Basic Computability Theory

Jaap van Oosten
Department of Mathematics
Utrecht University

1993, revised 2013

Introduction

The subject of this course is the theory of *computable* or *recursive* functions. *Computability Theory* and *Recursion Theory* are two names for it; the former being more popular nowadays.

0.1 Why Computability Theory?

The two most important contributions Logic has made to Mathematics, are formal definitions of *proof* and of *algorithmic computation*.

How useful is this? Mathematicians are trained in understanding and constructing proofs from their first steps in the subject, and surely they can recognize a valid proof when they see one; likewise, algorithms have been an essential part of Mathematics for as long as it exists, and at the most elementary levels (think of long division, or Euclid's algorithm for the greatest common divisor function). So why do we need a formal definition here?

The main use is twofold. Firstly, a formal definition of proof (or algorithm) leads to an analysis of proofs (programs) in elementary reasoning steps (computation steps), thereby opening up the possibility of numerical classification of proofs (programs)[1]. In *Proof Theory*, proofs are assigned ordinal numbers, and theorems are classified according to the complexity of the proofs they need. In *Complexity Theory*, programs are analyzed according to the number of elementary computation steps they require (as a function of the input), and programming problems are classified in terms of these complexity functions.

Secondly, a formal definition is needed once one wishes to explore the *boundaries* of the topic: what can be proved at all? Which kind of problems can be settled by an algorithm? It is this aspect that we will focus on in this course.

There are, of course, other reasons for studying a formal definition of algorithms and computations: for example, the definition leads to generalizations (for example, where computation is done not on numbers, but on objects of higher type such as functions, or functionals (see, e.g., [33]); or where basic properties of algorithms are taken as axioms for an abstract theory, such as in the study of *Partial Combinatory Algebras* (see chapter 1 of [41])).

After settling on a definition of 'algorithm' (a program in an extremely simple language), we define what we call a 'computable function' (a function whose values can be computed using an algorithm). This leads to an indexing of computable functions, and we study the structure of this indexed collection. We learn techniques for proving that for certain problems there *cannot exist* an algorithm. A further topic is the classification of problems according to 'relative computability': can I solve problem A on the assumption that I can solve B? Suppose an 'oracle' tells me how to solve B, can I then construct an algorithm for A?

In the meantime, we shall also learn about a class of functions (the *primitive recursive functions* which are of importance in other areas of Logic, such as the analysis of systems of Arithmetic (and Gödel's Incompleteness Theorems).

[1] Contrary to a widely held misconception, the aim of Logic is not to tell people how they should reason; the purpose of Logic is to understand how correct reasoning can be analyzed

0.2 A bit of early history

I have taken most of this section from the paper [9], which I highly recommend.

Calculation by a machine was thought of by the universal genius Gottfried Wilhelm Leibniz (1646–1716), who conceived the *Calculus ratiocinator* which is seen by some as a precursor to Turing's universal machine. But the first one who not only *thought* of such calculation but actually *constructed* calculating machines was Charles Babbage (1791–1871), an English mathematician and engineer. Babbage did not publish his ideas; this was done by Luigi Menabrea (1809–1896)[2], whose French text was translated, with notes, by Lady Ada Augusta Lovelace[3] (1815–1852).

It is, however, perhaps ironic that the theoretical work which immediately preceded the construction of the modern computer, focused not on what a *machine* can calculate, but on what a *human being* (a mathematician, following a precise routine) can calculate[4]. Although Babbage came quite close to formulating the notion of a computable function in the later sense of Church-Kleene-Turing-Post, it is argued in [9] that it may have been because of the focus on "hardware" that his work was not very influential[5].

Instead, another stream of thought played an important role. Mathematics became more and more the study of abstract axiomatic systems in the 19th century, and with axiom systems come questions such as: is the axiom system *consistent* (i.e., free of contradiction)? How can one determine the consequences of the axioms? No one was more influential in stressing the importance of such questions than David Hilbert (1862-1943). Hilbert was one of the greatest mathematicians of his time. He was a very modern mathematician; he embraced wholeheartedly Cantor's theory of sets with its higher infinities and transfinite recursions[6]. In 1900, Hilbert addressed the International Congress of Mathematicians in Paris and proposed 23 problems for the new century. Some are about Logic, for example:

1 Settle the Continuum Hypothesis

2 Prove that the axioms of arithmetic are consistent

and some have been resolved using techniques from Logic, such as

10 Find an algorithm to determine whether a polynomial equation with integer coefficients has a solution in the integers

17 Prove that a positive definite rational function is a sum of squares

We see, Hilbert often asked for an "algorithm". Another problem he posed, came to be known (also in English-language texts) as the "Entscheidungsproblem": find an algorithm to decide whether a given sentence of first-order logic is valid.

There were also other, foundational issues which influenced Hilbert. Some mathematicians (like Kronecker and Brouwer) openly doubted the existence of infinite objects in Mathematics and adopted the philosophical stance of *Finitism* (Brouwer called his philosophy *Intuitionism*). Hilbert's plan of attack on these positions was imaginative: he created *Proof Theory*[7]. His idea was as follows: let us analyze proofs, which are finite objects; and in analyzing proofs let us restrict ourselves to methods that are completely unproblematic from the finitist point of view. The kind of methods permitted is laid out in [12][8]. And then, using these unproblematic methods, *let us prove that classical mathematics is consistent.* This became known as *Hilbert's Program.* When

[2] Apart from being a mathematician, Menabrea was also a general who played an important role in the Unification of Italy; after conflicts with Garibaldi, he became Prime Minister

[3] A daughter of the poet Byron. The programming language Ada is named after her. Babbage named her "the Enchantress of Numbers"

[4] In Turing's seminal paper [40], the word "computer" refers to a human

[5] As [9] convincingly argues, Turing was probably not aware of Babbage's work when he wrote [40]

[6] He coined the phrase *Cantor's Paradise*

[7] *Beweistheorie*

[8] Some people have concluded from the discussion there of the "finitary point of view" that Hilbert was a finitist; in my opinion, he was certainly not

the finitist mathematician had to work with functions (which are usually infinite), these would have to have a finite presentation, that is: an *algorithm*. So, Hilbert became interested in the study of functions which can be produced by algorithms, and togethr with Ackermann ([11]) he developed the class of functions that Gödel later would call "recursive", and Kleene "primitive recursive", the terminology still in use today.

There is an enormous literature on Hilbert's discussion with the Finitists/Intuitionists, a discussion which came to be referred to as *Grundlagenstreit* (Battle of Foundations). For a bit of introduction, read Chapter III, section 1, of [37]. Also see [36, 7].

Gödel proved that if one took the "finitary point of view" to mean "first-order arithmetic", then Hilbert's program was bound to fail since first-order arithmetic cannot prove its own consistency, let alone that of Mathematics. But still, Hilbert's program and Hilbert's continued interest in foundational matters was extremely fruitful.

In the meantime, in the 1920's, the American mathematician Alonzo Church (1903–1995) had developed the *Lambda Calculus* (see [2] for an introduction) as a formalism for doing Mathematics.

In 1936, four papers appeared, each proposing a definition of "algorithmically computable": [5, 40, 19, 29]. Church proposed for "algorithmically computable": representable by a λ-term. Kleene proposed a definition in the form of schemes for constructing functions. Post's paper is very short and poor on details. The paper by Turing stands out in readability and novelty of approach; highly recommended reading! Read [14] for an account of Alan Turing's dramatic life.

It was soon realized that the four approaches all defined the *same* class of functions. Church and Turing moreover noted that with the new, precise definition of algorithmically computable and Gödel's results, one could settle the Entscheidungsproblem in the negative: there cannot exist such an algorithm.

0.3 Some later developments

Of the four pioneers in 1936 (Church, Kleene, Turing and Post), only Kleene and Post developed the theory of "recursive functions" (as they were now called) further. The most important was Stephen Cole Kleene (1909–1994), who shaped much of it single-handedly. In particular, he discovered the *Recursion Theorem* (see theorem 2.4.3 below).

Here, I just mention a number of topics that were investigated.

Definability Recursion theory studies *definable sets*, subsets of $\mathbb{N}^k$ for some k which have a logical description. By a technique called *many-one reducibility* such sets can be *classified* in an *Arithmetical Hierarchy*. We shall see that in these notes. The classification of arithmetically definable sets is carried further in the study of the *Analytical Hierarchy*, and also in the field of *Descriptive Set Theory*.

Subrecursion There is a hierarchy, indexed by the countable ordinals, of classes of computable functions. In this hierarchy, the primitive recursive functions arise at level ω and the functions which are provably total in Peano Arithmetic (see section 2.1.2 for an explanation of this notion) arise at level ε_0. For an exposition of this area, see [32].

Degrees of Unsolvabiity We can ask ourselves which problems can be effectively solved provided a given problem is solved for us, in whatever way (Turing used the word "oracle"). This leads to the important notion of "Turing reducibility", which we shall also meet in this course. This relation divides the subsets of $\mathbb{N}$ into equivalence classes, called "Turing degrees". The degrees form an upper semilattice, which is vast and very complicated. Already for degrees of 'recursively enumerable sets' (the lowest class of non-computable sets) there is a lot of theory; see the excellent books [38, 13].

Recursion in Higher Types When is a function on functions, that is, a *functional of type 1*, to be called computable? And a function on functionals of type 1, and so forth? Kleene started thinking about this in the 1950's and produced a number of deep papers ([20, 21] among others) about it. Apart from the already mentioned book by Sanchis, an

introduction to this area can be found in [17]. The study of recursive functionals of higher type was revived when theoretical computer scientists studied the programming language PCF and "fully abstract" models for it.

Realizability Kleene also observed, that the theory of computable functions can be used to give a model for a form of reasoning (advocated by Brouwer) in which the 'principle of excluded third', $A \vee \neg A$, need not be valid. The basic paper is [18]. Realizability, originally a proof-theoretic tool, turned into an independent field of study with M. Hyland's paper [15], which connects the world of partial recursive functions with *Topos Theory*. See also [41].

Recursive Mathematics This is the study of familiar mathematical structures (groups, rings, lattices, CW-complexes, real analysis) from the point of computability. For analysis, see [30]

Hilbert 10 and Number Theory In 1970, Yuri Matiyasevich used notions of Computability Theory to show that Hilbert's 10-th Problem cannot be solved: there is no algorithm which decides, for a given polynomial equation with integer coefficients, whether it has a solution in the integers. This result was extremely important and has applications throughout Logic, for example in the theory of Models of Peano Arithmetic ([16]). But even more interestingly, number theorists have been thinking about replacing $\mathbb{Z}$ by other number rings and number fields (for example, $\mathbb{Q}$). Some introduction can be found in the very readable paper [28].

0.4 What is in these notes?

These notes consist of 6 chapters.

The first chapter treats our preferred model of computation, *Register Machines*. We define the notion of an *(RM)-computable function* and prove some closure properties of these functions.

Chapter 2 starts with the theory of primitive recursive functions. We develop coding of sequences, then coding of programs for the Register Machine; the Kleene T-predicate; Smn-Theorem and Recursion Theorem.

Chapter 3 introduces unsolvable problems and then moves on to recursively enumerable sets. We give the "Kleene tree". We treat extensional sets and prove the Myhill-Shepherdson, Rice-Shapiro and Kreisel-Lacombe-Shoenfield Theorems. Examples of r.e. sets outside Recursion Theory are given.

Chapter 4 deals with *many-one reducibility* and the Arithmetical Hierarchy. We practice some exact classifications. A final section treats the equivalence: m-complete $\Leftrightarrow$ creative, via Myhill's Isomorphism Theorem. Post's construction of a simple set is given.

Chapter 5 gives the concept of computability relative to an oracle. The jump is defined and its basic properties are derived. We prove a theorem by Friedberg saying that the jump operation is surjective on the degrees above the junp of the empty set.

Chapter 6 provides a glimpse at the Analytical Hierarchy. Starting with *recursive functionals*, we define the notion Σ^1_n-set. We show that the collection of indices of recursive well-founded trees is m-complete in Π^1_1. We prove that the set of (codes of) true arithmetical sentences is Δ^1_1. Without proof, we mention the "Kleene tree" analogue for Δ^1_1 and the Suslin-Kleene Theorem.

0.5 Literature

Authoritative monographs on the general theory of computable functions are [31] and [26, 27]. An outline is [35]. Good introductory text books are [6] and [3].

0.6 Acknowledgements

I learned Recursion Theory from Dick de Jongh, who was a student of Kleene. De Jongh had his own notes, but also used a set of lecture notes prepared by Robin Grayson. My lecture notes grew out of these notes by De Jongh and Grayson, but gradually I reworked the material and added to it.

Contents

Chapter 1

Register Machines and Computable Functions

In his pioneering paper [40], Alan Turing set out to isolate the essential features of a calculation, done by a human (the "computer"), who may use pen and paper, and who is working following a *definite method*. Turing argues that at any given moment of the calculation, the mind of the "computer" can be in only one of a finite collection of *states*, and that in each state, given the intermediate results thus far obtained, the next calculation step is completely determined.

A mathematical formulation of this is the *Turing machine*. Turing machines still have a prominent place in text books on Computability Theory and Complexity Theory, but for this course we prefer a concept which is closed to present-day programming on an actual computer: the *Register Machine*, developed in the 1960s. There are many names associated with this development (Cook, Elgot, Hartmanis, Lambek, Melzak, Minsky, Reckhow, Robinson) but the most cited is that of *Marvin Lee Minsky* (born 1927).

