbounded functions φ in our last two examples only for the sake of convenience; we could have used also bounded functions. For example, the last φ is obviously recursive in the function χ given by

$$\chi(\lambda) = \begin{cases} 0 & \text{if } (\exists \nu)(\lambda = \alpha_\nu) \\ 1 & \text{otherwise} \end{cases}, \quad \text{which is bounded.}$$

11. THE EQUIVALENCE OF RECURSIVITY AND MACHOVER'S NOTION OF RECURSIVITY

We shall now define the notion of a functional recursive according to Machover [6] (in brief, M-recursive). Unlike Machover we shall deal only with functionals of a finite number of variables. The notion of M-recursive functionals is introduced with the aid of a formalism of Machover which is analogous to Kleene's formalism for recursive functions (see [4], Chapter XI).

PRIMITIVE SYMBOLS. "=", "′" (stroke), "sup", "0", a variable " x_n " for each n , a function letter (f.l.) f_n for each $n < \omega$.

NUMERALS. 0 followed by a sequence of length β ($< \aleph$) of strokes is the numeral for β ; we denote it with $\underline{\beta}$.

- TERMS. (a) Each numeral is a term.
- (b) A variable followed by a finite sequence of strokes is a term.
- (c) If $\{r_i\}_{i < n}$ is a sequence of terms and f is a f.l. then $f(r_0, \dots, r_{n-1})$ is a term.
- (d) If r is a term, x is a variable and y is a numeral or a variable then $\sup_{x < y} (r)$ is a term.
- (e) There are no other terms.

A numerical equation is an equation of the form $f(\underline{\beta}_0, \dots, \underline{\beta}_{n-1}) = \underline{\gamma}$.

An ascent is a sequence of numerical equations of the form

$$\begin{aligned}
 f(0, \beta_0, \dots, \beta_{n-1}) &= \gamma^{(0)} \\
 f(1, \beta_0, \dots, \beta_{n-1}) &= \gamma^{(1)} \\
 \hline
 f(\xi, \beta_0, \dots, \beta_{n-1}) &= \gamma^{(\xi)} \quad \xi < \alpha
 \end{aligned}$$

and α is said to be the length of the ascent. A term $\sup_{x < \alpha} f(x, \beta_0, \dots, \beta_{n-1})$, where x is any variable is said to correspond to the ascent above. The supremum of the ascent is the supremum of the set $\{\gamma^{(\xi)} \mid \xi < \alpha\}$

The rules of deduction are:

(R₁) To pass from an equation e to an equation d resulting from e by a simultaneous substitution of numerals for free occurrences of variables.

(R₂) To pass from an equation $r = s$ without free variables (the major premise) and a sequence of numerical equations (the minor premises) to an equation $r = s_1$, where s_1 is obtained from s by simultaneously replacing in s each part, which is the same as the left hand side of one of the minor premises, by the corresponding right hand side.

(R₃) To pass from an equation $r = s$ (the major premise) and a sequence T of ascents (the members of the members of T being the minor premises) to an equation $r = s_1$, where s_1 arises from s by simultaneously replacing each term in s which corresponds to one of the ascents of T by the supremum of that ascent.

(The s_1 in (R₂) and (R₃) are not uniquely determined, if some of the minor premises are "contradictory".)

The notion of a deduction (in tree form) of an equation from a set of equations is defined in the natural way (compare [4]), with the restriction that a deduction shall contain $< \aleph$ occurrences of symbols (and, a fortiori $< \aleph$ equations). If E is a set of equations and e an equation we say that e is deducible from E if there is a deduction of e from E ; and we denote this with $E \vdash e$.

A non-void sequence of equations in which there are $< \aleph$ occurrences of symbols is called a system of equations. If E is a system of equations and the first equation in E starts with the f.l. f then f is said to be the principal f.l. of E . The f.l.'s which occur in the right-hand sides but not in the left hand sides of the equations of E are called given f.l.'s. A system E of equations is called fully consistent if for no f.l. f and numbers $\beta_0, \dots, \beta_{n-1}, \gamma, \delta$ $\gamma \neq \delta$ and both $f(\beta_0, \dots, \beta_{n-1}) = \gamma$ and $f(\beta_0, \dots, \beta_{n-1}) = \delta$ are deducible from E . A system of equations E is said to be finitary if it is a finite sequence of finite equations (i.e., each equation contains only a finite number of occurrences of symbols). Let $\varphi_1, \dots, \varphi_m$ be functions (of a single variable) and let $g_1, \dots, g_m$ be f.l.'s. We shall denote with

$E \begin{smallmatrix} \varphi_1, \dots, \varphi_m \\ g_1, \dots, g_m \end{smallmatrix}$ the set $\left\{ g_i(\alpha) = \beta \mid 1 \leq i \leq m, \varphi_i(\alpha) = \beta \right\}$. Let

$F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ be a functional (where $\varphi_1, \dots, \varphi_m$ vary over total functions). We shall say that a system of equations E defines F if the following four conditions are satisfied.

