E. Casari (Ed.)

CIME Summer Schools

Aspects of Mathematical Logic

48

Varenna, Italy 1968

Springer

FONDAZIONE CIME ROBERTO CONTI

E. Casari (Ed.)

Aspects of Mathematical Logic

Lectures given at a Summer School of the
Centro Internazionale Matematico Estivo (C.I.M.E.),
held in Varenna (Como), Italy,
September 9-17, 1968

Springer

FONDAZIONE
CIME
ROBERTO CONTI

C.I.M.E. Foundation
c/o Dipartimento di Matematica “U. Dini”
Viale Morgagni n. 67/a
50134 Firenze
Italy
cime@math.unifi.it

ISBN 978-3-642-11078-8 e-ISBN: 978-3-642-11080-1
DOI:10.1007/978-3-642-11080-1
Springer Heidelberg Dordrecht London New York

Reprint of the 1st ed. C.I.M.E., Ed. Cremonese, Roma 1969
With kind permission of C.I.M.E.

Printed on acid-free paper

Springer.com

CENTRO INTERNAZIONALE MATEMATICO ESTIVO

(C. I. M. E.)

3° Ciclo - Varenna dal 9 al 17 Settembre 1968

"ASPECTS OF MATHEMATICAL LOGIC"

Coordinatore : Prof. E. Casari

H. HERMES	:	Basic notions and applications of the theory of decidability.	pag. 1
D. KUREPA	:	On several continuum hypotheses.	" 55
A. MOSTOWSKI	:	Models of set theory	" 65
A. ROBINSON	:	Problems and methods of model theory	" 181
A. SOCHOR - B. BALCAR	:	The general theory of semisets. Syntactic models of the set theory.	" 267

CENTRO INTERNAZIONALE MATEMATICO ESTIVO

(C. I. M. E.)

H. HERMES

BASIC NOTIONS AND APPLICATIONS OF THE THEORY OF DECIDABILITY

Corso tenuto a Varenna dal 19 al 17 Settembre 1968

BASIC NOTIONS AND APPLICATIONS OF THE THEORY OF DECIDABILITY

by

H. Hermes (Freiburg- Germany)

Preliminary Remarks. The first three lectures contain an exposition of the fundamental concepts of some main theorems of the theory of recursive functions. One of the more difficult theorems of the theory of recursive functions is Friedberg-Mučniks theorem which asserts the existence of non-trivial enumerable degrees. In Lectures 4 and 5 we prove this theorem, following the treatment given by Sacks, but stressing somewhat more the combinatorial part of the proof (Lecture 4). Lecture 6 deals with problems in the theory of primitive recursive functions. As a typical example of the application of the theory of recursitivy we give in Lecture 7 in detail a proof for the unsolvability of the domino problem in the simplest case of the origin-restricted problem and show in Lecture 8 how the domino problem is connected with the $\forall\exists\forall$- case of the Entscheidungsproblem.

Lecture 6 has been given before Lectures 4 and 5. The interchange is due to systematical reasons.

The interdependece of the lectures may be indicated as follows:

1 2 3 4 5, 2 6, 2 7 8.

Contents

Lecture 1: Computability, Enumerability, Decidability

Lecture 2: - Recursiveness, Turing Machines, Degrees of Unsolvability

Lecture 3: Kleenes' Normal Form Theorem; the Jump Operator

Lecture 4: Theorem of Friedberg-Mučnik, Part I

Lecture 5: Theorem of Friedberg-Mučnik, Part II

Lecture 6: Primitive Recursive Functions

Lecture 7: The Domino Problem

Lecture 8: $\forall\exists\forall$- Case of the Decision Problem of Predicate Calculus.

Bibliography

H. Hermes

Lecture 1: Computability, Enumerability, Decidability

1. Algorithmic procedures and calculi always have been an essential part of mathematics. In the last thirty or forty years a theory has been developed in order to study the fundamental notions which are connected with this part of mathematics. Everybody knows algorithmic procedures for computing the sum of two decimals. The existence of such procedures shows that the sum-function is a computable function. If a mathematical theory T is given based on a finite number of axioms and on the rules of first-order logic calculus, we may generate one by one the theorems of T. Hence the set of theorems of T is a generable set. Using lexicographical principles it is possible to get theorems in a sequence, so that we may speak of the 0^{th}, 1^{st}, 2^{nd}, ... theorem of T. In this way we get an (effective) enumeration of T, and we call T an enumerable set. The notions of generability and enumerability may be identified. For any natural number it is decidable whether it is a prime or not. Hence the set of primes is called a decidable set.

The concepts of computability, enumerability and decidability are narrowly related (cf. no. 4). In order to be able to develop a mathematical theory concerned with these notions it is necessary to replace intuitive concepts by precise mathematically defined concepts. For each of these concepts different definitions have been proposed and proved to be equivalent to each other. Practically everybody is convinced that the precise notions correspond "exactly" to the intuitive concepts. This fact, the so-called Church's Thesis (1936), may be compared with the statement that there exists no perpetuum mobile. In the following (cf. no. 5, 6, 8 and Lecture 2) we give several precise concepts which lead to definitions of enumerability and computability. Referring to such definitions we have notions like Turing-computability, recursiveness, μ-recursiveness etc. But since these concepts can be proved to be extensionally equivalent, we later on may interchange them arbitrarily.