The notions of Turing Machine and Register Machine give rise to the same notion of 'computable function'.

1.1 Register Machines

We picture ourselves a machine, the *Register Machine*, which has an infinite, numbered collection of memory storage places or *registers* $R_1, R_2, \ldots$, each of which can store a natural number of arbitrary size. The number stored in register R_i is denoted r_i.

Furthermore the Register Machine modifies the contents of its registers in response to commands in a *program*. Such a program is a finite list of commands which all have one of the following two forms:

> the command $r_i^+ \Rightarrow n$ (which is read as: "add 1 to r_i, and move to the n-th command")
>
> the command $r_i^- \Rightarrow n, m$ (which is read as: "if $r_i > 0$, subtract 1 from r_i and move to the n-th command; otherwise, move to the m-th command")

It may happen that the 'move to the n-th command' is impossible to execute because there is no n-th command; in that case, the machine stops.

Let us, before formulating a more mathematical definition, see a few simple programs and their intended effects. In these examples, we put numbers before the commands for clarity; these are not part of the programming language.

Example 1.1.1 a) $\quad 1 \quad r_i^+ \Rightarrow 2$

The machine adds 1 to r_i, and stops.

b) $1 \quad r_i^+ \Rightarrow 1$

The machine keeps on adding 1 to r_i.

c) $1 \quad r_i^- \Rightarrow 1, 2$

The machine empties register R_i, and stops.

d)
$1 \quad r_i^- \Rightarrow 2, 3$
$2 \quad r_j^+ \Rightarrow 1$

The machine carries the content of R_i over to R_j.

e)
$1 \quad r_i^- \Rightarrow 2, 4$
$2 \quad r_j^+ \Rightarrow 3$
$3 \quad r_k^+ \Rightarrow 1$

The machine carries the content of R_i simultaneously over to R_j and R_k.

Exercise 1 An alternative notion of machine, the *Unlimited Register Machine* of [6] has 4 types of commands:

1) The command $Z(i)$ is read as: "empty R_i, and move to the next command"

2) The command $S(i)$ is read as: "add 1 to r_i, and move to the next command"

3) The command $T(m, n)$ is read as: "replace r_n by r_m and move to the next command"

4) The command $J(m, n, q)$ is read as: "if $r_m = r_n$, move to the q-th command; otherwise, move to the next command"

For each of these commands, write a short RM-program which has the same effect.

Clearly, the Register Machine is not really a mathematical object. A program, on the other hand, can be seen as a finite list of which each item is either an ordered pair or an ordered triple of positive integers.

We now define the notion of a *computation of the RM with input $a_1, \ldots, a_k$ and program P*.

Definition 1.1.2 Let P be a program for the RM, and $a_1, \ldots, a_k$ a list of natural numbers. A *computation of the Register Machine with program P and input $a_1, \ldots, a_k$* is a finite or infinite list of $l + 1$-tuples

$$(n_i, r_1^i, \ldots, r_l^i)_{i \geq 1}$$

with the following properties:

a) $l \geq k$, and $(n_1, r_1^1, \ldots, r_l^1) = (1, a_1, \ldots, a_k, 0, \ldots, 0)$

b) If $n_i = m$ then exactly one of the following three situations occurs:

b1 The program P does not have an m-th command, and the i-th tuple is the last item of the list (so the computation is finite);

b2 The m-th command of P is $r_j^+ \Rightarrow u$, $j \leq l$, $n_{i+1} = u$ and

$$(r_1^{i+1}, \ldots, r_l^{i+1}) = (r_1^i, \ldots, r_{j-1}^i, r_j^i + 1, r_{j+1}^i, \ldots, r_l^i)$$

b3 The m-th command of P is $r_j^- \Rightarrow u, v$, $j \leq l$, and now *either* $r_j^i > 0$ and

$$(n_{i+1}, r_1^{i+1}, \ldots, r_l^{i+1}) = (u, r_1^i, \ldots, r_j^i - 1, \ldots, r_l^i)$$

or $r_j^i = 0$ and

$$(n_{i+1}, r_1^{i+1}, \ldots, r_l^{i+1}) = (v, r_1^i, \ldots, r_l^i)$$

If a computation is finite, with last element $(n_K, r_1^K, \ldots, r_l^K)$, then the number r_1^K is the *output* or *result* of the computation.

Remark 1.1.3 Clearly, if $(n_i, r_1^i, \ldots, r_l^i)_{i\geq 1}$ is a computation with P and input $a_1, \ldots, a_k$, and $l' \geq l$, then the list of $l'+1$-tuples

$$(n_i, r_1^i, \ldots, r_l^i, 0, \ldots, 0)_{i\geq 1}$$

is also a computation with P and input $a_1, \ldots, a_k$. But once l is fixed, computations are unique: the RM is *deterministic*.

A program can be the empty list. Clearly, the list consisting of the single $l+1$-tuple

$$(1, a_1, \ldots, a_k, 0, \ldots, 0)$$

is the unique computation with the empty program and input $a_1, \ldots, a_k$.

Definition 1.1.4 (Composition of programs) If P and Q are programs, there is a composition PQ, defined as follows. Suppose P has k commands. First modify P to P' by replacing every command number $n > k$ by $k+1$ (so $r_i^+ \Rightarrow n$ becomes $r_i^+ \Rightarrow k+1$, etc.). Then modify Q to obtain Q'' by replacing each command number m in Q by $k+m$ (so $r_i^+ \Rightarrow m$ becomes $r_i^+ \Rightarrow k+m$, etc.)

The program PQ is now the list of commands P' followed by the list of commands Q''.

Exercise 2 Show that the operation of composition on programs is associative. Show also that if Q is the empty program, $QP = P$ for any program P. What about PQ?

Notation for computations. We write

$$\begin{array}{l} a_1\ a_2 \cdots a_k \\ \quad \Downarrow\ P \\ b_1\ b_2 \cdots b_m \end{array} \quad \text{or} \quad \begin{array}{l} \vec{a} \\ \Downarrow P \\ \vec{b} \end{array}$$

when there is a finite computation $(n_i, r_1^i, \ldots, r_l^i)_{i=1}^K$ with program P and input $a_1, \ldots, a_k$, such that $(r_1^K, \ldots, r_l^K) = (b_1, \ldots, b_m, 0, \ldots, 0)$.

Clearly,

$$\text{if } \begin{array}{l} \vec{a} \\ \Downarrow P \\ \vec{b} \end{array} \text{ and } \begin{array}{l} \vec{b} \\ \Downarrow Q \\ \vec{c} \end{array} \text{ then } \begin{array}{l} \vec{a} \\ \Downarrow PQ \\ \vec{c} \end{array}$$

1.2 Partial computable functions

A function $f : A \to \mathbb{N}$ with $A \subseteq \mathbb{N}^k$ is called a *partial function of k arguments*, or a *k-ary partial function*. For $k = 1, 2, 3$ we say *unary, binary, ternary*, respectively. The set A is the *domain* of f, written $\mathrm{dom}(f)$. If $\mathrm{dom}(f) = \mathbb{N}^k$ then f is called *total*; for us, a total function is a special case of a partial function.

Definition 1.2.1 Let f be a k-ary partial function. Then f is called *RM-computable* or *computable* for short, if there is a program P such that for every k-tuple $a_1, \ldots, a_k$ of natural numbers we have:

i) if $\vec{a} \in \mathrm{dom}(f)$ then there is a finite computation $(n_i, r_1^i, \ldots, r_l^i)_{i=1}^K$ with P and input $\vec{a}$, such that $r_1^K = f(\vec{a})$;

ii) if $\vec{a} \notin \mathrm{dom}(f)$ then there is no finite computation with P and input $\vec{a}$.

We say that the program P *computes* f.

Note, that it does *not* follow from definition 1.2.1 that if f is computable, then the restriction of f to a smaller domain is also computable!

Exercise 3 Suppose f is a computable function of k variables. Show that there is a program P which computes f in such a way that for every $\vec{a} \in \mathrm{dom}(f)$,

$$\begin{array}{c} \vec{a} \\ \Downarrow P \\ f(\vec{a})\,\vec{a} \end{array}$$

We derive some closure properties of the class of computable functions.

Definition 1.2.2 Suppose $g_1, \ldots, g_l$ are partial k-ary functions and h is partial l-ary. The partial k-ary function f with domain

$$\{\vec{x} \in \mathbb{N}^k \mid \vec{x} \in \bigcap_{i=1}^{l} \mathrm{dom}(g_i) \text{ and } (g_1(\vec{x}), \ldots, g_l(\vec{x})) \in \mathrm{dom}(h)\}$$

and defined by $f(\vec{x}) = h(g_1(\vec{x}, \ldots, g_l(\vec{x}))$ is said to be defined from $g_1, \ldots, g_l, h$ by *composition*.

Definition 1.2.3 Suppose g is a partial k-ary function and h is partial $k+2$-ary. Let f be the *least* partial $k+1$-ary function with the properties:

i) if $(x_1, \ldots, x_k) \in \mathrm{dom}(g)$ then $(0, x_1, \ldots, x_k) \in \mathrm{dom}(f)$, and

$$f(0, x_1, \ldots, x_k) = g(x_1, \ldots, x_k)$$

ii) if $(y, x_1, \ldots, x_k) \in \mathrm{dom}(f)$ and $(y, f(y, x_1, \ldots, x_k), x_1, \ldots, x_k) \in \mathrm{dom}(h)$ then $(y{+}1, x_1, \ldots, x_k) \in \mathrm{dom}(f)$ and

$$f(y+1, x_1, \ldots, x_k) = h(y, f(y, x_1, \ldots, x_k), x_1, \ldots, x_k)$$

(Here, partial functions are seen as relations; so f is the intersection of all partial functions satisfying i) and ii))
Then f is said to be defined from g and h by *primitive recursion*.

Exercise 4 Suppose f is defined by primitive recursion from g and h as above. Show that if $(y, \vec{x}) \in \mathrm{dom}(f)$ then for all $w < y$, $(w, \vec{x}) \in \mathrm{dom}(f)$.

Remark 1.2.4 In definition 1.2.3 we *do not* exclude the case $k = 0$; a 'partial 0-ary function' is either a number, or undefined. Clearly, if g is the undefined partial 0-ary function and f is defined by primitive recursion from g and h, $\mathrm{dom}(f) = \emptyset$ (f is the *empty function*). In the case that $g = n$ we have: $f(0) = n$, and if $y \in \mathrm{dom}(f)$ and $(y, f(y)) \in \mathrm{dom}(h)$, then $y + 1 \in \mathrm{dom}(f)$ and $f(y+1) = h(y, f(y))$.

Definition 1.2.5 Suppose g is partial $k+1$-ary. Let f be the partial k-ary function such that $(x_1, \ldots, x_k) \in \mathrm{dom}(f)$ precisely if there is a number y such that the $k+1$-tuples $(0, \vec{x}), \ldots, (y, \vec{x})$ all belong to $\mathrm{dom}(g)$ and $g(y, \vec{x}) = 0$; and $f(\vec{x})$ is the least such y, in that case. Then f is said to be defined from g by *minimalization*.

Theorem 1.2.6 *The class of partial computable functions is closed under definition by composition, primitive recursion and minimalization.*

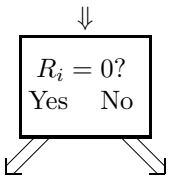
Proof. Recall the result of exercise 3: if f is partial computable then there is a program P which computes f and is such that for all $\vec{a} \in \mathrm{dom}(f)$,

$$\begin{array}{c} \vec{a} \\ \Downarrow P \\ f(\vec{a})\vec{a} \end{array}$$

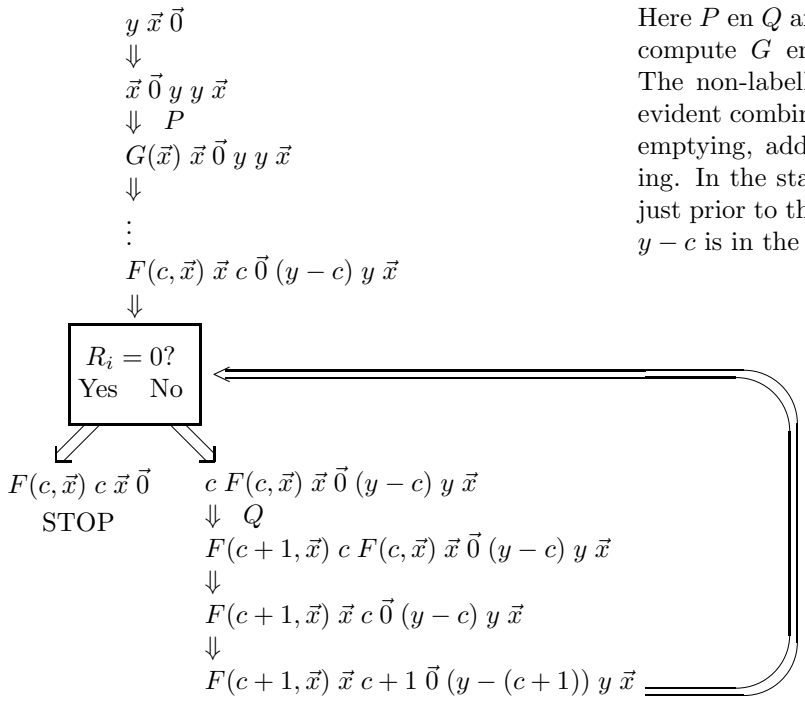
Furthermore, we may have, in a program, a pair of commands like this:

$$\begin{array}{ll} k & r_i^- \Rightarrow k+1, l \\ k+1 & r_i^+ \Rightarrow m \end{array}$$

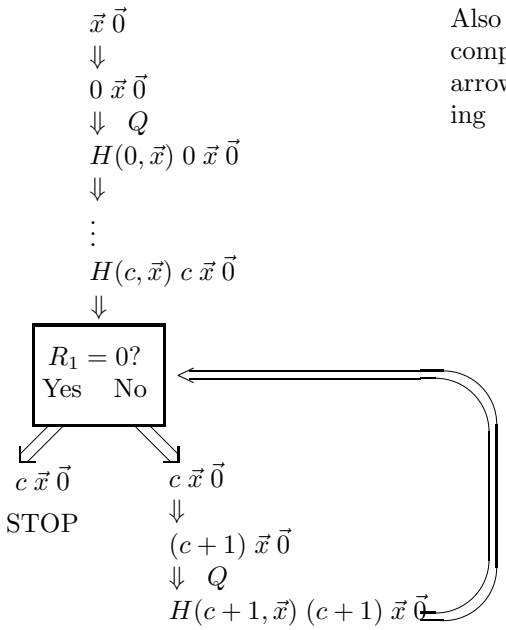
(with $l, m \notin \{k, k+1\}$). This pair has the effect of *definition by cases*: if $r_i = 0$, move to command number l; otherwise, move to command number m. In diagrams of computations we picture this as:



The asserted closure properties now follow by an inspection of the diagrams for computations below; you should convince yourself that you can write a program which performs the diagrammed computations.