1. E is fully consistent.
2. E has a principal f.l. F which is followed in E by sequences of terms of length n only.
3. The given f.l.'s of E form in the natural order of their indices a sequence $K_1, \dots, K_m$, and each K_i is followed in E by a single term only.
4. For any m functions $\varphi_1, \dots, \varphi_m$ and numbers $\alpha_1, \dots, \alpha_n$

$$E_{\varphi_1, \dots, \varphi_m}^{k_1, \dots, k_m} \cup E \vdash F(\alpha_1, \dots, \alpha_n) = \beta \quad \text{if and only if}$$

$$F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = \beta.$$

DEFINITION 3. The functional F is said to be finitarily M-recursive if there is a finitary system of equations which defines F . F is said to be numerically M-recursive if there is a constructible system of equations which defines F . F is said to be general M-recursive if there is a system of equations which defines F .

THEOREM 23. If $\mathcal{Q}$ is regular then every f.r.f. is a finitarily M-recursive functional.

OUTLINE OF PROOF. This is shown by induction on F , according to the 8 schemata. We shall, in each case, present a finitary system of equations, with the principal f.l. F , which defines F .

$$(1) \quad F(x_1, \dots, x_n) = x_i \quad 1 \leq i \leq n.$$

$$(2) \quad F(0) = 0', \quad F(x') = 0.$$

$$(3) \quad F(0, 0) = 0, \quad F(x', y) = 0', \quad F(x, y') = 0'.$$

$$(4) \quad \text{That } \langle \alpha, \beta \rangle \text{ is finitarily M-recursive is shown in [6].}$$

$$(5) \quad F(x) = K_1(x)$$

(6) By the induction hypothesis there are finitary systems of equations $\bar{E}, E_1, \dots, E_1$ with the principal f.l.'s $G, H_1, \dots, H_1$, respectively, which define the functionals $G, H_1, \dots, H_1$, respectively. F is defined by the system $\{F(x_1, \dots, x_n) = G(H_1(x_1, \dots, x_n), \dots, H_1(x_1, \dots, x_n))\} \cup E \cup E_1 \cup \dots \cup E_1$.

(7) Let $\bar{E}$ be a finitary system of equations which defines the functional $G(\varphi_0, \varphi_1, \dots, \varphi_n, \alpha_1, \dots, \alpha_n)$, with the principal f.l. G . We can assume, without loss of generality, that the variables $x_1, \dots, x_n$ do not occur in $\bar{E}$. Let E^* be the system of equations obtained from $\bar{E}$ by replacing each term $f(r_0, \dots, r_{k-1})$, where f is a f.l. which is not a given f.l., by the term $f(x_1, \dots, x_n, r_0, \dots, r_{k-1})$ and by replacing each term $K_0(u)$ by $H(<(u, x_1), u, x_1, \dots, x_n)$, where H and $<$ are new f.l.'s. Let $\bar{Z}$ be a finitary system of equations which defines the function $<(\alpha, \beta)$ with $<$ as principal f.l. F is defined by the system

$$\{F(x_1, \dots, x_n) = G(x_1, \dots, x_n, x_1, \dots, x_n)\} \cup E^* \cup \{H(0, u, x_1, \dots, x_n) = F(u, x_2, \dots, x_n), \\ H(1, u, x_1, \dots, x_n) = 0\} \cup \bar{Z}$$

(8) This case is dealt with in [6].

THEOREM 24. If Ω is regular then every n.r.f. F is a numerically M-recursive functional.

OUTLINE OF PROOF. In the beginning of the proof of Theorem 3 it was shown that if F is a n.r.f. then there is a f.r.f. H and fixed numbers

$\beta_1, \dots, \beta_k$ such that

$F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = H(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_k)$ for all $\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n$. By Theorem 23 H is finitarily M -recursive.

Let E^* be a finitary system of equations which defines H , with H as the principal f.l. F is defined by the system

$\{F(x_1, \dots, x_n) = H(x_1, \dots, x_n, \beta_1, \dots, \beta_k)\} \cup E^*$, which is obviously recursive.

THEOREM 25. If Ω is regular then every g.r.f. F is a general M -recursive functional.

OUTLINE OF PROOF. For every g.r.f. F there is a f.r.f. H and fixed

b.e.v.f.'s $\psi_1, \dots, \psi_k$ such that

$F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n) = H(\psi_1, \dots, \psi_k, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ for all

$\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n$. By Theorem 23 there is a finitary system of equations E

with the principal f.l. F which defines H . As shown in [6] the function

$\alpha_1 + \alpha_2$ is finitarily M -recursive, hence there is a finitary system $\bar{E}$

of equations, with $+$ as the principal f.l., which defines it. Since

$\psi_1, \dots, \psi_k$ are b.e.v.f.'s there is a number γ such that $\psi_i(\lambda) = 0$

for $\lambda \geq \gamma$, $1 \leq i \leq k$. F is defined by the system

$E \cup \{K_1(\underline{\alpha}) = \underline{\beta} \mid \psi_i(\alpha) = \beta, \alpha < \gamma, 1 \leq i \leq k\} \cup \{K_1(\gamma + x) = 0 \mid 1 \leq i \leq k\}$.