2. In order to compute (calculate) it is necessary to manipulate objects, i.e. to treat objects by manual means. Not every set S has the property that every element of S can be used in this way (e.g. the classical set of real numbers). A set of objects which can be used for computation may be called a set of manipulable objects. Typical example for manipulable objects are the words composed of letters from a given finite alphabet A. If A has only one element, these words may be identified with the natural numbers. An infinite set S of manipulable objects is denumerable. If S_1 and S_2 are two (infinite) sets of manipulable objects there exists a 1 - 1 mapping f from S_1 onto S_2 which is effective in both ways, i.e. : if any x S_1 is given it is possible to compute f(x), and if any y S_2 is given it is possible to compute $f^{-1}(y)$. Such a mapping is often called a Gödelization, especially if S_2 coincides with the set of natural numbers (in this case f(x) is called the Gödel number of x). In principle it is irrelevant on which (infinite) set of manipulable objects the theory is based. Very often (following Gödel) we choose for this purpose the set of natural numbers. But many applications may be much easier if other sets are chosen. - We speak of an enumerable or of a decidable set S only if a fixed set S' of manipulable objects is given and if $S \subset S'$.

3. For most questions concerning computability if is irreleveant whether we consider 1-place or n-place functions (or similarly 1-place or n-place predicates). Let us consider e.g. n=2 . It may be easily shown that there exist computable functions σ_2, σ_{21}, σ_{22}, s.t.

(1.1) $\sigma_2(\sigma_{21}(x), \sigma_{22}(x)) = x$ for each natural number x

(1.2) $\sigma_{21}(\sigma_2(x,y)) = x$ for each pair x, y of natural numbers

(1.3) $\sigma_{22}(\sigma_2(x,y)) = y$

Using these functions we may associate with every 2-place function f a 1- place function g, defined by

(1.4) $g(x) = f(\sigma_{21}(x), \sigma_{22}(x))$.

Now we get

(1.5) $f(x, y) = g(\sigma_2(x, y))$.

As far as questions of computability are concerned we may replace f by g.

4. The following statements hold intuitively:

(1.6) A set is enumerable iff it is void or the range of a computable function.

(1.7) A 1-place function is computable, iff the 2-place relation R is enumerable, where R holds for y and x iff y=f(x).

(1.8) A set S is decidable iff S and its complement are enumerable.

(1.9) A set S is decidable iff its characteristic function f is computable. f (x) has the value 0 or 1 according as $x \in S$ or $x \notin S$.

5. Here and in no. 6 we give two definitions of the notion of enumerability. Here we are concerned with sets whose elements are words over a finite alphabet.

Let be given four mutually disjoint alphabets A, B, C, D. The elements of A are called constants, the elements of B variables, the elements of C predicates. With each predicate is associated a natural number $n \geq 1$ as its place number. $D = \{ ; , \rightarrow \}$. The words over A called proper words, the words over $A \cup B$ terms. If P is an n-place predicate and $t_1, \ldots, t_n$ are terms, then $Pt_1;\ldots;t_n$ is called an atomic formula. If $p_1, p_2, \ldots$ are atomic formulas, then the words $p_1, p_1 \rightarrow p_2$, $p_1 \rightarrow p_2 \rightarrow p_3$ etc. are called formulae. Rule 1 permits the transition from a formula F to a formula G by substituting a proper word for a variable, Rule 2 the transition from an atomic formula p and a formula $p \rightarrow F$ to the Formula F. A formal system (Smulyan) is given by a finite set ϕ of formulae. A formula is derivable in a formula system ϕ, if it can be obtained by applications of Rule 1 and/ or Rule 2, starting with the elements of ϕ. An n-place relation R between words over a finite alphabet A_o is enumerable (in the sen-

se of Smullyan) iff there is a formal system ϕ, belonging to the alphabets A, B, C, D, where $A_o \subset A$, and an n-place predicate P, s.t. for each n-tuple $w_1, \ldots, w_n$ of words over A_o the formula $Pw_1;\ldots;w_n$ is derivable in ϕ iff R holds for $w_1, \ldots, w_n$.

6. Another way to define enumerable relations is given by Fitch's minimal logic. We start with the 3-letter alphabet $\{(,), *\}$. A word over this alphabet is called an expression if it coincides with $*$, or if it may obtained, starting with $*$, by the rule which permits to go over from words a and b to the word (a, b). $(* (**))$ is an example for an expression. Take the set of all expressions as the underlying set of manipulable objects. We choose certain expressions and call them $=, \neq, \wedge, \vee, \bigvee, 1_1, 1_2, 1_3, \ldots$. With these expressions are connected certain rules. We confine ourselves here to indicate the rules connected with $=$, $\vee$, $\bigvee$, and 1_1:

(1.10) For each expression a we may write down the expression =aa(this is an abbreviation for ((=a)a) (parentheses to the left, also in the following).

(1.11) For all expressions a, b we may go over from a to $\vee ab$.

(1.12) For all expressions a, b we may go over from b to $\vee ab$.

(1.13) For all expressions a b we may go over from ab to $\bigvee a$.

(1.14) For all expressions a, b, c, d, where a is variable, and d the result of substituting c for a in b, we may go over from d to $1_1 a b c$.

(These rules are similar to rules of logic, hence "minimal logic".) An expression is called derivable if it can be obtained by the rules. E.g. the derivation $(* (**))$, $(\bigvee *), \bigvee\bigvee)$ shows, that $(\bigvee\bigvee)$ is derivable. A relation T between expressions is (Fitch-) enumerable iff there is an expression r s.t. for each n-tuple $a_1, \ldots, a_n$ of expressions, the expression $ra_1 \ldots a_n$ is derivable iff R holds for $a_1, \ldots, a_n$.