Here P en Q are programs which compute G en H, respectively. The non-labelled arrows denote evident combinations of copying, emptying, adding and subtracting. In the state of the machine just prior to the case distinction, $y - c$ is in the i-th register



Also here Q is a program which computes H and the non-labelled arrows have their evident meaning

■

Chapter 2

Recursive Functions

2.1 Primitive recursive functions and relations

Notation for functions. In mathematical texts, it is common to use expressions containing variables, such as $x + y$, x^2, $x \log y$ etc., both for a (variable) *number* and for the *function* of the occurring variables: we say "the function $x + y$". However, in Computability Theory, which is to a large extent about ways of defining functions, it is better to distinguish these different meanings by different notations. The expression $x \log y$ may mean, for example:

- a real number
- a function of (x, y), that is a function: $\mathbb{R}^2 \to \mathbb{R}$
- a function of (y, x), i.e. another function: $\mathbb{R}^2 \to \mathbb{R}$
- a function of y (with parameter x, so actually a parametrized family of functions: $\mathbb{R} \to \mathbb{R}$)
- a function of (x, y, z), that is a function: $\mathbb{R}^3 \to \mathbb{R}$

In order to distinguish these meanings we employ the so-called λ-notation: if $\vec{x}$ is a sequence of variables $x_1 \cdots x_k$ which might occur in the expression G, then $\lambda\vec{x}.G$ denotes the function which assigns to the k-tuple $n_1 \cdots n_k$ the value $G(n_1, \ldots, n_k)$ (substitute the n_i for x_i in G). In this notation the 5 meanings above can be distuinguished by notation as follows: $x \log(y)$, $\lambda xy.x \log(y)$, $\lambda yx.x \log(y)$, $\lambda y.x \log(y)$ and $\lambda xyz.x \log(y)$.

Definition 2.1.1 The class of *primitive recursive* functions $\mathbb{N}^k \to \mathbb{N}$ (where k is allowed to vary over $\mathbb{N}$) is generated by the following clauses:

i) the number 0 is a 0-ary primitive recursive function;

ii) the *zero function* $Z = \lambda x.0$ is primitive recursive;

iii) the *successor function* $S = \lambda x.x + 1$ is primitive recursive;

iv) the *projections* $\Pi^k_i = \lambda x_1 \cdots x_k.x_i$ (for $1 \leq i \leq k$) are primitive recursive;

v) the primitive recursive functions are closed under definition by composition and primitive recursion.

Recall remark 1.2.4: if $H : \mathbb{N}^2 \to \mathbb{N}$ is primitive recursive and $n \in \mathbb{N}$, then the function F, defined by

$$\begin{aligned} F(0) &= n \\ F(y+1) &= H(y, F(y)) \end{aligned}$$

is also primitive recursive. The constant n is a primitive recursive 0-ary function, being the composition of 0 and n times the function S.

Exercise 5 Prove that the primitive recursive functions are total. Prove also that the primitive recursive functions are computable.

When we speak of a *k-ary relation*, we mean a subset of $\mathbb{N}^k$. We shall stick to the following convention for the *characteristic function* $\chi_A : \mathbb{N}^k \to \mathbb{N}$ of the k-ary relation A:

$$\chi_A(\vec{x}) = \begin{cases} 0 & \text{if } \vec{x} \in A \\ 1 & \text{else} \end{cases}$$

A relation is said to be primitive recursive if its characteristic function is.

Examples of primitive recursive functions. The following derivations show for a couple of simple functions that they are primitive recursive:

a) $\lambda xy.x + y$. For, $0 + y = y = \Pi^1_1(y)$, and
$(x+1)+y = S(x+y) = S(\Pi^3_2(x, x+y, y))$, hence $\lambda xy.x+y$ is defined by primitive recursion from Π^1_1 and a function defined by composition form S and Π^3_2;

b) $\lambda xy.xy$. For, $0y = 0 = Z(y)$, and
$(x+1)y = xy+y = (\lambda xy.x+y)(\Pi^3_2(x, xy, y), \Pi^3_3(x, xy, y))$, hence $\lambda xy.xy$ is defined by primitive recursion from Z and a function defined by composition from $\lambda xy.x+y$ and projections;

c) $\lambda x.\mathrm{pd}(x)$ (the *predecessor function*: $\mathrm{pd}(x) = x - 1$ if $x > 0$, and $\mathrm{pd}(0) = 0$). For, $\mathrm{pd}(0) = 0$, and
$\mathrm{pd}(x + 1) = x = \Pi^2_1(x, \mathrm{pd}(x))$

Exercise 6 Prove that the following functions are primitive recursive:

i) $\lambda xy.x^y$

ii) $\lambda xy.x \dot{-} y$. This is *cut-off subtraction*: $x \dot{-} y = x - y$ if $x \geq y$, and $x \dot{-} y = 0$ if $x < y$.

iii) $\lambda xy.\min(x, y)$

iv) sg (the *sign function*), where

$$\mathrm{sg}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{else} \end{cases}$$

v) $\overline{\mathrm{sg}}$, where

$$\overline{\mathrm{sg}}(x) = \begin{cases} 0 & \text{if } x > 0 \\ 1 & \text{else} \end{cases}$$

vi) $\lambda xy.|x - y|$

vii) $\lambda x.n$ for fixed n

viii) $\lambda x.x!$

ix) $\lambda xy.\mathrm{rm}(x, y)$ where $\mathrm{rm}(x, y) = 0$ if $y = 0$, and the remainder of x on division by y otherwise.

Exercise 7 Prove that if A is a primitive recursive relation, so is the complement of A.

Exercise 8 Prove that the following relations are primitive recursive:

i) $\{(x, y) \mid x = y\}$

ii) $\{(x, y) \mid x \leq y\}$

iii) $\{(x, y) \mid x|y\}$

iv) $\{x \mid x \text{ is a prime number}\}$

Exercise 9 Show that the function C is primitive recursive, where C is given by

$$C(x,y,z) = \begin{cases} x & \text{if } z = 0 \\ y & \text{else} \end{cases}$$

Therefore, we can define primitive recursive functions by 'cases', using primitive recursive relations.

Proposition 2.1.2

i) If the function $F : \mathbb{N}^{k+1} \to \mathbb{N}$ *is primitive recursive, then so are the functions:*

$$\lambda\vec{x}z.\textstyle\sum_{y<z} F(\vec{x},y)$$
$$\lambda\vec{x}z.\textstyle\prod_{y<z} F(\vec{x},y)$$
$$\lambda\vec{x}z.(\mu y < z.F(\vec{x},y) = 0)$$

The last of these is said to be defined from F *by* bounded minimalization*, and produces the least* $y < z$ *for which* $F(\vec{x},y) = 0$*; if such an* $y < z$ *does not exist, it outputs* z*);*

ii) If A *and* B *are primitive recursive* k*-ary relations, then so are* $A \cap B$*,* $A \cup B$ *and* $A - B$*;*

iii) If A *is a primitive recursive* $k+1$*-ary relation, then the relations* $\{(\vec{x},z) \mid \exists y < z(\vec{x},y) \in A\}$ *and* $\{(\vec{x},z) \mid \forall y < z(\vec{x},y) \in A\}$ *are also primitive recursive.*

Proof.

a) $\sum_{y<0} F(\vec{x},y) = 0$ and $\sum_{y<z+1} F(\vec{x},y) = \sum_{y<z} F(\vec{x},y) + F(\vec{x},z)$;
$\prod_{y<0} F(\vec{x},y) = 1$ and $\prod_{y<z+1} F(\vec{x},y) = (\prod_{y<z} F(\vec{x},y))F(\vec{x},z)$;
$(\mu y < 0.F(\vec{x},y) = 0) = 0$ and $(\mu y < z+1.F(\vec{x},y) = 0) = (\mu y < z.F(\vec{x},y) = 0) + \mathrm{sg}(\prod_{y<z+1} F(\vec{x},y))$

b) $\chi_{A\cap B} = \lambda x.\mathrm{sg}(\chi_A(x) + \chi_B(x))$
$\chi_{A\cup B} = \lambda x.\chi_A(x)\chi_B(x)$

Exercise 10 Finish the proof of Proposition 2.1.2.

∎

Exercise 11 If $F : \mathbb{N}^2 \to \mathbb{N}$ is primitive recursive, then so is $\lambda n.\sum_{k<n} F(n,k)$.

Proposition 2.1.3 *If* G_1*,* G_2 *and* H *are primitive recursive functions* $\mathbb{N}^n \to \mathbb{N}$*, then so is the function* F*, defined by*

$$F(\vec{x}) = \begin{cases} G_1(\vec{x}) & \textit{if } H(\vec{x}) = 0 \\ G_2(\vec{x}) & \textit{else} \end{cases}$$

Proof. For, $F(\vec{x}) = C(G_1(\vec{x}), G_2(\vec{x}), H(\vec{x}))$, where C is the function from exercise 9. ∎

Exercise 12 Let $p_0, p_1, \ldots$ be the sequence of prime numbers: $2, 3, 5, \ldots$ Show that the function $\lambda n.p_n$ is primitive recursive.

2.1.1 Coding of pairs and tuples

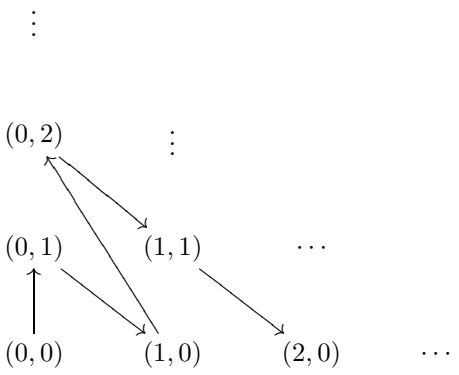
One of the basic techniques in Computability Theory is the *coding of programs and computations as numbers*.

We shall code sequences of numbers as one number, in such a way that important operations on sequences, such as: taking the length of a sequence, the i-th element of the sequence, forming a sequence out of two sequences by putting one after the other (*concatenating* two sequences), are primitive recursive *in their codes*. This is carried out below.

Any bijection $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is called a *pairing function*: if $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is bijective we say that $f(x, y)$ *codes the pair* (x, y). An example of such an f is the primitive recursive function $\lambda xy.2^x(2y + 1) - 1$.

Exercise 13 Let $f(x, y) = 2^x(2y + 1) - 1$. Prove that the functions $k_1 : \mathbb{N} \to \mathbb{N}$ and $k_2 : \mathbb{N} \to \mathbb{N}$ which satisfy $f(k_1(x), k_2(x)) = x$ for all x, are primitive recursive.

A simpler pairing function is given by the "diagonal enumeration" j of $\mathbb{N} \times \mathbb{N}$:



So, $j(0, 0) = 0,\ j(0, 1) = 1,\ j(1, 0) = 2,\ j(0, 2) = 3$ etc. We have:

$$j(n, m) = \#\{(k, l) \in \mathbb{N} \times \mathbb{N} \mid k + l < n + m \ \vee (k + l = n + m \wedge k < n)\}$$

in other words

$$j(n, m) = \frac{1}{2}(n + m)(n + m + 1) + n = \frac{(n + m)^2 + 3n + m}{2}$$

The function j is given by a polynomial of degree 2. By the way, there is a theorem (the Fueter-Pólya Theorem, see [37]) which says that j and its 'twist' i.e. the function $\lambda nm.j(m, n)$ are the *only* polynomials of degree 2 that induce a bijection: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

It is convenient that $x \leq j(x, y)$ and $y \leq j(x, y)$, so if we define:

$$j_1(z) = \mu x \leq z.[\exists y \leq z.j(x, y) = z]$$
$$j_2(z) = \mu y \leq z.[\exists x \leq z.j(x, y) = z]$$

then $j(j_1(z), j_2(z)) = z$.

Exercise 14 Prove this and prove also that j_1 and j_2 are primitive recursive.

