# Theory of Recursive Functions and Effective Computability

Hartley Rogers, Jr.

## Theory of Recursive Functions and Effective Computability

Hartley Rogers, Jr.

Massachusetts Institute of Technology

McGraw-Hill Book Company

New York St. Louis San Francisco Toronto London Sydney

revenues to the first of the section of the particular,

# THEORY OF RECURSIVE FUNCTIONS AND EFFECTIVE COMPUTABILITY

Copyright © 1967 by McGraw-Hill, Inc. All Rights Reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Library of Congress Catalog Card Number 65-25921

53522

1234567890 MAMM 7432106987

### pho Y we Vi was good to de till wer Doll mising los as bar Preface

In addressing the American Mathematical Society in 1944, E. L. Post concluded, "Indeed, if general recursive function is the formal equivalent of effective calculability, its formulation may play a role in the history of combinatory mathematics second only to that of the formulation of the

concept of natural number."†

This book may be viewed as a progress report on some of the ideas and hopes expressed in Post's paper. The subject of recursive functions had been studied by a number of researchers prior to 1944. In particular, Church, Kleene, and Turing, as well as Post himself, had already made major contributions. Although Post's paper concerns only one part of a larger subject, it marks an epoch in that subject, not only for the specific problems and methods that it presents, but also for the emphasis that it places on intuitive naturalness in basic concepts.

Post's remark, quoted above, is undoubtedly extravagant; and one may well question whether the kind of calculability represented by general recursive functions will possess a theory of much practical usefulness. (We discuss this further in Chapter 1.) Nevertheless, this book is intended to be a partial vindication of Post's remark and of his attitude. The book presents its subject matter in semiformal terms, with an intuitive emphasis that is, hopefully, appropriate and instructive. The use of semiformal procedures in recursive function theory is analogous to the use, in other parts of mathematics, of set-theoretical arguments that have not been fully formalized. It is possible for one who possesses a good grasp of the simple, primitive ideas of our theory to do research, just as it is possible for a student of elementary algebra in school to do research in the theory of natural numbers.

Since 1944, and especially since 1950, the subject of recursive function theory has grown rapidly. Many researchers have been active. The present book is not intended to be comprehensive or definitive.. Moreover, its informal and intuitive emphasis will prove, in some respects, to be a limitation. Certain important parts of the theory (e.g., the study of various proper subclasses of the general recursive functions) and certain interesting applications (e.g., the identification of recursively unsolvable problems in other parts of mathematics) cannot, by their nature, be treated without more extensive and detailed formalism. Several works already exist to supply this

<sup>†</sup> Recursively enumerable sets of positive integers and their decision problems. Bulletin of the American Mathematical Society, vol. 50 (1944), pp. 284-316.

defect, and the reader is urged to use them as a more formal supplement to the present text. One of these is Kleene's Introduction to metamathematics, D. Van Nostrand Company, Inc., Princeton, N.J., 1952; another is Davis's Computability and unsolvability, McGraw-Hill Book Company, New York, 1958. These works, while valuable as a supplement, are not a prerequisite for this book.

The book is in sixteen chapters. Chapters 1 and 2 present the basic concept of partial recursive function and give simple examples of the unsolvability phenomenon. Chapters 3 and 4 summarize and characterize the theory to be developed. Chapter 5 gives further basic concepts. Chapters 6 to 10 present results on reducibilities and degrees of unsolvability (concepts emphasized in Post's 1944 paper). Chapter 11 presents the recursion theorem, a fundamental tool in the theory. Chapters 12 to 16 present certain major areas of current research activity. A more detailed outline of the book is given in Chapter 3. The sale of the book is given in Chapter 3.