7. The last example shows that the enumerable sets (of expressions) are manipulable themselves, because they may be given by expressions, and each expression determines such an enumerable set. Unfortunately we do not have this pleasant fact for the computable functions. In order to show this let us assume that we have an enumerable set S of words s.t. (a) each element of S determines effectively a unary computable function and that (b) each such function may be given in this way (think of the elemets of S being descriptions of the computing processes). Then we get a contradiction as follows : We get in a effective way for each n a prescription how to compute a certain unary function f_n. We introduce a new function f by postulating that $f(n) = f_n(n)+1$. According to our assumptions there is an m s.t. $f=f_m$. This leads to a contradiction for the argument m. (A diagonal argument of this kind is often used in the theory of recursive functions).

It is possible to remedy this defect by enlarging the set of functions hitherto considered. Until now we only have admitted total functions. The domain of an n-ary total function consists of all n-typles of objects in question. We now consider partial functions. The domain of an n-ary total function consists of an n-ary partial function does not necessarily have all n-tuples as elements, it may even be void. Intuitively a partial function is called computable, if there is a procedure which terminates for a given argument iff the function has a value for this argument which determines in that case that value.

With partial function we do not get the contradiction of no. 7. It is only possible to conclude that f is not defined for the argument m.

If we admit also partial functions, the statement (1.7) remains true. (1.6) may be simplified :

(1.15) A set is enumerable iff it is the range (or the domain) of a computable partial function.

8. There are different important precise definitions for computability for partial functions. For Turing-computability and μ-recursivity cf. Lecture 2. Here we mention only the concept of Markov's algorithm. Let be given a finite alphabet A and words A_i, B_i (i=1,...,p) over A. A Markov's algorithm is given by sequence

(1.16) $$A_i \rightarrow (.) B_i \qquad (i=1,\ldots,p),$$

where "(.)" indicates that there may be a dot behind the arrow or not. (1.16) determines a unary partial function. f . The domain and range of f are contained in the set of all words over A. For any word W over A we determine uniquely a sequence $W=W^{(0)}, W^{(1)}, W^{(2)},\ldots$ of words. f is defined iff the sequence terminates, and in that case f(W) is the last element of the sequence.

If $W^{(n+1)}$ is defined we will have a uniquely determined number p_{n+1} $(1\leq p_{n+1}\leq p)$, which describes in the sequence (1.16) the rule which is responsible for the transition from $W^{(n)}$ to $W^{(n+1)}$.

We call a word K a part of L iff there are words K_1, K_2 s.t. $L=K_1KK_2$. Given K, there may be different decompositions of L of this kind. If K_1 has minimal length, the decomposition of L is uniquely determined and called the normal decomposition. We now procede to define $W^{(n+1)}$ and p_{n+1}:

$W^{(n+1)}$ and p_{n+1} are only defined if there is an i s.t. A_i is a part of $W^{(n)}$ and if n=0 or (n>0 and the p_n^{th} term of (1.16) has no dot). In this case let be p_{n+1} the smallest i, s.t. A_i is a part of $W^{(n)}$. Let be $W^{(n)}=K_1A_iK_2$ the normal decomposition of $W^{(n)}$ relative to A_i. Now $W^{(n+1)} = K_1B_iK_2$.

A unary partial function g(whose domain and range is contained in the set of all words over a finite slphabet) is called computable by a Markov's algorithm over an alphabet A, iff $A_o \subset A$ and if for each word W

over A_o(a) if f (the function determined by this algorithm) is defined for W if f(W) is a word over A_o, then g is is defined for W and g(W)=f(W), and (b) if g is defined for W then also f is defined for W and again f(W)= =g(W).

REFERENCES: Davis [1], Hermes [1] (also for the minimal logic of Fitch).

Kleene [1], [2], Markov [1], Rogers [1], Smullyan [1].

Lecture 2: μ- Recursiveness, Enumerability, Decidability.

1. In no. 1 we use natural numbers as manipulable objects. Let be C_0^0 the 0-place function with value 0, S^1 the 1-place successor-function and U_i^n the n-ary function whose value coincide with the i-th argument ($i = 1, \ldots, n$). The functions C_0^0, S^1, U_i^n are called initial functions. The initial functions are computable total functions.

The process of substitution leads from function $g, h_1, \ldots, h_r$ to a function $f = g(h_1, \ldots, h_r)$, where

$$f(x_1, \ldots, x_n) = g(h_1(x_1, \ldots, x_r), \ldots, h_r(x_1, \ldots, x_n)) . \tag{2.1}$$

Substitution preserves totality and computability.

The process of primitive recursion leads from functions g, h to a function f, where

$$f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n), \tag{2.2}$$

(2.3) $$f(x_1, \ldots, x_n, S^1(y)) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y)).$$

Primitive recursion preserves totality and computability.

The process of application of the μ-operator leads from a function g to a function f, where

(2.4) $$f(x_1, \ldots, x_n) = \mu y \; g(x_1, \ldots, x_n, y) = 0.$$

(μy = the least y. $\mu y \; g(x_1, \ldots, x_n, y) = 0$ is defined iff there is a y s.t. for all $z < y$ $g(x_1, \ldots, x_n, z)$ is defined and $\neq 0$ and if $g(x_1, \ldots, x_n, y) = 0$; in this case $y = \; y \; g(x_1, \ldots, x_n, y) = 0$. Application of the μ operator preserves computability but in general not totality. Computability would not generally be preserved if we would not postulate that $g(x_1, \ldots, x_n, z)$ be defined for $z < y$.

The μ-operator may also be applied to a relation R. We define $\mu y \; Rx_1 \ldots x_n y$ by $\mu y \; g(x_1, \ldots, x_n y) = 0$, where g is the characteristic function of R.