Exercise 15 (Simultaneous recursion) Suppose the functions $G_1, G_2 : \mathbb{N}^k \to \mathbb{N}$ and $H_1, H_2 : \mathbb{N}^{k+3} \to \mathbb{N}$ are primitive recursive. Define the functions F_1 and $F_2 : \mathbb{N}^{k+1} \to \mathbb{N}$ 'simultaneously' by the following scheme:

$$\begin{array}{lcllcl} F_1(0, \vec{x}) & = & G_1(\vec{x}) & F_1(y + 1, \vec{x}) & = & H_1(y, F_1(y, \vec{x}), F_2(y, \vec{x}), \vec{x}) \\ F_2(0, \vec{x}) & = & G_2(\vec{x}) & F_2(y + 1, \vec{x}) & = & H_2(y, F_1(y, \vec{x}), F_2(y, \vec{x}), \vec{x}) \end{array}$$

Check that F_1 en F_2 are well-defined, and use the pairing function j and its projections j_1 and j_2 to show that F_1 and F_2 are primitive recursive.

We are also interested in good bijections: $\mathbb{N}^n \to \mathbb{N}$ for $n > 2$. In general, such bijections can be given by polynomials of degree n, but we shall use polynomials of higher degree:

Definition 2.1.4 The bijections $j^m : \mathbb{N}^m \to \mathbb{N}$ for $m \geq 1$ are defined by:

$$\begin{array}{rl} j^1 & \text{is the identity function} \\ j^{m+1}(x_1, \ldots, x_m, x_{m+1}) & = j(j^m(x_1, \ldots, x_m), x_{m+1}) \end{array}$$

Then we also have *projection functions* $j_i^m : \mathbb{N} \to \mathbb{N}$ for $1 \leq i \leq m$, satisfying

$$j^m(j_1^m(z), \ldots, j_m^m(z)) = z$$

for all $z \in \mathbb{N}$, and given by:

$$\begin{array}{rcl} j_1^1(z) & = & z \\ j_i^{m+1}(z) & = & \left\{ \begin{array}{rl} j_i^m(j_1(z)) & \text{if } 1 \leq i \leq m \\ j_2(z) & \text{if } i = m+1 \end{array} \right. \end{array}$$

Exercise 16 Prove:

i) $j_i^m(j^m(x_1, \ldots, x_m)) = x_i$ for $1 \leq i \leq m$; and

ii) the functions j^m and j_i^m are primitive recursive.

Exercise 16 states that for every m and i, the function j_i^m is primitive recursive. However, the functions j_i^m are connected in such a way, that one is led to suppose that there is also one big primitive recursive function which takes m and i as variables. This is articulated more precisely in the following proposition.

Proposition 2.1.5 *The function F, defined by*

$$F(x, y, z) = \left\{ \begin{array}{rl} 0 & \textit{if } y = 0 \textit{ or } y > x \\ j_y^x(z) & \textit{else} \end{array} \right.$$

is primitive recursive.

Proof. We first note that the function $G(w, z) = (j_1)^w(z)$ (the function j_1 iterated w times) is primitive recursive. Indeed: $G(0, z) = z$ and $G(w + 1, z) = j_1(G(w, z))$. Now we have:

$$F(x, y, z) = \left\{ \begin{array}{rl} 0 & \text{if } y = 0 \text{ or } y > x \\ G(x - 1, z) & \text{if } y = 1 \\ j_2(G(x - y, z)) & \text{if } y > 1 \end{array} \right.$$

Hence F is defined from the primitive recursive function G by means of repeated distinction by cases. ∎

Exercise 17 Fill in the details of this proof. That is, show that the given definition of F is correct, and that from this definition it follows that F is a primitive recursive function

The functions j^m and their projections j_i^m give primitive recursive bijections: $\mathbb{N}^m \to \mathbb{N}$. Using proposition 2.1.5 we can now define a bijection: $\bigcup_{m \geq 0} \mathbb{N}^m \to \mathbb{N}$ with good properties. An element of $\mathbb{N}^m$ for $m \geq 1$ is an ordered m-tuple or *sequence* $(x_1, \ldots, x_m)$ of elements of $\mathbb{N}$; the unique element of $\mathbb{N}^0$ is the *empty sequence* $(-)$. The result of the function $\bigcup_{m \geq 0} \mathbb{N}^m \to \mathbb{N}$ to be defined, on input $(x_1, \ldots, x_m)$ or $(-)$ will be written as $\langle x_1, \ldots, x_m \rangle$ or $\langle\,\rangle$ and will be called the *code of the sequence*.

Definition 2.1.6

$$\begin{array}{rcl} \langle\,\rangle & = & 0 \\ \langle x_0, \ldots, x_{m-1} \rangle & = & j(m - 1, j^m(x_0, \ldots, x_{m-1})) + 1 \text{ if } m > 0 \end{array}$$

Exercise 18 Prove that for every $y \in \mathbb{N}$ the following holds: either $y = 0$ or there is a unique $m > 0$ and a unique sequence $(x_0, \ldots, x_{m-1})$ such that $y = \langle x_0, \ldots, x_{m-1} \rangle$.

Remark. In coding arbitrary sequences we have started the convention of letting the indices run from 0; this is more convenient and also consistent with the convention that the natural numbers start at 0.

We now need a few primitive recursive functions for the effective manipulation of sequences.

Definition 2.1.7 The function $\mathrm{lh}(x)$ gives us the *length* of the sequence with code x, and is given as follows:

$$\mathrm{lh}(x) = \begin{cases} 0 & \text{if } x = 0 \\ j_1(x-1)+1 & \text{if } x > 0 \end{cases}$$

The functions $(x)_i$ give us the i-th element of the sequence with code x (count from 0) if $0 \leq i < \mathrm{lh}(x)$, and 0 otherwise, and is given by

$$(x)_i = \begin{cases} j_{i+1}^{\mathrm{lh}(x)}(j_2(x-1)) & \text{if } 0 \leq i < \mathrm{lh}(x) \\ 0 & \text{else} \end{cases}$$

Exercise 19 Prove that the functions $\lambda x.\mathrm{lh}(x)$ and $\lambda xi.(x)_i$ are primitive recursive;
Show that $(\langle x_0, \ldots, x_{m-1} \rangle)_i = x_i$ and that $(\langle\,\rangle)_i = 0$;
Show that for all x: either $x = 0$ or $x = \langle (x)_0, \ldots, (x)_{\mathrm{lh}(x)-1} \rangle$.

The *concatenation function* gives for each x and y the code of the sequence which we obtain by putting the sequences coded by x and y after each other, and is written $x \star y$. That means:

$$\begin{array}{rcl} \langle\,\rangle \star y & = & y \\ x \star \langle\,\rangle & = & x \\ \langle (x)_0, \ldots, (x)_{\mathrm{lh}(x)-1} \rangle \star \langle (y)_0, \ldots (y)_{\mathrm{lh}(y)-1} \rangle & = & \langle (x)_0, \ldots (x)_{\mathrm{lh}(x)-1}, (y)_0, \\ & & \ldots, (y)_{\mathrm{lh}(y)-1} \rangle \end{array}$$

Exercise 20 Show that $\lambda xy.x \star y$ primitive recursive. (Hint: you can first define a primitive recursive function $\lambda xy.x \circ y$, satisfying

$$x \circ y = x \star \langle y \rangle$$

Then, define by primitive recursion a function $F(x, y, w)$ by putting

$$\begin{array}{rcl} F(x, y, 0) & = & x \\ F(x, y, w+1) & = & F(x, y, w) \circ (y)_w \end{array}$$

Finally, put $x \star y = F(x, y, \mathrm{lh}(y))$.)

Course-of-values recursion The scheme of primitive recursion:

$$F(y+1, \vec{x}) = H(y, F(y, \vec{x}), \vec{x})$$

allows us to define the value of $F(y+1, \vec{x})$ directly in terms of $F(y, \vec{x})$. Course-of-values recursion is a scheme which defines $F(y+1, \vec{x})$ in terms of *all* previous values $F(0, \vec{x}), \ldots, F(y, \vec{x})$.

Definition 2.1.8 Let $G : \mathbb{N}^k \to \mathbb{N}$ and $H : \mathbb{N}^{k+2} \to \mathbb{N}$ be functions. The function $F : \mathbb{N}^{k+1} \to \mathbb{N}$, defined by the clauses

$$\begin{array}{rcl} F(0, \vec{x}) & = & G(\vec{x}) \\ F(y+1, \vec{x}) & = & H(y, \tilde{F}(y, \vec{x}), \vec{x}) \\ \text{where } \tilde{F}(y, \vec{x}) & = & \langle F(0, \vec{x}), \ldots, F(y, \vec{x}) \rangle \end{array}$$

is said to be defined from G and H by *course-of-values recursion*.

Proposition 2.1.9 *Suppose $G : \mathbb{N}^k \to \mathbb{N}$ and $H : \mathbb{N}^{k+2} \to \mathbb{N}$ are primitive recursive functions and $F : \mathbb{N}^{k+1} \to \mathbb{N}$ is defined from G and H by course-of-values recursion. Then F is primitive recursive.*

Proof. Since

$$\begin{aligned} \tilde{F}(0, \vec{x}) &= \langle G(\vec{x}) \rangle \\ \tilde{F}(y+1, \vec{x}) &= \tilde{F}(y, \vec{x}) * \langle H(y, \tilde{F}(y, \vec{x}), \vec{x}) \rangle \end{aligned}$$

the function $\tilde{F}$ is primitive recursive. Now

$$F(y, \vec{x}) = (\tilde{F}(y, \vec{x}))_y$$

so F is primitive recursive too. ∎

We might also consider the following generalization of the course-of-values recursion scheme: instead of allowing only the values $F(w, \vec{x})$ for $w \leq y$ to be used in the definition of $F(y+1, \vec{x})$, we could allow all values $F(w, \vec{x'})$ (for $w \leq y$). This should be well-defined, for inductively we have already defined all functions $F_w = \lambda \vec{x}.F(w, \vec{x})$ when we are defining F_{y+1}. That this is indeed possible (and does not lead us outside the class of primitive recursive functions) if $\vec{x'}$ is a primitive recursive function of $\vec{x}$, is shown in the following exercise.

Exercise 21 Let $K : \mathbb{N} \to \mathbb{N}$, $G : \mathbb{N}^{k+1} \to \mathbb{N}$ and $H : \mathbb{N}^{k+3} \to \mathbb{N}$ be functions. Define F by:

$$\begin{aligned} F(0, \vec{y}, x) &= G(\vec{y}, x) \\ F(z+1, \vec{y}, x) &= H(z, F(z, \vec{y}, K(x)), \vec{y}, x) \end{aligned}$$

Suppose that G, H and K are primitive recursive.

a) Prove directly, suitably adapting the proof of proposition 2.1.9: if $\forall x(K(x) \leq x)$, then F is primitive recursive.

b) Define a new function F' by:

$$\begin{aligned} F'(0, m, \vec{y}, x) &= G(\vec{y}, K^m(x)) \\ F'(n+1, m, \vec{y}, x) &= H(n, F'(n, m, \vec{y}, x), \vec{y}, K^{m \dot{-} (n+1)}(x)) \end{aligned}$$

Recall that $K^{m \dot{-} (n+1)}$ means: the function K applied $m \dot{-} (n+1)$ times.

Prove: if $n \leq m$ then $\forall k[F'(n, m+k, \vec{y}, x) = F'(n, m, \vec{y}, K^k(x))]$

c) Prove by induction: $F(z, \vec{y}, x) = F'(z, z, \vec{y}, x)$ and conclude that F is primitive recursive, also without the assumption that $K(x) \leq x$.

Double recursion. However, the matter is totally different if, in the definition of $F(y+1, \vec{x})$, we allow values of F_y at arguments in which already known values of F_{y+1} may appear. In this case we speak of *double recursion*. We treat a simple case, with a limited number of variables.

Definition 2.1.10 Let $G : \mathbb{N} \to \mathbb{N}$, $H : \mathbb{N}^2 \to \mathbb{N}$, $K : \mathbb{N}^4 \to \mathbb{N}$, $J : \mathbb{N} \to \mathbb{N}$, en $L : \mathbb{N}^3 \to \mathbb{N}$ be functions; the function F is said to be defined from these by *double recursion* if

$$\begin{aligned} F(0, z) &= G(z) \\ F(y+1, 0) &= H(y, F(y, J(y))) \\ F(y+1, z+1) &= K(y, z, F(y+1, z), F(y, L(y, z, F(y+1, z)))) \end{aligned}$$

Proposition 2.1.11 *If G, H, K, J and L are primitive recursive and F is defined from these by double recursion as in definition 2.1.10 then all functions $F_y = \lambda z.F(y, z)$ are primitive recursive, but F itself need not be primitive recursive.*

Proof. It follows from the definition that all functions F_y are primitive recursive. We give an example of a non-primitive recursive function that can be defined by double recursion. The idea is, to code all definitions of primitive recursive functions $\mathbb{N} \to \mathbb{N}$ as numbers, in the following way:

- The basic functions are the functions $\lambda x.0$, $\lambda x.x+1$ and j_i^m, which get codes $\langle 0\rangle$, $\langle 1\rangle$ and $\langle 2, i, m\rangle$ respectively;
- if $H, G_1, \ldots, G_p$ have codes $n, m_1, \ldots, m_p$ respectively, and F is defined by

  $$F(x) = H(j^p(G_1(x), \ldots, G_p(x)))$$

  then F has code $\langle 3, n, m_1, \ldots, m_p\rangle$;
- if H and G have codes n and m and F is defined by

  $$\begin{array}{rcl} F(j(x,0)) & = & G(x) \\ F(j(x,y+1)) & = & H(j^3(x, F(j(x,y)), y)) \end{array}$$

  then F has code $\langle 4, n, m\rangle$.