In most chapters, the final section gives exercises. The exercises appear in an order parallel to the order of topics in the chapter. They fall into three categories: unmarked, marked with a triangle  $(\land)$ , and (rarely) marked with a solid triangle (A). The first category may be solved in a straightforward manner with a proper understanding of the text. Secondcategory exercises are somewhat more difficult. Third-category exercises are still more difficult or else require, for easy solution, knowledge of results and methods not yet presented. Hints are supplied for many of the exercises (a triangle or solid triangle applies to the exercise without its hint). Solid-triangled and more difficult triangled exercises may be discussed at a later point in the text. Special topics, augmenting the text, are occasionally covered in exercises and may (though only rarely) be referred to in later portions of the text. To ear od T ... ovitourstant bas etsingorges .vilabaged

The reader is urged to make every effort to do the unmarked and triangled exercises. If an exercise has a hint, he should make a first attempt to solve the exercise without reading the hint. Similarly, in the text itself, he should attempt to prove theorems, whenever possible, without first reading the proof in the text. As in other mathematics, there is no better way to develop insight, research ability, and an appreciation of the difference between genuinely new and fruitful ideas on the one hand, and routine, though interesting, extensions of old ideas on the other.

The literature of recursive function theory has grown rapidly in recent years. The bibliography given in this book is far from complete. Only a few formal citations are made for 1965 and after, although results from this period are described in the text.

Some of the material in this book was first presented in courses and seminars at Massachusetts Institute of Technology. I am particularly indebted to Noam Chomsky and Burton Dreben for their patient encouragement, and to other members of these courses and seminars for a variety of help and

suggestions that I cannot hope to acknowledge in detail. I owe a special debt of gratitude to Mary Marto and Marsha Scherr for patient and loyal work in typing and preparing the manuscript, and to Katharine Kumar for help with the index and final stages of proof. I am also grateful to Ann Singleterry, James Geiser, and Leslie Tharp for help with the manuscript and to Patrick Fischer, Carl Jockusch, Donald Kreider, and Warren Teitelman for a particularly careful reading of portions of the text. Others who have provided comments, suggestions, and help of great value have included: J. Addison, Y. Bar-Hillel, D. Bobrow, M. Blum, J. Conger, J. Denton, H. Enderton, R. Friedberg, L. Hodes, C. Kent, D. Knutson, A. Lévy, D. Luckham, J. Lukas, J. MacIntyre, D. Martin, T. McLaughlin, M. Minsky, Y. Moschovakis, R. Parikh, D. Park, H. Putnam, W. Quine, M. Rabin. H. Rice, W. Ritter, J. Rosenthal, G. Sacks, C. Shannon, J. Shoenfield. C. Spector, J. Stillwell, J. Ullian, H. Wang, C. Yates, and P. Young. Numerous other colleagues and students have given important help. These persons, named and unnamed, have supplied many ideas, but they have not had an opportunity to criticize the final text.

I owe a special intellectual debt to James Dekker, John Myhill, and Norman Shapiro. Shapiro introduced me to recursive function theory. The methods and attitudes of Dekker and of Myhill have been a major

influence on the present work.

A portion of the material in the present text appeared, in somewhat different order and form, as Theory of recursive functions and effective computability, volume I, published in mimeographed form by the Mathematics

Department of Massachusetts Institute of Technology in 1957.

This book would not exist but for support provided me by the National Science Foundation through the Mathematics Department at Massachusetts Institute of Technology under grants G-19992, GP-379, GP-2496, and GP-6982. I am deeply grateful for the time that this support has made possible.

Hartley Rogers, Jr.

## Introduction: Prerequisites and Notation

This book is intended for use as a senior undergraduate or first-year graduate text. It assumes a knowledge of basic set-theoretical terminology and techniques such as might be obtained in an undergraduate course in modern algebra. In most parts of the book, no knowledge of logic is assumed, but the reader will find some knowledge of logic helpful. We shall use a few notations from elementary logic, as described below.

The literature of recursive function theory unfortunately lacks a single, commonly used terminology and notation. We present our choice of basic terminology and notation in this introductory section. The reader is urged to give this section a careful, preliminary reading and then to return to it from time to time as may later prove necessary. Initial effort by the reader here will, hopefully, be rewarded with clarity and facility in the main text.

For the most part, we deal with nonnegative integers, sets of nonnegative integers, and mappings from nonnegative integers to nonnegative integers. Unless specifically indicated otherwise, we use the words number and integer to mean nonnegative (rational) integer. N is the set of all integers. A, B, C, . . . (Latin capitals, early in the alphabet) denote subsets of N.  $\emptyset$  is the empty set. x, y, z, . . . (Latin lowercase, late in the alphabet) denote members of N, i.e., integers.