The functions which may be obtained starting with the initial functions and using substitution and primitive recursion are called primitive recursive functions. If we admit in addition the application of the μ-operator we get the μ-recursive functions. Every primitive recursive function is total and computable. The μ-recursive functions coincide with the computable (partial) functions. There are total μ-recursive functions which are not primitive recursive.

2. Turing machines. A TM (Turing machine) M is given by (a) a finite (ordered) alphabet $a_0, \ldots, a_N$, (b) a finite (ordered) set of states $q_0, \ldots, q_P$ and a finite (ordered) set of quadruples. (The set of quadruples is often called the table of M.) A quadruple is of the form $q\,a\,b\,q'$, where q, q' are states, a is a letter (of the alphabet), and b either a letter or one of the symbols R ("right") or L ("left"). (We assu-

me, that the set of letters, the set of states and the set $[R, L]$ are mutually disjoint.) We require that there is at most one quadruple in the set of quadruples which begins with qa, where q is a fixed state and a is a fixed letter.

A tape is a two-way infinite sequence of squares. (Sometimes also one-way infinite tapes are considered.) An inscription I is a mapping of the set of squares into the alphabet. We assume that in I almost every square is blank (empty), i.e. mapped on the void letter a_0. A (complete) configuration C is a triple $C = (I, s, q)$, where I is an inscription, s is a square and q a state. We write $C = (I(C), s(C), q(C))$. s(C) is called the scanned square of C, and $a(C) = I(C)(s(C))$ the letter on the scanned square of C. C is called terminal, if there is no quadruple in M which begins with q(C), a(C). Otherwise we associate with C in a unique way a configuration C', called the successor of C: let be q(C) a(C) b q^* the quadruple of M which begins with q(C) a(C). We want to define $C' = (I', s', q')$. We put $q' = q$. I' and s' depend on b. We distinguish two cases:

(1) If be is a letter, we puts $s' = s$ and $I' = I$ with the (possible) exception that $I'(s') = b$.

(2) If $b = R(L)$, s' is the right (left) neighbor of s and $I' = I$.

If we start with an arbitrary C, we obtain in a unique way a sequence C, C', C" = (C')', C"', ..., which may be finite or infinite. "To start with C" may be expressed as" to apply M on I(C) in s(C)", if $q(C) = q_0$. "To proceed from C to C' "may be expressed as printing, if we have case (1), and as going to right (left), if we have case (2). "M halts, starting with C, at C^*" means, that the sequence C, C', ... has a last terminal term which is C^*.

3. With each Turing machine M and each natural numer n we associate an n-ary partial function f_M^n. (We assume in the following

that the alphabet of M has at least the letters a_0 (blank) and a_1.) Let be $x_1, \ldots, x_n$ natural numbers and $C^M_{x_1, \ldots, x_n}$ a configuration with $q(C^M_{x_1 \ldots x_n}) = q_0$, $I(C^M_{x_1 \ldots x_n})$ an inscription where the arguments $x_1, \ldots, x_n$ are represented by sequences of $(x_1+1), \ldots, (x_n+1)$ consecutive squares which bear the letter a_1 and are separated from each other by one blank square, and $s(C^M_{x_1 \ldots x_n})$ is the first square on the tape which bears the letter a_1. For any configuration C let be $v(C)$ (the value of C) the number squares which bear the letter a_1 in $I(C)$. Now we introduce f^n_M as follows: $f^n_M(x_1, \ldots, x_n)$ is defined iff there is a C^* s.t. M, starting with $C^M_{x_1, \ldots, x_n}$, halts at C^*; in this case $f^n_M(x_1, \ldots, x_n) = v(C^*)$. An n-ary function f is called Turing computable if there is a Turingmachine s.t. $f = f^n_M$.

4. Relative computability. Mathematicians not only are interested in the question whether a partial function f is computable, but also in the question whether a partial function f is computable if we suppose that some other partial function g is computable (where g may be computable or not). This question has a positive answer iff there is a procedure to get the values of f under the assumption that the values of g are given. It is not assumed that the values of g are given by any effective procedure. If f is computable under the assumption that g is computable, we call f computable relative to g or g-computable (g-recursive) and write $f \leq g$. As an example we have $g^2 \leq g$ for every function g.

A precise definition for the relation $f \leq g$ runs as follows: We have $f \leq g$ for exactly those functions f which may be obtained

starting with the initial functions (cf. no. 1) and g, using substitution, primitive recursion and the application of the μ-operator.
It is obvious that $\leq$ is reflexive and transitive. Hence the relation defined by ($f \leq g$ and $g \leq f$) divides the class of all (partial) functions in mutually disjoint subclasses. These classes are called degrees (of unsolvability). Let be $\underline{f}$ the class to which f belongs. It is possible to define $\underline{f} \leq \underline{g}$ by $f \leq g$. Hence the set D of degrees is a partially ordered set. D has a least element 0 which consists of all (μ-) recursive functions. Not in every degree we leave a total function (Medvedev 1955, cf. Rogers [1]). Degrees which have total functions as elements are called total degrees. A total degree may be identified with the set of all total functions belonging to it. The set T of total degrees is not only partially ordered but in addition a semi-upper-lattice: Let be f, g total unary functions and $h(x, y) = \mathfrak{S}_2(f(x), g(y))$. Then $\underline{h}$ is the least upper bound of $\underline{f}$ and $\underline{g}$ with respect to $\leq$.