Check for yourself that every primitive recursive function of one variable can be defined by the clauses above, and hence has a code (actually, more than one, because there are many definitions of one and the same primitive recursive function).

The next step in the proof is now to define a function Val (actually by double course-of-value recursion) of two variables k and n, such that the following holds: if k is the code of a definition of a primitive recursive function F, then $\mathrm{Val}(k, n) = F(n)$. This is done as follows:

$$\mathrm{Val}(k,x) = \begin{cases} 0 \text{ if } k = \langle 0\rangle \\ x+1 \text{ if } k = \langle 1\rangle \\ j_i^m(x) \text{ if } k = \langle 2, i, m\rangle \\ \mathrm{Val}(n, j^p(\mathrm{Val}(m_1, x), \ldots, \mathrm{Val}(m_p, x))) \\ \qquad \text{if } k = \langle 3, n, m_1, \ldots, m_p\rangle \\ \mathrm{Val}(m, j_1(x)) \text{ if } k = \langle 4, n, m\rangle \text{ and } j_2(x) = 0 \\ \mathrm{Val}(n, j^3(j_1(x), \mathrm{Val}(k, j(j_1(x), j_2(x)-1)), j_2(x)-1)) \\ \qquad \text{if } k = \langle 4, n, m\rangle \text{ and } j_2(x) > 0 \\ 0 \text{ else} \end{cases}$$

Note that $\mathrm{Val}(k, x)$ is defined in terms of $\mathrm{Val}(n, y)$ for $n < k$ or $n = k$ and $y < x$; so Val is well-defined as a function.

The apotheosis of the proof is an example of *diagonalisation*, a form of reasoning similar to Cantor's proof of the uncountability of the set of real numbers; this is a technique we shall meet more often.

Suppose the function Val is primitive recursive. Then so is the function $\lambda x.\mathrm{Val}(x, x)+1$, which is a function of one variable; this function has therefore a code, say k.
But now by construction of Val, we have that $\mathrm{Val}(k, k) = \mathrm{Val}(k, k) + 1$; which is a contradiction. We conclude that the function Val, which was defined by double recursion from primitive recursive functions, is not primitive recursive, which is what we set out to show. ∎

Remark 2.1.12 The class of total computable functions is closed under definition by double recursion, as we shall see below (2.4.7).

In 1927, the Romanian mathematician Sudan ([39]) gave an example of a total computable function which is not primitive recursive. In 1928, W. Ackermann ([1]) gave an example of a function $G(x, y)$ of two variables, defined by double recursion from primitive recursive functions, which has the following property: for every unary primitive recursive function $F(x)$ there is a number x_0 such that for all $x > x_0$, $F(x) < G(x, x)$. Check yourself that it follows that G cannot be primitive recursive! Such functions G are called *Ackermann functions*.

Ackermann's example was later simplified by Rosza Péter; this simplification is presented in the exercise below.

Exercise 22 (Ackermann-Péter) Define by double recursion:

$$\begin{array}{rcl} A(0,x) & = & x+1 \\ A(n+1,0) & = & A(n,1) \\ A(n+1,x+1) & = & A(n,A(n+1,x)) \end{array}$$

Again we write A_n for $\lambda x.A(n,x)$. For a primitive recursive function $F : \mathbb{N}^k \to \mathbb{N}$, we say that F is *bounded by* A_n, written $F \in \mathcal{B}(A_n)$, if for all $x_1, \ldots, x_k$ we have $F(x_1, \ldots, x_k) < A_n(x_1 + \cdots + x_k)$. Prove by inductions on n and x:

i) $n + x < A_n(x)$

ii) $A_n(x) < A_n(x+1)$

iii) $A_n(x) < A_{n+1}(x)$

iv) $A_n(A_{n+1}(x)) \leq A_{n+2}(x)$

v) $nx + 2 \leq A_n(x)$ for $n \geq 1$

vi) $\lambda x.x+1$, $\lambda x.0$ and $\lambda \vec{x}.x_i \in \mathcal{B}(A_1)$

vii) if $F = \lambda \vec{x}.H(G_1(\vec{x}), \ldots, G_p(\vec{x}))$ and $H, G_1, \ldots, G_p \in \mathcal{B}(A_n)$ for some $n > p$, then $F \in \mathcal{B}(A_{n+2})$

viii) for every $n \geq 1$ we have: if $F(0,\vec{x}) = G(\vec{x})$ and $F(y+1,\vec{x}) = H(y, F(y,\vec{x}), \vec{x})$ and $G, H \in \mathcal{B}(A_n)$, then $F \in \mathcal{B}(A_{n+3})$

Concluide that for every primitive recursive function F there is a number n such that $F \in \mathcal{B}(A_n)$; hence, that A is an Ackermann function.

Exercise 23 Define a sequence of functions $G_0, G_1, \ldots : \mathbb{N} \to \mathbb{N}$ by

$$\begin{array}{rcl} G_0(y) & = & y+1 \\ G_{x+1}(y) & = & (G_x)^{y+1}(y) \end{array}$$

and then define G by putting $G(x,y) = G_x(y)$. Give a definition of G by double rcursion and composition (use a definition scheme for double recursion which allows an extra variable) and prove that G is an Ackermann function.

A few simple exercises to conclude this section:

Exercise 24 Show that the following "recursion scheme" does not define a function:

$$\begin{array}{rcl} F(0,0) & = & 0 \\ F(x+1,y) & = & F(y,x+1) \\ F(x,y+1) & = & F(x+1,y) \end{array}$$

Exercise 25 Show that the following "recursion scheme" is not satisfied by any function:

$$\begin{array}{rcl} F(0,0) & = & 0 \\ F(x+1,y) & = & F(x,y+1)+1 \\ F(x,y+1) & = & F(x+1,y)+1 \end{array}$$

2.1.2 A Logical Characterization of the Primitive Recursive Functions

Consider the *language of ordered rings*, that is: the language with symbols for elements 0 and 1, function symbols + and × for addition and multiplication, and a symbol < for the order relation.

The theory Q (*Robinson's Arithmetic*) has the following axioms:

$$\begin{array}{ll} \forall x \neg(x+1=0) & \forall xy(x+1=y+1 \rightarrow x=y) \\ \forall x(\neg(x=0) \rightarrow \exists y(x=y+1)) & \forall x(x+0=x) \\ \forall xy(x+(y+1)=(x+y)+1) & \forall x(x\times 0=0) \\ \forall xy(x\times(y+1)=(x\times y)+x) & \forall xy(x<y \leftrightarrow \exists z(x+(z+1)=y)) \end{array}$$

In a formula in this language, a quantifier $\forall x$ or $\exists x$ is called *bounded* if it occurs as $\forall x(x < t \rightarrow \cdots)$ (respectively, $\exists x(x < t \wedge \cdots)$) where t is a term of the language which does not contain the variable x. A formula is called bounded if every quantifier in it is bounded.

A formula is Σ_1 if it is of the form $\exists x_1 \cdots \exists x_k \psi$, with ψ a bounded formula.

The theory $I\Sigma_1$ extends Q by Σ_1-*induction*:

$$\varphi(0) \wedge \forall x(\varphi(x) \rightarrow \varphi(x+1)) \;\rightarrow\; \forall x \varphi(x)$$

for every Σ_1-formula φ.

For every natural number n, let $\overline{n}$ be the following term in the language of ordered rings:

$$\begin{array}{rcl} \overline{0} & = & 0 \\ \overline{n+1} & = & \overline{n}+1 \end{array}$$

So, $\overline{n} = (\cdots \underbrace{(1+1)+\cdots+1}_{n \text{ times}})$.

Let F be a total k-ary function. If T is an axiom system in the language of ordered rings, we say that F is *provably total* in T if there is a formula $\phi(x_1, \ldots, x_{k+1})$ in the language, such that the sentences:

$$\begin{array}{ll} \psi(\overline{n_1}, \ldots, \overline{n_k}, \overline{F(n_1, \ldots, n_k)}) & \text{for all } n_1, \ldots, n_k \\ \forall \vec{x} yz\,(\psi(\vec{x}, y) \wedge \psi(\vec{x}, z) \rightarrow y = z) & \\ \forall \vec{x} \exists y \psi(\vec{x}, y) & \end{array}$$

are all consequences of T (i.e., true in every model of T).
We have the following theorem.

Theorem 2.1.13 *For a k-ary total function F the following two assertions are equivalent:*

i) *F is primitive recursive*

ii) *F is provably total in $I\Sigma_1$*

For a proof, see [4].

2.2 Partial recursive functions

We return to partial functions.

We shall use the symbol $\simeq$ (*Kleene equality*) between expressions $F(x)$ and $G(x)$ for partial functions: $F(x) \simeq G(x)$ means that $F(x)$ is defined precisely when $G(x)$ is defined, and whenever this is the case, $F(x) = G(x)$. In particular, $F(x) \simeq G(x)$ holds if both sides are undefined.

Composite terms built up from partial functions are interpreted in the way we have defined composition. That means, that a term cannot be defined unless all its subterms are defined. Example: if Π^2_1 denotes the first projection $\mathbb{N}^2 \rightarrow \mathbb{N}$ as before, and G is a unary partial function, then $\Pi^2_1(x, G(y))$ is only defined when $G(y)$ is defined, and $\Pi^2_1(x, G(y)) \simeq x$ need not hold.

Definition 2.2.1 The class of *partial recursive functions* is generated by the following clauses:

i) all primitive recursive functions are partial recursive

ii) the partial recursive functions are closed under definition by minimalization

iii) whenever G is a k-ary partial recursive function and F is a unary primitive recursive function, then $\lambda\vec{x}.F(G(\vec{x}))$ is k-ary partial recursive.

Definition 2.2.2 A relation $A \subseteq \mathbb{N}^k$ is called recursive if its characteristic function χ_A is partial recursive.

A partial recursive function is *total recursive* or *recursive* if it is total. Because χ_A is always a total function for every relation A, there is no notion of "partial recursive relation".

Proposition 2.2.3

i) If R is a $k+1$-ary recursive relation and the k-ary partial function F is defined by

$$F(\vec{x}) \simeq \mu y.R(\vec{x}, y)$$

then F is partial recursive;

ii) If R is a recursive relation and G is a primitive recursive function, and F is defined by

$$F(x) \simeq \begin{cases} G(x) \text{ if } \exists y.R(y,x) \\ \text{undefined else} \end{cases}$$

then F is partial recursive;

Proof. For,

i) $F(\vec{x}) \simeq \mu y.\chi_R(\vec{x}, y) = 0$

ii) $F(x) \simeq G((\mu y.\chi_R(y, x) = 0)0 + x)$. Recall our convention about when terms are defined!

∎

2.3 Coding of RM-programs and the equality Computable = Recursive

Our first goal in this section is to show that the class of partial computable functions coincides with the class of partial recursive functions. We do this by coding programs and computations in such a way that we can show the following:

> The relation T, defined by
>
> > $T(m, e, x, y)$ holds if and only if e is the code of a program P and y is the code of a terminating computation with P and input $j_1^m(x), \ldots, j_m^m(x)$
>
> is primitive recursive;
>
> There is a primitive recursive function U such that whenever $T(m, e, x, y)$, $U(y)$ is the result of the computation coded by y (i.e., the content of the first register in the final state).

This suffices for our first goal. For, suppose F is a k-ary partial computable function. Then according to definition 1.2.1 there is a program P which computes F; say P has code e. Again by definition 1.2.1, we have that

$$F(x_1, \ldots, x_k) \simeq U(\mu y.T(k, e, j^k(x_1, \ldots, x_k), y))$$

so F is defined by minimalization from the primitive recursive relation T (and the primitive recursive function j^k), hence partial recursive by definition 2.2.1.

The converse is a consequence of exercise 5 and Theorem 1.2.6: every primitive recursive function is computable, and the computable functions are closed under minimalization, so every partial recursive function is computable.

The coding of programs and computations is completely straightforward.

First we code every basic command of an RM program as follows:

$$\begin{array}{rll} r_i^+ \Rightarrow j & \text{gets code} & \langle i,j\rangle \\ r_i^- \Rightarrow j,k & \text{gets code} & \langle i,j,k\rangle \end{array}$$

Then, we code a program P, which is a list of commands $(p_1,\ldots,p_n)$, as $\langle \overline{p}_1,\ldots,\overline{p}_n\rangle$, where $\overline{p}_i$ is the code of p_i.

The codes of programs form a primitive recursive set Prog:

$$e \in \text{Prog} \iff \forall i < \text{lh}(e)(\,((e)_i)_0 \geq 1 \wedge (\text{lh}((e)_i) = 2 \vee \text{lh}((e)_i) = 3))$$

Exercise 26 Denote the code of a program P by $\overline{P}$. Recall the functions $P \mapsto P'$ and the composition $P,Q \mapsto PQ$ from definition 1.1.4.