We use the following set-theoretical notations. A=B means that A and B are identical as sets, i.e., have the same members.  $x\in A$  means that x is a member of A.  $\{\ |\ \}$  is the notation to indicate set formation.  $\{x|\cdots x\cdots \}$  is the set of all x such that the expression  $\cdots x\cdots$  is true when "x" is interpreted as the integer x. The universe from which a set is formed is indicated by the style of symbol appearing before the vertical bar. Thus  $\{x|\cdots \}$  must be a set of integers.

 $A \cup B$  is the union of A and B, that is,  $\{x | x \in A \text{ or } x \in B \text{ or both}\}$ .  $A \cap B$  is the intersection of A and B, that is,  $\{x | x \in A \text{ and } x \in B\}$ .  $\overline{A}$  is the complement of A, that is,  $\{x | not \ x \in A\}$ .  $A \subset B$  means that A is a subset of B; that is to say, for all x, if  $x \in A$  then  $x \in B$ .  $A \supset B$  means that  $B \subset A$ . A is a proper subset of B if  $A \subset B$  and not A = B. (Thus  $A \subset B$  and  $B \subset A$  imply A = B.)

We occasionally denote a finite set by an expression in braces listing its members, in any order. For example  $\{2,5,3\}$  is the set of the first three primes. We sometimes suggest certain infinite sets by a "listing" in braces. For example,  $\{0,2,4,\ldots,2n,\ldots\}$  is the set of even integers.

Given x and y,  $\langle x,y \rangle$  is the ordered pair consisting of x and y in that order. Similarly  $\langle x_1,x_2,\ldots,x_k \rangle$  is the ordered k-tuple consisting of  $x_1,\ldots,x_k$  in that order.  $A \times B$  is the cartesian product of A and B, that is,  $\{\langle x,y \rangle | x \in A \text{ and } y \in B\}$ . Similarly

 $A_1 \times A_2 \times \cdots \times A_k = \{\langle x_1, \ldots, x_k \rangle | x_1 \in A_1 \text{ and } \cdots x_k \in A_k \}.$  The cartesian product of A with itself k times is denoted as  $A^k$ . P, Q, R,  $\ldots$  (Latin capitals, late middle alphabet) will stand for relations on N, i.e., subsets of  $N^k$  for some k > 0. If  $R \subset N^k$ , R is called a k-ary relation.

Let R be any k-ary relation. We say that R is single-valued if, for every  $\langle x_1, \ldots, x_{k-1} \rangle$ , there exists at most one z such that  $\langle x_1, \ldots, x_{k-1}, z \rangle \in R$ . If R is single-valued, we say that its domain is  $\{\langle x_1, \ldots, x_{k-1} \rangle | \text{there is a } z \text{ such that } \langle x_1, \ldots, x_{k-1}, z \rangle \in R\}$ , and we call this collection domain R. Clearly a single-valued k-ary relation may be viewed as a mapping from its domain into N.

For this reason, instead of saying that R is a single-valued k-ary relation, we shall sometimes say, synonymously, that R is a partial function of k-1variables. (Here the word "partial" suggests that the domain of R may not be all of  $N^{k-1}$ .) We shall use  $\varphi, \psi, \ldots$  (Greek lowercase, late alphabet) to denote partial functions, and we shall often use ordinary functional notation with these symbols; thus  $\varphi(x,y) = z$  will mean that  $\langle x,y,z \rangle \in \varphi$ . The reader should keep in mind that, fundamentally, a partial function is to be construed as a relation. (We thus identify a partial function with its "graph.") We shall most often be concerned with the case k = 2, i.e., partial functions of one variable, and partial function will mean partial function of one variable unless otherwise indicated. With partial functions, the functional notation can result in ambiguity. For example, the assertion that  $\varphi(x)$  not = y might mean that  $\langle x,y \rangle$  not  $\in \varphi$ , or it might mean that there is a z such that  $\langle x,z\rangle \in \varphi$  and z not = y. Our intended meaning in such situations will always be clear. If  $\varphi$  is a partial function, we say that  $\varphi$  is defined (or convergent) at x if  $x \in domain \varphi$ ; otherwise  $\varphi$  is undefined (or divergent) at x. (Similarly for partial functions of more than one variable.)