5. If g is a unary total function the relation $f \leq g$ may be defined using an extended concept of Turing machines. Here we admit also quadruples $q\ a\ b\ q'$ where b is a state (cf. no. 2). If a configuration C is not terminal (no. 2) we associate with C as its successor a configuration C'_g depending on g. C'_g is determined by the quadruple which begins with $q(C)$, $a(C)$ (cf. no. 2). Let be $q\ a\ b\ q'$ this quaduple. If b is not a state we define $C'_g = C'$ (no. 2). If $b = q''$ we define $C'_g = (I(C), s(C), q^*)$ where $q^* = q'$ if $g(\mathfrak{S}_{21}(v(C)))) = \mathfrak{S}_{22}(v(C)),$ and $q^* = q''$ otherwise. Hence in order to get C'_g one has in general to ask an "oracle" (Turing) about the value of g for a certain argument.

With each Turing machine (of the extendet kind), each function g and each natural number n we associate an n-ary function $f^n_{M,g}$. The definition runs like the definition of f^n_M in no. 2, 3, but with

C'_g in place of C' .

An n-ary (partial) function is called Turing computable relative to g iff there is a Turing machine (off the extended kind) s.t. $f=f^n_{M,g}$.

6. It is often convenient to identify a predicate (set, relation) with its characteristic function (which is a total function). Hence we may speak of the degree of a relation and extend $\leq$ to relations.

REFERENCES : See Lecture 1.

Lecture 3: Kleene's Normal Form Theorem; the Jump Operator.

1. With each total degree d it is possible to associate a degree d' (the jump of d) which has the property that $d < d'$ (but d' is not an upper neighbor of d). To prepare the definition of d' we introduce Kleene's normal form for g-recursive total functions.

2. In the following we assume that we have mutually disjoint infinite sequences $a_0, a_1, a_2, \ldots$ and $q_0, q_1, q_2, \ldots$, and that the alphabets and states of any Turing machine are initial segments of these sequences (which is not a serious restriction). A Turing machine M (of the extended

kind, cf. Lecture, no. 5) may be described by its Godel number G(M). We assume in addition that also finite sequences $C_0, C_1, \ldots, C_m$ of configurations are described by Gödel numbers $G(C_0, \ldots, C_m)$. The following constructions depend on the (fixed) Gödelization G.

We want to introduce a unary total function U and for each n an (n+2)-place predicate T^n_g (which depends on a total function g) in order to describe the function $f^n_{M,g}$ (cf. also Lecture 2, nos. 2, 3 ,5) . Let be

$$(3.1)\quad U(g) = \begin{cases} \text{the value } v(C_m) \text{ if there is a sequence of configurations} \\ \quad C_0, \ldots, C_m \text{ s.t. } g = G(C_0, \ldots, C_m), \\ 0 \quad \text{otherwise.} \end{cases}$$

(3.2) $T^n_g z x_1 \ldots x_n y$ iff there are a Turing machine M and configurations $C_0, \ldots, C_m$ s.t. $z = G(M)$, $y = G(C_0, \ldots, C_m)$, $C_0 = C^M_{x_1 \ldots x_n}$, $C_{j+1} = (C_j)'_g$ (' depending on M) and C_m is terminal.

Now we repeat the definition of $f^n_{M,g}$ (Lecture 2, no. 5) by writing

$$(3.3)\quad f^n_{M,g}(x_1, \ldots, x_n) = U(\ \mu y T^n_g G(M) x_1 \ldots x_n y) \ .$$

This is Kleene's Normal Form Theorem. It shows that each n-ary g-recursive function may be represented by $U(\mu y T^n_g z x_1 \ldots x_n y)$ with suitable z. It is easy (but somewhat tedious) to show that U is a recursive function and T^n_g a g-recursive predicate (i.e. its characteristic function is g-recursive (g-computable)). From this we infer that $U(\mu y T^n_g z x_1 \ldots x_n y)$ is an n-ary g-recursive (partial) function for every z. Hence, varying the number $z = 0, 1, 2, \ldots$ we get every n-place g-recursive function.

In addition it can be shown that in order to obtain U and the characteristic function of T^n_g starting with the initial functions and the fun-

ctions g, (cfr. Lecture 2, no. 4) it is not necessary to apply the μ-operator (U and T^n_g are "g-primitive recursive"). Hence (3.3) shows that we get every g-recursive function by applying the μ-operator at most once.

For later application it is convenient to notice that

(3.4) M, starting with C^M_x, halts iff $\forall y T^1_g G(M)xy$

Finally we remark that for the usual choice of $\mathfrak{G}_{21}$ and of $G(C_0, \ldots, C_m)$ we have

$$\mathfrak{G}_{21}(v(I(C_i)) < G(C_0, \ldots, C_m) \quad (i = 0, \ldots, m-1) .$$

This shows that in order to check whether $T^1_g zxy$ holds or not, the oracle for g is asked only for arguments which are less than y.

3. For each total unary function g we define

(3.5) g' = characteristic function of the unary predicate $\forall y T^1_g xxy$.

We want to show that

(3.6) $$g \leqslant g' ,$$

(3.7) $$\underline{g} \neq \underline{g}' \text{ (hence with (3.6) } \underline{g} < \underline{g}'),$$

(3.8) $$f \leqslant g \rightarrow f' \leqslant g' .$$

Using (3.6), (3.7), (3.8) we may extend the operator '(jump) to elements of T(total degrees) by defining $(\underline{f})' = (\underline{f'})$. It follows that $d < d'$.

4. We obtain (3.7) by proving (Church):

(3.9) $$g' \not\leqslant g.$$

Otherwise let be $g' \leq g$. We introduce a total function h by

$$h(x) = \begin{cases} U(\mu y T_g^1 xxy)+1, & \text{if this is defined for x,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

Under our assumption $g' \leq g$ we find h is g-computable. Hence according to no. 2 there is a number s.t. $h(x) = U(\mu y T_g^1 zxy)$ for every x. We get as a special case $h(z) = U(\mu y T_g^1 zzy)$, which contradicts (3.10).