Show that there exist primitive recursive functions F and G suvh that for all programs P and Q,

$$\begin{array}{rcl} F(\overline{P}) & = & \overline{P'} \\ G(\overline{P},\overline{Q}) & = & \overline{PQ} \end{array}$$

The definition of the relation T is now a direct translation of definition 1.1.2. The statement $T(e,m,x,y)$ is the conjunction of the following statements:

$\text{Prog}(e)$

$\text{lh}(y) > 0$

$\forall i < \text{lh}(y)[\text{lh}((y)_i) = \text{lh}((y)_0)]$

$\text{lh}((y)_0) \geq m+1$

$((y)_0)_0 = 1 \wedge \forall i \leq m\,(1 \leq i \Rightarrow ((y)_0)_i = j_i^m(x))$

$\forall i < \text{lh}((y)_0)\,(m < i \Rightarrow ((y)_0)_i = 0)$

$\forall i < \text{lh}(y)[((y)_i)_0 > \text{lh}(e) \Leftrightarrow i = \text{lh}(y) - 1]$

$\forall k,l < e\,\forall i < \text{lh}(y) - 1[(e)_{((y)_i)_0} = \langle k,l\rangle \Rightarrow$

$$(y)_{i+1} = (y)_i[l/((y)_i)_0, ((y)_i)_k + 1/((y)_i)_k]$$

$\forall k,l,m < e\,\forall i < \text{lh}(y) - 1[(e)_{((y)_i)_0} = \langle k,l,m\rangle \Rightarrow$

$$\{((y)_i)_k = 0 \ \wedge\ (y)_{i+1} = (y)_i[m/((y)_i)_0]\} \vee$$

$$\{((y)_i)_k > 0 \ \wedge\ (y)_{i+1} = (y)_i[l/((y)_i)_0, ((y)_i)_k - 1/((y)_i)_k]\}]$$

Here the notation $y[a/(y)_i]$ stands for the code of the the sequence which results from the sequence coded by y, by replacing its $i+1$-st element, i.e. the number $(y)_i$, by a; similarly, $y[a/(y)_i, b/(y)_j]$ is the code of the sequence which results from the sequence coded by y by making two replacements. You should check that the functions

$$\begin{array}{rcl} y,a,i & \mapsto & y[a/(y)_i] \\ y,a,i,b,j & \mapsto & y[a/(y)_i, b/(y)_j] \end{array}$$

are primitive recursive.

Convince yourself, by going over these statements, that $T(e, m, x, y)$ has the intended meaning, and that it is primitive recursive.

Because the result of a computation is the number stored in the first register when the machine stops, the function

$$U(y) = ((y)_{\mathrm{lh}(y)-1})_1$$

gives the result; and U is clearly primitive recursive.

The letters T and U are standard in Computability Theory. The relation T is also called the *Kleene T-predicate* and U is often called the *output function.*

We summarize our observations in the following theorem.

Theorem 2.3.1 (Enumeration Theorem)

i) A partial function is computable if and only if it is partial recursive.

ii) Define the partial recursive function Φ by

$$\Phi(m, e, x) \simeq U(\mu y.T(m, e, x, y))$$

Then for every k-ary partial recursive function F there is a number e such that for all k-tuples $x_1, \ldots, x_k$:

$$F(x_1, \ldots, x_k) \simeq \Phi(k, e, j^k(x_1, \ldots, x_k))$$

In other words, we have a partial recursive enumeration of the partial recursive functions.

Corollary 2.3.2 *The partial recursive functions are closed under composition and primitive recursion.*

Since the function Φ from Theorem 2.3.1 is partial recursive, there is a program which computes it; such a program is called a *universal program.*

By contrast, there is no analogous theorem for *total* recursive functions:

Proposition 2.3.3 *There is no total recursive function $\Psi(m, e, x)$ witrh the property that for every k-ary total recursive function F there is a number e such that*

$$F \;=\; \lambda x_1 \cdots x_m.\Psi(m, e, j^m(x_1, \ldots, x_m))$$

Proof. For suppose such a function Ψ exists; we argue by diagonalization as in the proof of proposition 2.1.11. The function

$$G(x_1 \cdots x_m) \;=\; \Psi(m, j^m(x_1, \ldots, x_m), j^m(x_1, \ldots, x_m)) + 1$$

is total recursive, and therefore there should be an e such that for all $x_1, \ldots, x_m$: $G(x_1 \cdots x_m) = \Psi(m, e, j^m(x_1, \ldots, x_m))$. However, for such e we would have

$$\Psi(m, e, e) = \Psi(m, e, j^m(j_1^m(e), \ldots, j_m^m(e))) = G(j_1^m(e), \ldots, j_m^m(e)) = \Psi(m, e, e) + 1$$

which is a clear contradiction. ∎

Exercise 27 Show that for every m-ary partial recursive function F there exist infinitely many numbers e such that

$$F = \lambda x_1 \cdots x_m.\Phi(m, e, j^m(x_1, \ldots, x_m))$$

Exercise 28 Let $R_1, \ldots, R_n \subseteq \mathbb{N}^k$ be recursive relations such that $R_i \cap R_j = \emptyset$ for $i \neq j$; let $G_1, \ldots, G_n$ be k-ary partial recursive functions. Then the partial function F, defined by

$$F(\vec{x}) \simeq \begin{cases} G_1(\vec{x}) & \text{if } R_1(\vec{x}) \\ \vdots & \vdots \\ G_n(\vec{x}) & \text{if } R_n(\vec{x}) \\ \text{undefined} & \text{else} \end{cases}$$

is also partial recursive; prove this.

2.4 *Smn*-Theorem and Recursion Theorem

If $F = \lambda x_1 \cdots x_m.\Phi(m, e, j^m(x_1, \ldots, x_m))$, then e is called an *index* for F, and we write ϕ_e (or sometimes $\phi_e^{(m)}$ if we want to make the arity of F explicit) for F.

However, subscripts are a bore, especially in compound terms, and therefore we shall write $e{\cdot}(x_1, \ldots, x_m)$ for $\phi_e(x_1, \ldots, x_m)$[1].

The following theorem has an odd name: "*Smn*-theorem". A better name would be "parametrization theorem". If we have an $(m+n)$-ary partial recursive function ϕ_e and m numbers $a_1, \ldots, a_m$, then the n-ary partial function $(x_1, \ldots, x_n) \mapsto \phi_e(a_1, \ldots, a_m, x_1, \ldots, x_n)$ is also partial recursive. The theorem says that an index for this last partial function can be obtained primitive-recursively in e and $a_1, \ldots, a_m$. We sketch a proof.

Theorem 2.4.1 (*Smn*-theorem; Kleene) *For every $m \geq 1$ and $n \geq 1$ there is an $m+1$-ary primitive recursive function S_n^m such that for all $e, x_1, \ldots, x_m, y_1, \ldots, y_n$:*

$$S_n^m(e, x_1, \ldots, x_m){\cdot}(y_1, \ldots, y_n) \simeq e{\cdot}(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

Proof. If e is not the code of an RM-program we put $S_n^m(e, x_1, \ldots, x_m) = e$. If e codes a program P, then for every $x_1, \ldots, x_m$, the number $S_n^m(e, x_1, \ldots, x_m)$ should code a program that performs the following computation:

	$b_1 \cdots b_n$
remove junk	$\Downarrow$
	$b_1 \cdots b_n \, \vec{0}$
input register contents $\vec{x}$	$\Downarrow (R_1^+)^{x_1} \cdots (R_m^+)^{x_m}$
	$x_1 \cdots x_m \, b_1 \cdots b_n$
	$\Downarrow \; P$
	$e{\cdot}(x_1, \ldots, x_m, b_1, \ldots b_n) \, \vec{c}$

(Here, $(R_i^+)^n$ is the program that adds 1 to the i-th register n times) It is left to you to convince yourself that S_n^m is primitive recursive. ∎

The result of the following exercise will be used in the sequel!

Exercise 29 Show, by modifying the proof of Theorem 2.4.1 a bit, that the function S_n^m can be assumed to be injective.

We have already noted (corollary 2.3.2) that the partial recursive functions are closed under composition. The following corollary of Theorem 2.4.1 states that one can find an index for a composition $G \circ F$ of partial recursive functions, primitive-recursively in indices for G and F. We restrict to the case of composition of two unary functions; the general case is left to you to formulate and prove.

Corollary 2.4.2 *There is a primitive recursive function H such that for all e, f, x the following holds:*

$$H(e, f){\cdot}x \simeq e{\cdot}(f{\cdot}x)$$

Proof. The function $\lambda e f x.e{\cdot}(f{\cdot}x)$ is partial recursive. Indeed,

$$e{\cdot}(f{\cdot}x) \simeq U(j_2(\mu z.[T(1, f, x, j_1(z)) \wedge T(1, e, U(j_1(z)), j_2(z))]))$$

Therefore, there exists a number g such that $e{\cdot}(f{\cdot}x) \simeq g{\cdot}(e, f, x)$ for all e, f, x; put $H(e, f) = S_1^2(g, e, f)$ ∎

Exercise 30 Formulate and prove a generalization of corollary 2.4.2 for arbitrary compositions (as in definition 1.2.2).

[1] Kleene invented the terrible notation $\{e\}(x_1, \ldots, x_m)$

The following theorem is a surprising consequence of the *Smn*-theorem. It allows us to define a partial recursive function F in terms of an index for F!

Theorem 2.4.3 (Recursion Theorem, Kleene 1938) *For every $k \geq 1$ and $k+1$-ary partial recursive function F there exists an index e such that for all $x_1, \ldots, x_k$ the following holds:*

$$e\cdot(x_1, \ldots x_k) \simeq F(x_1, \ldots, x_k, e)$$

Moreover, there is a primitive recursive function which produces such an e for every index of F.

Proof. Suppose f is an index for F, so $f\cdot(x_1, \ldots, x_{k+1}) \simeq F(x_1, \ldots, x_{k+1})$ for all $x_1, \ldots, x_{k+1}$. Now, choose an index g satisfying for all $h, y, x_1, \ldots, x_k$:

$$g\cdot(h, y, x_1, \ldots, x_k) \simeq h\cdot(x_1, \ldots, x_k, S^1_k(y, y))$$

(Such g exists because the function on the right hand side is clearly partial recursive)
Now let

$$e = S^1_k(S^1_{k+1}(g, f), S^1_{k+1}(g, f))$$

Clearly, e depends primitive-recursively on f. Moreover,

$$\begin{array}{ccll} e\cdot(x_1, \ldots, x_k) & \simeq & \\ S^1_k(S^1_{k+1}(g, f), S^1_{k+1}(g, f))\cdot(x_1, \ldots, x_k) & \simeq & \text{by the } Smn\text{-theorem} \\ S^1_{k+1}(g, f)\cdot(S^1_{k+1}(g, f), x_1, \ldots, x_k) & \simeq & \\ g\cdot(f, S^1_{k+1}(g, f), x_1, \ldots, x_k) & \simeq & \text{by choice of } g \\ f\cdot(x_1, \ldots, x_k, S^1_k(S^1_{k+1}(g, f), S^1_{k+1}(g, f))) & \simeq & \text{by definition of } e \\ f\cdot(x_1, \ldots, x_k, e) & \simeq & \text{by assumption on } f \\ F(x_1, \ldots, x_k, e) & & \end{array}$$

∎

Theorem 2.4.3 is often called (e.g. in [26]) the Second Recursion Theorem. The First Recursion Theorem is Theorem 2.4.5 below.

Corollary 2.4.4 *Let G be k-ary partial recursive, and H $k+2$-ary partial recursive. Then an index for the function F which is defined from G and H by primitive recursion, can be found primitive-recursively in indices for G and H.*

Proof. The function

$$L(g, h, y, \vec{x}, f) \simeq \begin{cases} g\cdot(\vec{x}) & \text{if } y = 0 \\ h\cdot(y-1, f\cdot(y-1, \vec{x}), \vec{x}) & \text{if } y > 0 \end{cases}$$

is clearly partial recursive; let l be an index for L. By Theorem 2.4.3 we can find an index f such that for all $y, \vec{x}$ we have: $f\cdot(y, \vec{x}) \simeq S^2_{k+2}(l, g, h)\cdot(y, \vec{x}, f)$. And moreover, f is found primitive-recursively in $S^2_{k+2}(l, g, h)$ hence primitive rcursively in g, h.
Applying the *Smn*-theorem, we find that $f\cdot(y, \vec{x}) \simeq L(g, h, y, \vec{x}, f)$ and thus, if g and h are indices for G and H respectively, that f is an index for the function defined from G and H by primitive recursion. ∎

The recursive relations are closed under bounded quantifiers: if $R \subseteq \mathbb{N}^{k+1}$ is recursive, then so are

$$\{(\vec{x}, y) \mid \forall w < y.R(\vec{x}, w)\}$$

and

$$\{(\vec{x}, y) \mid \exists w < y.R(\vec{x}, w)\}$$

because their characteristic functions are defined by primitive recursion from those of R (see proposition 2.1.2iii)). And again, an index for the characteristic function of a relation defined by bounded quantification from R can be obtained primitive-recursively from an index for the characteristic function of R.

Exercise 31

i) Show that for every recursive relation R, there is an e such that for all $\vec{x}$:

$$e{\cdot}(\vec{x}) \simeq \begin{cases} 0 & \text{if } R(\vec{x}, e) \\ 1 & \text{else} \end{cases}$$

ii) Show that for every recursive relation R and every partial recursive function F, there is an e such that for all $\vec{x}$:

$$e{\cdot}(\vec{x}) \simeq \begin{cases} F(\vec{x}) & \text{if } \exists y.R(\vec{x}, y, e) \\ \text{undefined} & \text{else} \end{cases}$$

The following theorem is often called (e.g. in [26]) the First Recursion Theorem, or Fixpoint Theorem.