In case a partial function of k variables has all of  $N^k$  as its domain, we call it a function, or, occasionally for emphasis, a total function. We use  $f, g, h, \ldots$  (Latin lowercase, early middle alphabet) to denote functions. As before, f(x) = y will mean  $\langle x, y \rangle \in f$ .

a, B, C, . . . (script capitals, early alphabet) denote either sets of sub-

sets of N or sets of relations on N.

The range of a partial function  $\varphi$  of k variables is  $\{z \mid there \ exist \ x_1, \ldots, x_k \ such that <math>\langle x_1, \ldots, x_k, z \rangle \in \varphi \}$ , and we denote it range  $\varphi$ . Members of range  $\varphi$  are called values of  $\varphi$ . If  $\varphi(x_1, \ldots, x_k) = y$ , y is called the value of  $\varphi$  corresponding to argument  $\langle x_1, \ldots, x_k \rangle$ . A partial function is onto if its range = N. A partial function  $\varphi$  is one-one if, for every y, there is at most one k-tuple  $\langle x_1, \ldots, x_k \rangle$  such that  $\varphi(x_1, \ldots, x_k) = y$ .  $c_A$  will

be the characteristic function of the set A; hence  $c_A(x) = 1$  if  $x \in A$ , and  $c_A(x) = 0$  if x not  $\in A$ . Occasionally, a set A is represented by a (nonunique) function f such that  $A = \{x | f(x) = 0\}$ . Such an f is called a

representing function for A.

Let [-x-] be an expression such that given any integer in place of "x," the expression defines at most one corresponding value. (For example, the expression " $x^2 + x$ " defines a value for every integer; while the expression "least proper prime divisor of x" defines a value for integers which are not prime and are different from 1.) Then  $\lambda x[-x-]$  denotes the partial function:  $\{\langle x,y\rangle|[-x-]$  defines the value y when "x" is interpreted as the integer x}. This is Church's lambda notation for defining partial functions. For example, given  $\varphi_1$  and  $\varphi_2$ , then  $\lambda x[\varphi_1(x) + \varphi_2(x)]$  is the partial function  $\psi$  such that domain  $\psi = domain \varphi_1 \cap domain \varphi_2$  and  $\psi(x) = \varphi_1(x) + \varphi_2(x)$ for all x in  $domain \psi$ . We also use the lambda notation for partial functions of k variables, writing  $\lambda x_1 x_2 \cdot \cdot \cdot x_k$  in place of  $\lambda x$ .

If  $\psi$  and  $\varphi$  are partial functions,  $\psi\varphi$  indicates their composition, i.e., the partial function  $\{\langle x,y\rangle | \text{there is a z such that } \langle x,z\rangle \in \varphi \text{ and } \langle z,y\rangle \in \psi\}.$ (Note the reversed order of  $\varphi$  and  $\psi$  so that  $\psi\varphi(x)$  can be expressed as  $\psi(\varphi(x))$ .) Other common notations include  $\psi^{-1} = \{\langle y, x \rangle | \langle x, y \rangle \in \psi\}$ ;  $\psi(A) = \{y | \text{for some } x, x \in A \text{ and } \psi(x) = y\}; \psi^{-1}(A) = \{x | \text{for some } y, \}$ 

 $y \in A \text{ and } \psi(x) = y$ .

For binary (that is, 2-ary) relations, the terms transitive, reflexive, equivalence, linear ordering, partial ordering will be given their usual meanings (which we do not define here). Partial orderings will sometimes be

strict (<) and sometimes nonstrict ( $\le$ ).