5. In order to obtain (3.6) and (3.8) we prove the following Theorem (Kleene). Let be g a unary and total function, R a 2-place relation and $R \leqslant g$. Then there is a computable total unary function r s.t.

$$\forall y\, Rxy \text{ iff } \forall y T_g^1\, r(x)\, r(x)\, y\ . \tag{3.11}$$

Proof: For each number x it is possible to construct effectively an extended TM M(x) s.t. starting with $C_t^{M(x)}$ (where t is an arbitrary natural number) we get a sequence $C_0 = C_t^{M(x)}$, $C_1 = (C_0)'_g$, $C_2 = (C_1)'_g, \ldots$ of configurations s.t. the following statements hold: There is a k s.t. $I(C_k)$ is void. For a number $k_0 > k$ we have $C_{k_0} = C_{x,0}^{M(x)}$. Using the assumption $R \leqslant g$, M now "checks" whether Rx0 or not. If not, there will be a $k_1 > k_0$ s.t. $C_{k_1} = C_{x,1}^{M(x)}$. Now M "checks" whether Rx1 or not, etc. If there exists no y s.t. Rxy, M(x) does not halt. But if there a y s.t. Rxy, M(x) will halt. Hence we have

$$M(x), \text{ starting with } C_t^{M(x)}, \text{ halts iff } \forall y\, Rxy. \tag{3.12}$$

Now let be $r(x) = G(M(x))$. r is a computable total function. From (3.4) we infer:

$$M(x), \text{ starting with } C_t^{M(x)}, \text{ halts iff } \forall y\ T_g^1 G(M(x)))\, ty\ . \tag{3.13}$$

Comparing (3.12) and (3.13) we get (introducing r(x))

(3.14) $\bigvee y\ {}^{\prime}Rxy\ ;$ iff $\bigvee y T^1_g r(x)ty$,

which gives (3.11) for $t = r(x)$.

6. We now apply Kleene's Theorem in order to prove (3.6) and (3.8).

Proof of (3.6): We introduce R by postulating

$$R\ x\ y \quad \text{iff} \quad g(\mathfrak{S}_{21}(x)) = \mathfrak{S}_{22}(x) \wedge y = y.$$

It is obvious that $R \leqslant g$. Hence according to Kleene we have a computable total function r s.t.

(3.15) $\bigvee y R\ x\ y$ iff $\bigvee y T^1_g\ r(x)\ r(x)\ y$.

The left side is equivalent to $g(\mathfrak{S}_{21}(x)) = \mathfrak{S}_{22}(x)$. Hence from (3.15) we obtain $g < g'(r)$, and trivially $g'(r) \leqslant g'$.

Proof of (3.8) : Let be $f \leqslant g$. We define Rxy by $T^1_f xxy$. $R \leqslant g$, since $T^1_f \leqslant f \leqslant g$. Using Kleene's theorem we have

$$\bigvee y\ T^1_f xxy \quad \text{iff} \quad \bigvee y T^1_g\ r(x)\ r(x)\ y\ ,$$

which shows that $f' = g'(r) < g'$.

7. The upper-semi-lattice T (lecture 2, no. 4) with the additional jump-operator is a very complex structure which has been intensively studied. I want to mention only two results:

(1) Every countable partially ordered set is imbeddable in T.

(2) The complete degrees (e.g. the degrees of the form d') coincide with the degrees $\geq 0'$.

8. Of special interest are the degrees of enumerable sets. These degrees are called enumerable degrees. About the enumerable degrees we have the following elementary facts:

(a) 0 is an enumerable degree, since every decidable set belongs to 0 and every decidable set is enumerable.

(b) Going back to the intruitive notion of enumerability it is easy to see that every enumerable set may be expressed in the form $\forall y\, Rxy$, where R is decidable. Conversely each set of this form with decidable R is enumerable. If is a computable total function (e.g. $f = S^1$) then $T^1_f xxy$ is decidable. Hence $\forall y T^1_y xxy$ is enumerable. This shows that 0' is an enumerable degree (cf. (3.5)).

(c) As we have seen in (b), each enumerable set S may be expressed in the form $\forall y\, Rxy$ with decidable R. Using Kleene's Theorem (3.11) for a computable total unary function g, we obtain the result that

$$\text{degree of } S = \underline{g'(r)} < \underline{g'} = 0'.$$

We have shown that 0 and 0' are enumerable degrees and that for every enumerable degree we have $0 \leq d \leq 0'$. Post (1944) has asked whether there are enumerable degrees other then $0, 0'$. This question has been answered (positively) not before 1966/7. Cf. Lecture 4 and 5.

REFERENCES: See Lecture 1. - Kleene and Post [1], Post [1]

H. Hermes

Lecture 4: Theorem of Friedberg-Mučnik, Part I

1. The Friedberg-Mucnik Theorem answers in the affirmative the question whether there are enumerable degrees besides 0 and 0' (cf. Lecture 3, no. 8) . We follow the treatment of Sacks who tries to separate a combinatorial part of the proof (which he calls "prority method") from the rest which uses recursive concepts. This lecture is devoted to the combinatorial part. The proof is finished in the next lecture . For other proofs cf. the references.