Corollary 2.4.5 (Fixpoint Theorem) *For every total recursive function F and every n there is a number e such that e and $F(e)$ are indices for the same n-ary partial recursive function:*

$$\phi_e^{(n)} = \phi_{F(e)}^{(n)}$$

Proof. Let G be the partial recursive function defined by

$$G(\vec{x}, e) \simeq F(e){\cdot}(\vec{x})$$

Apply Theorem 2.4.3 to find an index e satisfying

$$e{\cdot}(\vec{x}) \simeq G(\vec{x}, e)$$

∎

Remark 2.4.6 Of course, Corollary 2.4.5 is not a "Fixpoint Theorem" in the usual sense of the word: there is no operation of which it is asserted that this operation has a fixed point. Observe that the function F is not an operation on partial recursive functions, but only on indices.

Exercise 32 Prove the primitive recursive version of corollary 2.4.5. That is: there is a primitive recursive function T satisfying for all f and $\vec{x}$:

$$T(f){\cdot}(\vec{x}) \simeq (f{\cdot}T(f)){\cdot}(\vec{x})$$

Exercise 33 Prove the *recursion theorem with parameters*: there is a primitive recursive function F satisfying for all $f, \vec{y}, \vec{x}$:

$$F(f, \vec{y}){\cdot}(\vec{x}) \simeq f{\cdot}(F(f, \vec{y}), \vec{y}, \vec{x})$$

and also: there is a primitive recursive F' such that for all $f, \vec{y}, \vec{x}$:

$$F'(f, \vec{y}){\cdot}(\vec{x}) \simeq (f{\cdot}(F'(f, \vec{y}), \vec{y})){\cdot}(\vec{x})$$

Remark 2.4.7 We conclude this chapter with the promised proof that the class of total computable functions is closed under definition by double recursion. Assume therefore that G, H, J, K and L are total recursive, and that the function F is defined by:

$$\begin{aligned} F(0, z) &= G(z) \\ F(y+1, 0) &= H(y, F(y, J(y))) \\ F(y+1, z+1) &= K(y, z, F(y+1, z), F(y, L(y, z, F(y+1, z)))) \end{aligned}$$

Then F is total recursive, for by theorem 2.4.3 we can find an index f satisfying

$$f{\cdot}(y, z) \simeq \begin{cases} G(z) & \text{if } y = 0 \\ H(y-1, f{\cdot}(y-1, J(y-1))) & \text{if } y > 0 \\ & \text{and } z = 0 \\ K(y-1, z-1, f{\cdot}(y, z-1), f{\cdot}(y-1, L(y-1, z-1, f{\cdot}(y, z-1)))) & \text{if } y > 0 \\ & \text{and } z > 0 \end{cases}$$

Exercise 34 Prove by double induction (on y and z) that the function ϕ_f is total and equal to F.

One more exercise.

Exercise 35 Prove *Smullyan's Simultaneous Recursion Theorem*: given two binary partial recursive functions F and G, for every k there exist indices a and b satisfying for all $x_1, \ldots, x_k$:

$$a{\cdot}(x_1, \ldots, x_k) \simeq F(a, b){\cdot}(x_1, \ldots, x_k)$$

and

$$b{\cdot}(x_1, \ldots, x_k) \simeq G(a, b){\cdot}(x_1, \ldots, x_k)$$

Chapter 3

Undecidability and Recursively Enumerable Sets

3.1 Solvable Problems

Every subset of $\mathbb{N}^k$ constitutes a "problem": the problem of determining whether a given k-tuple of natural numbers belongs to the set. For example the set

$$\{(f,x) \mid \exists z T(1,f,x,z)\}$$

is the problem of deciding whether or not $f{\cdot}x$ is defined.

We call a problem (i.e., set) *solvable* if the set is recursive. Another word is *decidable*.

The above mentioned problem: is $f{\cdot}x$ defined?, is classical and is called the *Halting Problem* (Turing): is the computation on the RM with program coded by f and input x finite?

Proposition 3.1.1 *The Halting Problem is not solvable.*

Proof. Suppose for a contradiction that we have a total recursive function F satisfying for all f, x:

$$F(f,x) \simeq \begin{cases} 0 & \text{if } f{\cdot}x \text{ is defined} \\ 1 & \text{otherwise} \end{cases}$$

Let G be a partial recursive function such that $\mathrm{dom}(G) = \mathbb{N} - \{0\}$ (for example, $G(x) \simeq \mu z.xz > 1$). Then by the Recursion Theorem (2.4.3) there is an index f such that for all x the following holds:

$$f{\cdot}x \simeq G(F(f,x))$$

But then, surely $f{\cdot}x$ is defined if and only if $F(f,x) \neq 0$; which is the case if and only if $f{\cdot}x$ is undefined. We have the desired contradiction. ∎

There is a simpler proof. The *Standard Problem* is: is $x{\cdot}x$ defined?

Clearly, if we can solve the Halting problem, then we can solve the Standard Problem, as you can check for yourself. Now suppose the Standard Problem is solvable. Then we have a total recursive function F such that for all x the following holds:

$$F(x) \simeq \begin{cases} 0 & \text{if } x{\cdot}x \text{ is defined} \\ 1 & \text{otherwise} \end{cases}$$

Let G be as in the proof of 3.1.1, and let f be an index for the partial recursive function $\lambda x.G(F(x))$.

We then have: $f{\cdot}f$ is defined if and only if $F(f) = 0$, which is the case if and only if $G(F(f))$, which is equal to $f{\cdot}f$, is undefined; contradiction. So it was not really necessary to appeal to the Recursion Theorem in the proof of 3.1.1.

The relation between the Standard Problem and the Halting Problem is an example of *reducibility* of one problem to another: if $R \subseteq \mathbb{N}^m$ and $S \subseteq \mathbb{N}^k$ are problems then R is said to be *reducible* to S if there exist k m-ary total recursive functions $F_1, \ldots, F_k$ such that for every m-tuple $\vec{x}$ we have:

$$\vec{x} \in R \Leftrightarrow (F_1(\vec{x}), \ldots, F_k(\vec{x})) \in S$$

Exercise 36 Show: if R is reducible to S and S is solvable, then so is R.

Later on, we shall deal with reducibility in more detail. A variation of the notion of solvability is *solvability with respect to*. The problem is then, to determine whether $\vec{x} \in R$ for $\vec{x} \in A$. We say: is R solvable with respect to A?

Definition 3.1.2 Let R and A be subsets of $\mathbb{N}^k$. We say that R is *solvable with respect to* A if there is a k-ary partial recursive function F such that $A \subseteq \mathrm{dom}(F)$ and for all $\vec{x} \in A$, $F(\vec{x}) = \chi_R(\vec{x})$.

Example 3.1.3 Denote the range of a function F by $\mathrm{rge}(F)$. The problem: is $0 \in \mathrm{rge}(\phi_e)$?, is not solvable with respect to $\{e \mid \phi_e \text{ is total}\}$.

Proof. We must show that there is no partial recursive function F which satisfies the following: whenever ϕ_e is total, $F(e)$ is defined and we have:

$$F(e) = \begin{cases} 0 & \text{if there exists a } z \text{ such that } e{\cdot}z = 0 \\ 1 & \text{otherwise} \end{cases}$$

Let g be an index such that for all x, y:

$$g{\cdot}(x,y) \simeq \begin{cases} 0 & \text{if } T(1,x,x,y) \\ 1 & \text{otherwise} \end{cases}$$

Then $S^1_1(g,x)$ is the index of a total recursive function, for $S^1_1(g,x){\cdot}y \simeq g{\cdot}(x,y)$. We have:

$$0 \in \mathrm{rge}(\phi_{S^1_1(g,x)}) \Leftrightarrow \exists y.T(1,x,x,y) \Leftrightarrow x{\cdot}x \text{ is defined}$$

Therefore, if such an F existed, the function $G = \lambda x.F(S^1_1(g,x))$ would be a solution of the Standard Problem; which contradicts 3.1.1. ∎

Exercise 37 Prove that the problem: is ϕ_e total?, is not solvable.
[Hint: first define a total recursive function F such that for all e the following holds: $e{\cdot}e$ is defined if and only if $\phi_{F(e)}$ is total. Then conclude the statement from this]

Exercise 38 Prove that the following problems are undecidable:

i) $\mathrm{dom}(\phi_e) = \emptyset$?

ii) $\mathrm{rge}(\phi_e)$ infinite?

iii) $\phi_e = \phi_f$?

Exercise 39 Determine which of the following problems are solvable with respect to indices of total functions:

i) $\exists x.e{\cdot}x \neq 0$

ii) $\exists x.e{\cdot}x \leq e{\cdot}(x+1)$

iii) $\exists x.e{\cdot}x \geq e{\cdot}(x+1)$

iv) $e{\cdot}x = y$ (as ternary relation)

v) ϕ_e has infinite range

Exercise 40 Show that there is no binary total recursive function F such that for all e, x the following holds:

$$e{\cdot}x \text{ is defined } \implies \exists y(T(1, e, x, y) \wedge \mathrm{lh}(y) \leq F(e, x))$$

[Hint: show that there is a primitive recursive function $S(e, x, n)$ such that $S(e, x, n)$ gives the first n stages of a computation of the RM with a program with code e and input x, provided Prog(e); then, conclude from existence of an F as described, that the Halting Problem is decidable]

Exercise 41 Show that there is no total recursive function F which satisfies the following, for all e: if ϕ_e is the characteristic function of a finite set, then $F(e)$ is an upper bound for this set.
[Hint: consider the set $\{\mathrm{lh}(\mu y.T(1, e, x, y))\}$. Show that an index for the characteristic function of this set can be obtained primitive-recursively in e and x]

Definition 3.1.4 A subset X of $\mathbb{N}$ is called *extensional for indices of partial recursive functions* (or *extensional* for short) if for all $e, f \in \mathbb{N}$ the following holds:

$$\text{if } \phi_e = \phi_f \text{ and } e \in X \text{ then also } f \in X$$

Another terminology one encounters in the literature, is *index set.*

Note, that the properties in exercise 39 are all extensional.

Rice's Theorem says that nontrivial extensional properties of indices for partial recursive functions can never be decidable:

Theorem 3.1.5 (Rice) *If $X \subseteq \mathbb{N}$ is recursive and extensional, then $X = \emptyset$ or $X = \mathbb{N}$.*

Proof. Note that if X is recursive and extensional, then so is its complement; therefore, if $X \neq \emptyset$, $X \neq \mathbb{N}$ then we may assume that all indices for the empty partial function are not members of X. Let f be such an index, and pick $e \in X$.
By the *Smn*-Theorem there is a primitive recursive function F such that for all x, y:

$$F(x){\cdot}y \;\simeq\; \begin{cases} e{\cdot}y & \text{if } \exists z T(1, x, x, z) \\ \text{undefined} & \text{else} \end{cases}$$

We have:
If $x{\cdot}x$ is defined then $\phi_{F(x)} = \phi_e$, so $F(x) \in X$; and if $x{\cdot}x$ is undefined, then $\phi_{F(x)} = \phi_f$, so $F(x) \notin X$. Therefore the Standard Set $\{x \mid x{\cdot}x \text{ is defined}\}$ is reducible to X via F. Hence X cannot ne decidable. ∎

For lovers of the Recursion Theorem an alternative proof of Theorem 3.1.5: Let X be recursive and extensional, $e \in X$, $f \notin X$ with f an index for the empty function. By the Recursion Theorem there is an index g such that for all x the following holds:

$$g{\cdot}x \;\simeq\; \begin{cases} f{\cdot}x & \text{if } g \in X \\ e{\cdot}x & \text{otherwise} \end{cases}$$

Since X is extensional we then have:

$$\begin{array}{lllll} g \in X & \Rightarrow & \phi_g = \phi_f & \Rightarrow & g \notin X \\ g \notin X & \Rightarrow & \phi_g = \phi_e & \Rightarrow & g \in X \end{array}$$

which is a contradiction.

3.2 Recursively Enumerable Sets

From our point of view the recursive sets are the simplest problems: they are solvable. One step up, we find the *recursively enumerable* (also called: computably enumerable) sets.

Definition 3.2.1 A subset X of $\mathbb{N}$ is called *recursively enumerable* (abbreviated: *r.e.*) if there is a partial recursive function ψ such that $R = \mathrm{dom}(\psi)$.