We use certain notations and conventions from elementary logic: "and" will sometimes be abbreviated as "&"; "or" will be used in the inclusive (and/or) sense and will sometimes be abbreviated as "V"; "⇒" will abbreviate "only if"; "⇔" will abbreviate "if and only if"; "¬" will abbreviate "not" and will be placed before the statement which it negates. We sometimes combine "¬" with "∈" or "=" as "∉" or "≠." Brackets will be used to indicate grouping of statements, except that (contrary to logical usage but following common mathematical usage) " $(1) \Rightarrow (2) \Rightarrow \cdots \Rightarrow$ (n)" (where (1), (2), . . . , (n) are statements) will abbreviate "[[(1)  $\Rightarrow$ (2)] & [(2)  $\Rightarrow$  (3)] &  $\cdots$  [(n - 1)  $\Rightarrow$  (n)]]," and "(1)  $\Leftrightarrow$  (2)  $\Leftrightarrow$   $\cdots$   $\Leftrightarrow$  (n)" will abbreviate "[[(1)  $\Leftrightarrow$  (2)] & [(2)  $\Leftrightarrow$  (3)] &  $\cdots$  [(n - 1)  $\Leftrightarrow$  (n)]]." " $\forall$ " and "3" are called universal and existential quantifier symbols, respectively. " $(\forall x)$ " is read: "for all x." " $(\exists x)$ " is read: "there exists an x such that . . .". Groups of symbols such as " $(\forall x)$ " and " $(\exists x)$ " are called, respectively, universal and existential quantifiers.

The above logical symbols serve as convenient abbreviations of ordinary mathematical language. For example, the meaning of  $A \subset B$  can be expressed  $(\forall x)[x \in A \Rightarrow x \in B]$ , and the definition of domain  $\varphi$  can be

expressed  $\{x | (\exists y) [\varphi(x) = y] \}$ .

 $\aleph_0$  is the cardinality of N.  $2^N$  is the set of all subsets of N; we sometimes call this collection  $\mathfrak{A}$ .  $2^{\aleph_0}$  denotes the cardinality of  $2^N$ , i.e., the cardinality of the continuum.  $\mu x[\cdots x\cdots]$  is the least integer x such that the expression  $\cdots x\cdots$  is true when "x" is interpreted as the integer x, if this least integer exists.

Various other general and special notations will be introduced as the

book progresses.

Both logic and recursive function theory lack a universally accepted system of notation. Our choice of logical abbreviations is not uncommon. A choice of notation for recursive function theory presents some difficulties, especially in a treatment that covers a variety of areas. Part of the current literature uses Greek lowercase letters for sets. We do not do so because of the almost universal use of these symbols for ordinals. The reader should note that by number-theoretic predicate, Kleene and others mean relation on N. The reader should also note that some writers use  $f, g, \ldots$  to denote partial functions in general, rather than only total functions, and that in a significant part of the literature, Greek lowercase letters early in the alphabet are used for total functions.

In the assertion of a theorem or lemma, unless otherwise specified, we follow the usual mathematical convention that all unquantified variable symbols (representing integers, sets, or relations) are to be taken as operated on by unexpressed universal quantifiers standing at the beginning of the

entire assertion.

In the course of a proof, certain variable symbols may appear as universal variables, e.g., "Let x be any integer such that · · · ." Other variable symbols may be introduced as temporary names, e.g., "Choose  $x_0$  to be some fixed integer greater than x . . . . "† In general, but not always, we shall use subscripted symbols in the latter case. (Subscripted symbols may also be used in the former case when sufficiently many distinct variable symbols are required.) Occasionally, for emphasis, we shall also use letters in the middle of the Latin lowercase alphabet, i, j, k, m, n, . . . , with or without subscripts, as temporary names of integers. Occasionally, for emphasis, we shall use letters late in the Latin capital alphabet, X, Y, Z, as universal variables for sets. In this respect our usage will not always be consistent, but it will be clear from the context whether we adhere to these conventions or deviate from them.

Chapters are divided into sections. Thus §7.4 is the fourth section of Chapter 7. Theorems receive roman numerals, beginning anew in each chapter. Theorem 7-VI is the sixth theorem in Chapter 7. Exercises receive arabic numerals. Exercise 7-14 is the fourteenth exercise at the end of Chapter 7. When reference is made to a theorem or exercise occurring

<sup>†</sup> In the terminology of elementary logic, a universal variable is a variable symbol introduced by universal specification, and a temporary name is a variable symbol introduced by existential specification (see Suppes [1957]).

xix

in the same chapter, the chapter number may be omitted. The conclusion

of a proof is indicated by the symbol X.