2. The individuals considered here are natural numbers. Let be E, F, F' unary and H, D binary predicates, and g a total unary function (whose arguments and values are natural numbers). We introduce the following abbreviations :

(4.1) Lrs for $0<r<s \wedge \neg F' r \wedge Fr \wedge \neg Hrs - 1$,

(4.2) Ps for $\forall r(gr)<g(s) \wedge Lrs \wedge Drs)$,

(4.3) Qs for $\forall r(g(r) = g(s) \wedge Lrs)$,

(4.4) ϕsk for $0<s \wedge \neg F' s \wedge Fs \wedge g(s) = k$,

(4.5) ψsk for $\forall r(r<s \wedge \phi rk \wedge \neg Hrs - 1 \wedge Hrs)$,

(4.6) $\phi(k)$ for $[s: \phi rk]$,

(4 7) $\psi(k)$ for $[s: \psi sk]$.

The derivations in this lecture are based on the following

Axioms:

A1: $\neg F' s \wedge Fs \leftrightarrow \neg Es$

A2: $\neg Hrs - 1 \wedge Hrs \rightarrow Drs \wedge \neg Es$

A3: $Hss \rightarrow Es$

A4: $Ps \rightarrow Es$

A5: $Qs \rightarrow Es$

A6: $\neg Fs \rightarrow Ps \vee Qs \vee Hss\text{-}1$

A7: $Hrs - 1 \rightarrow Hrs$

We first prove several lemmata. The most important are Lemma 4 and Lemma 5 which relate the predicates ϕ and ψ. These lemmata are used to derive Lemmata 8 and 9 which show that for each k the sets $\phi(k)$ and $\psi(k)$ are finite. This immediately leads to Lemma 10. Axioms 6 and 7, not used hitherto (and no other axiom) will be used in order to derive Lemma 11. In the next lecture we apply only Lemmata 10 and 11.

3. Lemmata 1 to 5.

Lemma 1. $\phi sk \rightarrow \neg Es$.

Proof: Axiom 1.

Lemma 2: $\psi sk \rightarrow \neg Es$.

Proof: Axiom 2.

Lemma 3. $\phi sk \rightarrow \neg Hss$.

Proof: Lemma 1, Axiom 3.

Lemma 4: $r < s \wedge \phi rk \wedge \phi sk \rightarrow \vee u(r<u<s \wedge \psi uk)$.

Proof: From ϕsk we get $\neg Qs$ (Lemma 1, Axiom 5). Using ϕrk and ϕsk we have $g(r) = g(s)$. Hence $\neg Qs$ gives $\neg Lrs$. Since we have $0 < r < s$, $\neg F'r$ and Fr (from ϕrk), we get $Hrs\text{-}1$. From ϕrk we obtain $\neg Hrr$ (Lemma 3). Comparing $\neg Hrr$ and $Hrs\text{-}1$ we find that there is a number u s.t. $r < u < s$, $\neg Hru\text{-}1$, Hru. This together with ϕrk gives ψ_{uk}.

Lemma 5: $\psi sk \rightarrow \vee i(i<k \wedge \phi si)$.

Proof: Using the definition of ψ_{sk} we have a number r s.t. r<s, ϕrk, $\neg$ Hrs-1, Hrs. Let be i = g(s). Remember that k = g(r) (from ϕrk).

If we assume $i \leq k$ we get a contradiction: Then we have g(r)=g(s) and Lrs (using ϕrk and $\neg$ Hrs-1). Hence we get Qs and, with Axiom 5, Es. But we also have $\neg$Es by Lemma 2.

If we assume k<i we get a contradiction: Then we have g(r) < g(s), Lrs (as above) and Drs (using Axiom 2). Hence we get Ps and (with Axiom 4) Es. But have $\neg$Es by Lemma 2.

Hence we have i < k. From ψsk we get $\neg$Es (Lemma 2), then $\neg$F's and Fs (Axiom 1). Since 0 < s and g(s) = i we have ϕ si. -

4. Lemmata 6 to 9.

Lemma 6. $\psi(k) \subset \bigcup_{i<k} \phi(i)$.

Proof: Lemma 5.

Lemma 7. card $\phi(k) \leq$ card $\psi(k)+1$.

Proof: According to Lemma 4 between two numbers r, s with r < s, ϕrk and ϕsk, there is a number u s.t. ψuk. Hence if $\phi(k)$ is infinite then also $\psi(k)$ is infinite. If $\phi(k)$ is finite let be $\phi(k) = \{s_o, \ldots, s_n\}$ with $s_o < s_1 < \ldots < s_n$. Then we have numbers $u_1, \ldots, u_n$ s.t. $s_o < u_1 < s_1 < u_2 < s_2 < \ldots < u_n < s_n$ s.t. $\psi u_1 k, \ldots, \psi u_n k$. Hence card $(k) + 1 \geq n+1 =$ card $\phi(k)$.

Lemma 8. card $\psi(k) < 2^k$.

Proof by induction: (a) (0) is void (cf. Lemma 5). Hence card $\psi(0) < 2^o$. (b) Let Lemma 8 be true for all i < k. Then we have

$$\begin{aligned} \text{card } \psi(k) &\leq \sum_{i \subset k} \text{card } \phi(i) \text{ (Lemma 6)} \\ &\leq \sum_{i \subset k} (\text{card } \psi(i) + 1) \text{ (Lemma 7)} \\ &\leq \sum_{i \subset k} 2^i \text{ (induction hypothesis)} \end{aligned}$$

$< 2^k$.

Lemma 9. $\text{card}\,\phi(k) \leq 2^k$

Proof: Lemmata 7, 8.

Lemma 10: $\wedge k \vee u \wedge r \wedge i\,(u<r \wedge i \leq k \rightarrow \neg\phi ri \wedge \neg\psi ri)$.

Proof: The gist of Lemmata 8, 9 is the fact that $\phi(k)$ and $\psi(k)$ are finite for each k. From this we infer immediately Lemma 10.