THEOREM 26. Every (finitarily, numerically, general) M -recursive functional F is (f., n., g.) recursive, with the stipulation that if Ω is singular then the function variables $\varphi_1, \dots, \varphi_m$ vary over b.e.v.f.'s only.

OUTLINE OF PROOF. Following Machover [6] we arithmetize the formalism of the M -recursive functionals. We use relative constructibility to avoid using the axiom of constructibility. The fully consistent system E of equations is

represented by a b.e.v.f. st which corresponds to the structure function of a Turing machine in §8. As in §8 k is a set of numbers and l a function which represent st as well as $\varphi_1, \dots, \varphi_m$. All the deductions from E can be assumed to be $k \cap \alpha$ -constructible for some number α , since the set of all equations deducible from a fully consistent k -constructible system E of equations by such deductions can be easily seen to be closed under the rules R_1 - R_3 . Therefore the formalism of the deductions can be arithmetized by means of recursive functions like we did for the Turing machines in §8. Thus we get that if E defines the functional $F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ then F is f.r. in $st, \varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n$ and hence, since st is a b.e.v.f., F is g.r. in $\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n$. If the system E is constructible then, as in the proof of Theorem 15, we get that F is n.r. If E is finitary then, as in the proof of Theorem 16, F is f.r.

THEOREM 27. If Ω is singular then every (f., n., g.) recursive functional F , in which the function variables $\varphi_1, \dots, \varphi_m$ are restricted to b.e.v.f.'s, is (finitarily, numerically, general) M-recursive; furthermore it is defined by the system of equations assigned to define it in the proofs of Theorems 23-25.

OUTLINE OF PROOF. Looking at the proofs of Theorems 23-25 we see that the systems E of equations assigned to define F in the regular case will also define F in the singular case if we allow deductions which contain Ω or more equations. By applying the arithmetization, and using Theorems 20 and 6, we get, in analogy to the proof of Theorem 21, that whenever

$F(\varphi_1, \dots, \varphi_m, \alpha_1, \dots, \alpha_n)$ is defined and equals β then $F(\underline{\alpha}_1, \dots, \underline{\alpha}_n) = \underline{\beta}$ is

deducible from the system $E_{k_1, \dots, k_m}^{\varphi_1, \dots, \varphi_m} \cup E$ by a deduction which contains $< \Omega$

occurrences of symbols.

B I B L I O G R A P H Y

- [1] K. Gödel The consistency of the continuum hypothesis, third printing, Princeton, 1953.
- [2] A. Hajnal On a consistency theorem connected with the generalized continuum problem, Acta Mathematica Academiae Scientiarum Hungaricae, vol. 12 (1961), pp. 321-376. See also an abstract under the same title in the Zeitschrift für mathematische Logik und Grundlagen der Mathematik, vol. 2 (1956), pp. 131-136.
- [3] A. Kino and G. Takeuti A note on predicates of ordinal numbers, Journal of the Mathematical Society of Japan, vol. 14 (1962), pp. 367-368.
- [4] S.C. Kleene Introduction to metamathematics, New York and Toronto, 1952.
- [5] A. Lévy A generalization of Gödel's notion of constructibility, Journal of Symbolic Logic, vol. 25 (1960), pp. 147-155. See also an abstract titled Indépendance conditionnelle de $V = L$ et d'axiomes qui se rattachent au système de M. Gödel, comptes Rendus des séances de l'Académie des Sciences, Paris, vol. 245 (1957), pp. 1582-1583.
- [6] M. Machover The theory of transfinite recursion (Hebrew with English summary), Ph.D. thesis, Hebrew University, Jerusalem, 1962. See also an abstract under the same title in the Bulletin of the American Mathematical Society, vol. 67 (1961), pp. 575-578.

- [7] E. Mendelson Some proofs of independence in axiomatic set theory,
Journal of Symbolic Logic, vol. 21 (1956), pp. 291-303.
- [8] J.C. Shepherdson Inner models for set theory - Part I, Journal of
Symbolic Logic, vol. 16 (1951), pp. 161-190.
- [9] J.R. Shoenfield A relative consistency proof, *ibid*, vol. 19 (1954),
pp. 21-28.
- [10] J.R. Shoenfield The independence of the axiom of constructibility,
American Journal of Mathematics, vol. 81 (1959), pp. 537-540.
See also an abstract titled Constructible sets, Summaries
of talks presented at the Summer Institute of Symbolic
Logic in 1957 at Cornell University (mimeographed), p. 214.
- [11] G. Takeuti Construction of the set theory from the theory of ordinal
numbers, Journal of the Mathematical Society of Japan,
vol. 6 (1954), pp. 198-220.
- [12] G. Takeuti On the theory of ordinal numbers, *ibid*, vol. 9 (1957),
pp. 93-113.
- [13] G. Takeuti On the recursive functions of ordinal numbers, *ibid*,
vol. 12 (1960), pp. 119-128.
- [14] G. Takeuti and A. Kino On hierarchies of predicates of ordinal numbers,
ibid, vol. 14 (1962), pp. 199-232.

UNCLASSIFIED

UNCLASSIFIED