Reference to entries in the bibliography is made by giving the author's name and a bracketed date. The item will be found under that name and beside that date in the bibliography.

此为试读,需要完整PDF请访问: www.ertongbook.com

#### Contents

		Definitions seducibility seducibility		vii
		Preface monoda nisadi.		
		Introduction: Prerequisites and Notation of the overlapped and Production of the Overlapped and Pro		xv
Cha	pter 1	RECURSIVE FUNCTIONS		1
		Finite sets		1
	§1.1	The informal notion of algorithm		5
	§1.2	An example: the primitive recursive functions		9
	§1.3	Extensionality		10
	§1.4	Diagonalization Formal characterization		11
	§1.5 §1.6	The Basic Result		
	§1.0 §1.7	Ohamah'a Thagia		20
	§1.8	C" 1-1 have emissioned iter a month conom	1,88	21
	§1.9	The halting problem		24
***	§1.10	Recursiveness		26
	81.10	1000 (410) 1 024000		
		ONE-ONE REDUCTBILLTY: MANY-ONE REDUCTBIL		
Cha	pter 2	UNSOLVABLE PROBLEMS		32
	§2.1	Further examples of recursive unsolvability	§7.1	32
	§2.2	Unsolvable problems in other areas of mathematics		35
	§2.3	Existence of certain partial recursive functions		36
	§2.4	Historical remarks		38
	§2.5	Discussion		39
	§2.6	Exercises		40
	511.	8 Constructive ordinals seemevisouborq	7.73	
Chapter 3		PURPOSES; SUMMARY		46
	§3.1	Goals of theory		46
	§3.2	Emphasis of this book		48
	§3.3	Summary		48
		2 Decomposition ates elquid		
		Immune sets stee evision S.		
Chapter 4		RECURSIVE INVARIANCE villidious benefit at the T		50
ind	piel 4	Troth table reducibility and many-one reducibility	\$8.4	100
	§4.1	Invariance under a group method sides-distribution		50
	84 2	Recursive nermutations		51

xii	Cont	tents		
	§4.3	Recursive invariance		52
	§4.4	Resemblance		53
	§4.5	Universal partial functions		53
	§4.6	Exercises		55
	niên	Co.		
Chaj	oter 5	RECURSIVE AND RECURSIVELY ENUMERABLE SI	ETS	57
	§5.1	Definitions		57
	§5.2	Basic theorem		60
	§5.3	Recursive and recursively enumerable relations; coding of $k$ -tuples		63
	§5.4	Projection theorems		66
	§5.5	Uniformity AMANTOM OF THE PROPERTY OF THE PROP		-
	§5.6	Finite sets		69
	§5.7	Single-valuedness theorem		71
	§5.8	Exercises		73
	30.0	Extensionality		.0
Chapter 6		REDUCIBILITIES noitexinatos tado larato d		77
	§6.1	General introduction		77
	§6.2	Exercises		79
	80.2	Exercises meldong and and off		19
Cha	pter 7	ONE-ONE REDUCIBILITY; MANY-ONE REDUCIBIL	LITY;	
		CREATIVE SETS		80
	§7.1	One-one reducibility and many-one reducibility		80
	§7.2	Complete sets and to an an analysis of standard sider local	. C.C.	82
	§7.3	Creative sets		84
	§7.4	One-one equivalence and recursive isomorphism		85
29	§7.5	One-one completeness and many-one completeness		87
	§7.6	Cylinders		89
	\$7.7	Productiveness		90
	§7.8	Logic		94
	§7.9	Exercises		99
46	8	PURPOSES; SUMMARY		
Cha	pter 8	TRUTH-TABLE REDUCIBILITIES; SIMPLE SETS		105
	§8.1	Simple sets		105
	§8.2	Immune sets		107
	§8.3	Truth-table reducibility		109
	§8.4	Truth-table reducibility and many-one reducibility		112
	§8.5	Bounded truth-table reducibility on a solid sometaval	84.1	114