Lemma 11: Let be

(1) $g(s) = k$,

(2) $\neg F' s$, $\neg Hss-1$,

(3) $u < s$,

(4) $\wedge r \wedge i(u < r \wedge i \leq k \rightarrow \neg\phi ri \wedge \neg\psi ri)$,

(5) $\wedge r(r \leq u \rightarrow \neg Drs)$.

Then we have

$$\vee r(\phi rk \wedge \wedge m \neg Hrm)$$

Proof: From (4) we get for r = s and i = k that $\neg\phi sk$. Looking at the definition (4.4) we find that $\neg Fs$. Hence $Ps \vee Qs \vee Hss-1$ (by Axiom 6).

Hss-1 is excluded by (2).

Ps can be excluded as follows: Assume Ps. Then there is a number r, s.t. $0 < r < s$, $F' r$, Fr, $\neg Hrs-1$, $g(r) < g(s)$, Drs. Let be $i = g(r)$. Then we have ϕri. Now $i = g(r) < g(s) = k$. Using (4) we get $r \leq u$. Hence $\neg Drs$ by (5), contradicting Drs.

Therefore we have Qs. By definition of Q we have a number r, s.t. $0 < r < s$, $\neg F'r$, Fr, $\neg Hrs-1$, $g(r) = g(s) = k$. For this r we have ϕrk. We want to show that $\neg Hrm$ for every m.

We have $\neg Hrs - 1$. Hence by Axiom 7 we have $\neg Hrm$ for $m<s-1$. If now Hrm for some m, we have a number $m \geq s \geq r$ .s.t $\neg Hrm-1$ and Hrm. Therefore we would have ψmk (by definition (4.5)). Now (4) shows that $\neg Hrm$ which contradicts Hrm.

References: Friedberg [1], Mucnik [1] , Lachlan [1] , Sacks [1] , Shoenfield]

H. Hermes

Lecture 5: Theorem of Friedberg-Mučnik, Part II

1. We want to show that there are enumerable degrees d_0, d_1, s.t. $d_0 \not\leqslant d_1$ and $d_1 \not\leqslant d_0$. Since for every enumerable degree d we have $0 \leqslant d \leqslant 0'$ (cf. Lecture 3, no. 8) neither d_0 nor d_1 can be 0 or $0'$.

In this lecture every set (relation, function) has natural numbers as elements (arguments, values). Small letters refer to natural numbers. For each s we will define sets A^0_s, A^1_s st. the binary relations $x \in A^0_s$ and $x \in A^1_s$ are recursive (i.e. decidable). Now we introduce the sets A^0, A^1 by

$$A^0 = \bigcup_s A^0_s \quad , \qquad A^1 = \bigcup_s A^1_s \, . \tag{5.1}$$

From the equivalence $x \in A^0$ iff $\forall s\, x \in A^0_s$ we infer that A^0 is enumerable (cf. Lecture 3, no. 8) . The same holds for A^1.

The definitions of A^0_s and A^1_s will be interrelated. We later on define sets T_s s.t. $x \in T_s$ is a binary recursive relation and introduce A^0_s, A^1_s by

$$n \in A^0_s \text{ iff } 2n \in T_{s-1}, \qquad n \in A^1_n \text{ iff } 2n+1 \in T_{s-1}. \tag{5.2}$$

2. The sets T_s will be defined together with sets F^s, H^s and a function $g(s)$ by simultaneous recursion. We introduce the following abbreviations:

$$\text{F's for } F^s \subset T_{s-1}, \qquad \bar{F}_s \text{ for } F^s \subset T_s \, , \tag{5.3}$$

$$\text{Hrs for } H^r \cap T_s \neq 0, \qquad \text{Drs for } H^r \cap F^s \neq 0, \tag{5.4}$$

$$\text{Es for } T_s = T_{s-1} \, . \tag{5.5}$$

The definitions of F^s, H^s, T_s and g(s) are given by (5.6) for the case $s = 0$. For the case $s>0$ we distinguish Case 1 where we have the definitions (5.9), (5.10), (5.11) and Case 2 where we have the definition (5.12) for F^s, H^s, and g(s). In both cases T_s is given by (5.13) and (5.14).

(5.6) $$F^0 = H^0 = T_0 = 0 \text{ (void set)}, \quad g(0) = 0.$$

In order to define F^s, H^s and g(s) for $s>0$ we suppose that F^{s-1}, H^{s-1} and g(s-1) are defined. Hence also A^0_s and A^1_s are given by (5.2). Let be f^0_s, f^1_s the characteristic functions of A^0_s, A^1_s. Let be

(5.7) $\varepsilon(s) = 0$ if s is even, $\varepsilon(s) = 1$ if s is odd,

(5.8) e(s) = the number of prime factors 3 occuring in the prime number representation of s.

Consider the following condition (where p is the k^{th} prime number) :

(*) $$\bigvee m \bigvee y \, (m \leq s \wedge y \leq s \wedge T_{f^{\varepsilon}_s}(s) \quad e(s) \quad p^m_{e(s)} \, y) \text{ and } U(y) = 1.$$

We have Case 1 if (*) is satisfied, otherwise Case 2. In Case 1 let be r(s) the greatest m for which $\bigvee y \, (\ldots)$. Then we define :

(5.9) $$F^s = \{ 2p^{r(s)}_{e(s)} + 1 - \varepsilon(s) \},$$

(5.10) $$H^s = z : \bigvee n(n \leq s \wedge n \notin A^{\varepsilon(s)}_s \wedge z = 2n+\varepsilon(s))$$

(5.11) $$g(s) = 2e(s) + \varepsilon(s) + 1.$$