§8.6

Structure of degrees

118

iz

		Contents	xiii
2.29	§8.7 §8.8	Other recursively enumerable sets  Exercises	120 121
	80.0	\$13.1 The jump operation	
Cha	pter 9	TURING REDUCIBILITY; HYPERSIMPLE SETS	127
Circa		\$13.4 Ordering of downers.	127
	§9.1	An example	128
	§9.2	Relative recursiveness Relativized theory	134
	§9.3 §9.4	Turing reducibility	137
	§9.4 §9.5	Hypersimple sets; Dekker's theorem	138
	§9.6	Turing reducibility and truth-table reducibility; Post's problem	141
	§9.7	Enumeration reducibility	145
	§9.8	Recursive operators	148
	§9.9	Exercises	154
Chaj	pter 10	POST'S PROBLEM; INCOMPLETE SETS	161
	§10.1	Constructive approaches	161
	§10.2		163
	§10.3	Further results and problems	167
	§10.4	Inseparable sets of any recursively enumerable degree	170
	§10.5	Theories of any recursively enumerable degree	171
	§10.6	Exercises	174
			Chap
Cha	pter 11	THE RECURSION THEOREM	179
	§11.1		179
	§11.2	The recursion theorem	180
	§11.3	Completeness of creative sets; completely productive sets	183
	§11.4	Other applications and constructions	185
	§11.5		192
	§11.6		199
	§11.7		205
	§11.8		211
	§11.9	Exercises	213
Cha	pter 12	he intersection of A and B. willidetugnioo-Conflered De 2012	223
Sty.	§12.1		223
	§12.1 §12.2	100	230
	§12.2 §12.3		231
	§12.3 §12.4	Collegive Scots	234
	§12.5		237
	§12.6		240
	§12.7	아보다 아이들은 경기를 가지면 가지면 하면 하면 보다면 하는데 되었다. 그리고 있다면 하는데	246
	0		

Chapter 13		DEGREES OF UNSOLVABILITY		254
	§13.1	The jump operation		254
	§13.2	Special sets and degrees		262
127	§13.3	Complete degrees; category and measure		265
	§13.4	Ordering of degrees elquees aA		273
	§13.5	Minimal degrees		276
	§13.6	Partial degrees		279
	§13.7	The Medvedev lattice		282
	§13.8	Further results		289
138	§13.9	Exercises, and substantial during the willid substruction		295
141				
Chapter 14		THE ARITHMETICAL HIERARCHY (PART 1)		301
	§14.1	The hierarchy of sets		301
	§14.2	Normal forms		305
101	§14.3	The Tarski-Kuratowski algorithm		307
	§14.4	Arithmetical representation		312
	§14.5	The strong hierarchy theorem adological additional and		314
	§14.6	Degrees Applications to logic ameldorg bas saluser redshift	\$10.2	316
	§14.7	Applications to logic		318
	§14.8	Computing degrees of unsolvability		323
171	§14.9	Exercises		331
Chap	oter 15	THE ARITHMETICAL HIERARCHY (PART 2)		335
179	§15.1	The hierarchy of sets of sets	II tell	335
	§15.2	The hierarchy of sets of functions		346
	§15.3	Functionals mental addition of the		358
	§15.4	Exercises and violation on takes evidents to exempted and		367
	17.8			
Chap	oter 16	THE ANALYTICAL HIERARCHY		373
	§16.1	The analytical hierarchy		373
	§16.2	Analytical representation; applications to logic		384
218	§16.3	Finite-path trees		392
	§16.4	$\Pi_1^1$ -sets and $\Delta_1^1$ -sets		397
	§16.5	Generalized computability		402
	§16.6	Hyperdegrees and the hyperjump; $\Sigma_2^1$ -sets and $\Delta_2^1$ -sets		409
	§16.7	Basis results and implicit definability		418
	§16.8	The hyperarithmetical hierarchy	\$12.2	434
	§16.9	Exercises also eviceded		445
	90.4			
		Bibliography		459
		Index of ivolations		469
		Subject Index		473

## Recursive Functions

81.1	The informal Notion of Algorithm	S Tarabasa
§1.2	An Example: The Primitive Recursive Functions	5
§1.3	Extensionality 9	
§1.4	Diagonalization 10	
§1.5	Formal Characterization 11	
21 6	The Bogic Regult 18	

§1.6 The Basic Result 16 §1.7 Church's Thesis 20

§1.8 Gödel Numbers, Universality, s-m-n Theorem 21

§1.9 The Halting Problem 24

§1.10 Recursiveness 26 Mark and the assumed the same and the assumed the same and t

#### §1.1 THE INFORMAL NOTION OF ALGORITHM

In this chapter we give a formal (i.e., mathematically exact) characterization of recursive function. The concept is basic for the remainder of the book. It is one way of making precise the informal mathematical notion of function computable "by algorithm" or "by effective procedure." In this section, as a preliminary to the formal characterization, we discuss certain aspects of the informal notions of algorithm and function computable by algorithm as they occur in mathematics.

Roughly speaking, an algorithm is a clerical (i.e., deterministic, book-keeping) procedure which can be applied to any of a certain class of symbolic inputs and which will eventually yield, for each such input, a corresponding symbolic output. An example of an algorithm is the usual procedure given in elementary calculus for differentiating polynomials. (The name calculus,

of course, indicates the algorithmic nature of that discipline.)

In what follows, we shall limit ourselves to algorithms which yield, as outputs, integers in some standard notation, e.g., arabic numerals, and which take, as inputs, integers, or k-tuples of integers for a fixed k, in some standard notation. Hence, for us, an algorithm is a procedure for computing a function (with respect to some chosen notation for integers). For our purposes, as we shall see, this limitation (to numerical functions) results in no loss of generality. It is, of course, important to distinguish between the notion of algorithm, i.e., procedure, and the notion of function computable by algorithm, i.e., mapping yielded by procedure. The same

function may have several different algorithms. We shall occasionally refer to functions computable by algorithm as algorithmic functions.  $\dagger$ 

Here are several examples of functions for which well-known algorithms

exist (with respect to the usual denary notation for integers).

- a.  $\lambda x[xth\ prime\ number]$ . (The method of Eratosthenes' sieve is an algorithm here.) (We are assuming Church's lambda notation. To say that  $f = \lambda x[xth\ prime\ number]$  is to say that for all x,  $f(x) = xth\ prime\ number$ .  $\ddagger$ )
- b.  $\lambda xy[the\ greatest\ common\ divisor\ of\ x\ and\ y].$  (The Euclidean algorithm serves here.)
- c.  $\lambda x[the\ integer \leq 9\ whose\ arabic\ numeral\ occurs\ as\ the\ xth\ digit\ in\ the\ decimal\ expansion\ of\ \pi=3.14159$  . . .]. (Any one of a number of common approximation methods will give an algorithm, e.g., quadrature of the unit circle by Simpson's rule.)

Of course there are even simpler and commoner examples of functions computable by algorithm. One such function is

d.  $\lambda xy[x+y]$ . Such common algorithms are the substance of elementary school arithmetic.

Several features of the informal notion of algorithm appear to be essential. We describe them in approximate and intuitive terms.

- \*1. An algorithm is given as a set of instructions of finite size. (Any classical mathematical algorithm, for example, can be described in a finite number of English words.)
- \*2. There is a computing agent, usually human, which can react to the instructions and carry out the computations.
- \*3. There are facilities for making, storing, and retrieving steps in a computation.
- \*4. Let P be a set of instructions as in \*1 and L be a computing agent as in \*2. Then L reacts to P in such a way that, for any given input, the computation is carried out in a discrete stepwise fashion, without use of continuous methods or analogue devices.

\*5. L reacts to P in such a way that a computation is carried forward deterministically, without resort to random methods or devices, e.g., dice.§

Virtually all mathematicians would agree that features \*1 to \*5, although inexactly stated, are inherent in the idea of algorithm. The reader will note an analogy to digital computing machines: \*1 corresponds to the

† Beginning in §1.5, we shall extend our use of the word algorithm to include pro-

cedures for computing nontotal partial functions.

‡ As we proceed, we shall assume, without further comment, the conventions of notation and terminology set forth in the Introduction. In addition to the lambda notation, the restriction of function and partial function to mean mappings on (non-negative) integers is important for Chapter 1.

§ In a more careful discussion, a philosopher of science might contend that \*4 implies \*5. Indeed, he might question whether there is any real difference between \*4 and \*5.