Proposition 3.2.2 *The following four statements are equivalent for a subset R of $\mathbb{N}$:*

i) R *is recursively enumerable*

ii) *There is a recursive relation $S \subseteq \mathbb{N}^2$ such that $R = \{x \mid \exists y((x,y) \in S)\}$*

iii) *There is a partial recursive function F such that $R = \mathrm{rge}(F)$*

iv) $R = \emptyset$ *or there is a primitive recursive function F such that $R = \mathrm{rge}(F)$*

Moreover, the implications i)⇒ii)⇒iii)⇒i) are primitive-recursively uniform in indices. *This means, for example, for i)⇒ii), that there is a primitive recursive function G such that, whenever f is an index for a partial recursive function which testifies that R is recursively enumerable, then $G(f)$ is an index for the characteristic function of a relation S which testifies ii).*

Proof. i)⇒ii). Suppose $R = \mathrm{dom}(\psi)$; let f be an index for ψ. Then $R = \{x \mid \exists y T(1, f, x, y)\}$; because T is primitive recursive, there is an index g such that for all h, x, y:

$$g{\cdot}(h,x,y) \;\simeq\; \begin{cases} 0 & \text{if } T(1,f,x,y) \\ 1 & \text{otherwise} \end{cases}$$

It follows that $\mathrm{dom}(\phi_f) = \{x \mid \exists y(S_2^1(g,f){\cdot}(x,y) = 0)\}$, and $\phi_{S_2^1(g,f)}$ is always an index for a characteristic function.
ii)⇒iii). We can choose an index g, such that whenever f is an index for a characteristic function F and $R = \{x \mid \exists y(f{\cdot}(x,y) = 0)\}$, then for all x the following holds:

$$g{\cdot}(f,x) \;\simeq\; \begin{cases} x & \text{if } \exists y(f{\cdot}(x,y) = 0) \\ \text{undefined} & \text{otherwise} \end{cases}$$

It is left to you to check that $R = \mathrm{rge}(\phi_{S_1^1(g,f)})$.
iii)⇒i). Suppose $R = \mathrm{rge}(\phi_f)$. Then $R = \mathrm{dom}(\phi_{S_1^1(g,f)})$, where g is an index such that for all f, x:

$$g{\cdot}(f,x) \;\simeq\; \begin{cases} 0 & \text{if } \exists y[T(1,f,j_1(y),j_2(y)) \text{ and } U(j_2(y)) = x] \\ \text{undefined} & \text{otherwise} \end{cases}$$

iv)⇒iii) is clear.
iii)⇒iv). If $R = \mathrm{rge}(F)$ then either $R = \emptyset$ or there is some $a \in R$. In the first case we are done; in the other case, let f be an index for F, and let g be an index such that

$$g{\cdot}(f,y) \;\simeq\; \begin{cases} a & \text{if } \neg T(1,f,j_1(y),j_2(y)) \\ U(j_2(y)) & \text{if } T(1,f,j_1(y),j_2(y)) \end{cases}$$

Then $R = \mathrm{rge}(\phi_{S_1^1(g,f)})$, and $\phi_{S_1^1(g,f)}$ is clearly primitive recursive. ∎

The equivalence of statement iv) of proposition 3.2.2 to the other statements cannot be made primitive-recursively uniform in indices, because of the case distinction whether $\mathrm{rge}(\phi_f) = \emptyset$ or not. This cannot be decided recursively, as we saw (exercise 38i)).

Remark 3.2.3 We extend the definition of r.e. sets to subsets of $\mathbb{N}^k$ by saying that a k-ary relation R is r.e. if R is the domain of a k-ary partial recursive function.
We have the following notation for r.e. sets:

$$W_e^k \;=\; \mathrm{dom}(\phi_e^{(k)})$$

If $R = W_e^k$ then e is said to be an *r.e. index* for R.
We often write W_e instead of $W_e^{(1)}$.

Exercise 42 Prove: a k-ary relation R is r.e. if and only if $\{j^k(x_1, \ldots, x_k) \,|\, \vec{x} \in R\}$ is an r.e. subset of $\mathbb{N}$.

Example 3.2.4

i) The unsolvability of the Standard Problem means that the set

$$\mathcal{K} = \{x \mid x{\cdot}x \text{ is defined}\}$$

is not recursive; $\mathcal{K}$ is r.e. for $\mathcal{K} = \{x \mid \exists y.T(1, x, x, y)\}$ hence satisfies ii) of proposition 3.2.2 (the symbol $\mathcal{K}$ is standard in the literature, for this set).

ii) Every recursive set is r.e.

Exercise 43 Prove: if S is r.e. and R is reducible to S, then R is r.e.

Exercise 44 Prove that $\{e \mid \phi_e \text{ is total}\}$ is *not* r.e.

Exercise 45 Prove that every r.e. set is of the form $\{x \mid \exists y((x, y) \in S)\}$ for some *primitive* recursive $S \subseteq \mathbb{N}^2$.

Proposition 3.2.5 (Post)

i) $R \subseteq \mathbb{N}^k$ *is recursive if and only if both* R *and its complement are r.e.;*

ii) $\mathbb{N} - \mathcal{K} = \{x \mid x{\cdot}x$ *is undefined*$\}$ *is not r.e.;*

iii) A k*-ary partial function is partial recursive if and only if its graph:* $\{(\vec{x}, y) \mid F(\vec{x}) = y\}$*, is r.e.;*

iv) if a k*-ary function is total recursive, then its graph is recursive.*

Proof.

i) If R is recursive, then so is its complement, so both are r.e.

Conversely, if $R = \{\vec{x} \mid \exists y S(\vec{x}, y)\}$ and $\mathbb{N}^k - R = \{\vec{x} \mid \exists y.T(\vec{x}, y)\}$ for recursive S and T, then define a partial recursive k-ary function F by $F(\vec{x}) \simeq \mu y.(S(\vec{x}, y) \vee T(\vec{x}, y))$. The function F is total and we have: $\vec{x} \in R$ if and only if $(\vec{x}, F(\vec{x})) \in S$. Hence R is recursive.

ii) This follows from i) because if $\mathbb{N} - \mathcal{K}$ were r.e. then $\mathcal{K}$ would be recursive, which is not the case.

iii) Let f be an index for F. The graph of F is

$$\{(\vec{x}, y) \mid y = F(\vec{x})\} = \{(\vec{x}, y) \mid \exists w[T(k, f, j^k(\vec{x}), w) \text{ and } U(w) = y]\}$$

and is therefore r.e.; conversely, if

$$\{(\vec{x}, y) \mid y = F(\vec{x})\} = \{(\vec{x}, y) \mid \exists z S(\vec{x}, y, z)\}$$

for some recursive S, then $F = \phi_g$, where g is an index such that

$$\forall \vec{x}[g{\cdot}(\vec{x}) \simeq j_1(\mu z.S(\vec{x}, j_1(z), j_2(z)))]$$

iv) The graph of F is r.e. by iii) but since F is total, the complement of the graph of F is

$$\{(\vec{x}, y) \mid \exists z[T(k, f, j^k(\vec{x}), z) \text{ and } U(z) \neq y]\}$$

for any index f for F; and is therefore also r.e. Hence the graph of F is recursive by i).

∎

Exercise 46 Prove that statement i) van proposition 3.2.5 holds primitive-recursively uniform in indices, that is: there is a binary primitive recursive function T such that, whenever W_x is the complement of W_y then $\chi_{W_x} = \phi_{T(x,y)}$.

Exercise 47

i) Use the Recursion Theorem in order to show that for every total recursive function F there is a number n such that $W_{F(n)} = W_n$; deduce from this that there exists an n such that $W_n = \{n\}$ and that there is an n such that $W_n = \{m \mid m > n\}$.

ii) Prove, using the Recursion Theorem, that there is a total recursive function F with the properties $\forall n(F(n) < F(n+1))$ and $\forall n(W_{F(n)} = \{F(n+1)\})$.

Exercise 48 Prove:

i) If a total recursive function F is nondecreasing and has infinite range, then $\mathrm{rge}(F)$ is recursive.

ii) Every infinite r.e. set contains an infinite recursive subset.

Do these results hold primitive-recursively uniform in indices?

Propositie 3.2.5ii) shows that the r.e. are not closed under complements. Exercise 44 implies that they are neither closed under arbitrary intersections (for $\{e \mid \varphi_e \text{ is total}\} = \bigcap_{x\in\mathbb{N}}\{e \mid \exists y T(1, e, x, y)\}$) or unions (for $A = \bigcup_{x\in A}\{x\}$ for every set A, and every singleton set is of course r.e.). We do have the following:

Proposition 3.2.6

i) If R and S are r.e. then so are $R \cup S$ and $R \cap S$;

ii) if $R \subseteq \mathbb{N}^{k+1}$ is r.e. then so is

$$\{(\vec{x}, z) \in \mathbb{N}^{k+1} \mid \forall w < z R(\vec{x}, w)\}$$

iii) if $R \subseteq \mathbb{N}$ is r.e. then so is $\bigcup_{x\in R} W_x$.

These results are primitive-recursively uniform in indices (that is, e.g. for i): $W_x \cap W_y = W_{F(x,y)}$ and $W_x \cup W_y = W_{G(x,y)}$ for primitive recursive functions F and G.).

Exercise 49 Prove proposition 3.2.6. For ii): use coding of sequences in order to replace a quantifier combination $\forall w < z \exists y$ by a combination $\exists y \forall w < z$.

Exercise 50 [Reduction Theorem] Prove that for every couple of r.e. sets $X, Y \subseteq \mathbb{N}$ there exist r.e. sets X' en Y' such that the following hold:

i) $X' \subseteq X$ and $Y' \subseteq Y$;

ii) $X' \cap Y' = \emptyset$;

iii) $X' \cup Y' = X \cup Y$.

Hint: if $X = W_e$ and $Y = W_f$, let

$$X' = \{x \mid \exists y(T(1, e, x, y) \text{ and } \forall z < y \neg T(1, f, x, z))\}$$

Definition 3.2.7 *Two r.e. sets X, Y are called* recursively inseparable *if there is no recursive set R with $X \subseteq R$ and $R \cap Y = \emptyset$ (equivalently, if there is no total recursive function F such that $F(x) = 0$ for all $x \in X$ and $F(x) > 0$ for all $x \in Y$).*

Proposition 3.2.8 *Recursively inseparable r.e. sets exist.*

Proof. Define

$$\begin{array}{lcl} X & = & \{x \mid x{\cdot}x = 0\} \quad = \quad \{x \mid \exists y[T(1, x, x, y) \text{ and } U(y) = 0]\} \\ Y & = & \{x \mid x{\cdot}x = 1\} \end{array}$$

We shall show that there is a primitive recursive function F with the following property: als $X \subseteq W_e$, $Y \subseteq W_f$ and $W_e \cap W_f = \emptyset$, then $F(e, f) \notin W_e \cup W_f$ (Check that this implies the statement! Use proposition 3.2.5i)).
Let g be an index such that for all e, f, x:

$$g{\cdot}(e, f, x) \simeq \begin{cases} 1 & \text{if } \exists y[T(1, e, x, y) \text{ and } \forall w \leq y \neg T(1, f, x, w)] \\ 0 & \text{if } \exists y[T(1, f, x, y) \text{ and } \forall w < y \neg T(1, e, x, w)] \\ \text{undefined} & \text{otherwise} \end{cases}$$

Define $F(e, f) = S_1^2(g, e, f)$ and let e and f be such that $X \subseteq W_e$, $Y \subseteq W_f$ and $W_e \cap W_f = \emptyset$. Then we have:
if $S_1^2(g, e, f) \in W_e$ then

$$S_1^2(g, e, f){\cdot}S_1^2(g, e, f) = g{\cdot}(e, f, S_1^2(g, e, f)) = 1$$

hence $S_1^2(g, e, f) \in Y$, contradiction;
if $S_1^2(g, e, f) \in W_f$ then

$$S_1^2(g, e, f){\cdot}S_1^2(g, e, f) = 0$$

hence $S_1^2(g, e, f) \in X$; with a similar contradiction.
We conclude that $S_1^2(g, e, f) \notin W_e \cup W_f$, as desired. ∎

Exercise 51 [Extension Problem] Show that there exist partial recursive functions which cannot be extended to a total recursive function.

Exercise 52 An r.e. set X is called *creative via a partial recursive function F* if for every e the following holds:

$$\text{If } W_e \cap X = \emptyset \text{ then } F(e) \text{ is defined and } F(e) \notin X \cup W_e$$

Show that for the sets X en Y from the proof of proposition 3.2.8 there exist partial recursive functions F and G such that X and Y are creative via F and G, respectively.

3.2.1 The Kleene Tree

A *tree* is (for us, here) a set T of finite sequences of natural numbers which is closed under initial segments: if $(s_0, \ldots, s_n) \in T$ and $m \leq n$ then also $(s_0, \ldots, s_m) \in T$. We also consider the empty sequence $(-)$, which is an initial segment of any sequence.
A tree is *finitely branching* if for every sequence $(s_0, \ldots, s_{n-1}) \in T$ there are only finitely many numbers a such that $(s_0, \ldots, s_{n-1}, a) \in T$.
A *path through a tree T* is a total function F such that for all n, $(F(0), \ldots, F(n)) \in T$.
A tree is called *recursive* if the set of codes of elements of T (in the sense of coding of sequences) is recursive.
In the context of arbitrary (not necessarily recursive) trees we have the following classical fact:

Proposition 3.2.9 (König's Lemma) *Let T be a finitely branching tree. If T is infinite, then there is a path through T.*

Proof. If s is an element of T, by T_s we denote the set of those elements of T which are either an initial segment of s or have s as initial segment. Clearly, T_s is a tree.
Assume T is finitely branching and infinite. Then T is nonempty so the empty sequence is in T; since T is finitely branching there are finitely many numbers a such that $(a) \in T$; say $a_0, \ldots, a_k$. Then

$$T = \bigcup_{i=0}^{k} T_{(a_i)}$$

which is a finite union of sets; since T is infinite we can find an i such that $T_{(a_i)}$ is infinite; let $F(0) = a_i$ for the least such i.
Inductively, we have defined $F(0), \ldots, F(n)$ in such a way that $T_{(F(0),\ldots,F(n))}$ is infinite. This tree is also finitely branching, so again, there are finitely many $a_0, \ldots, a_k$ such that

$$T_{(F(0),\ldots,F(n))} = \bigcup_{i=0}^{k} T_{(F(0),\ldots,F(n),a_i)}$$

and again we can pick $F(n+1) = a_i$ such that $T_{(F(0),\ldots,F(n),a_i)}$ is infinite.
You can check that the function F is well-defined and a path through T. ∎

The situation is different when we consider recursive trees and recursive paths:

Theorem 3.2.10 (Kleene Tree) *There is an infinite, finitely branching, recursive tree through which there is no recursive path.*

Proof. Let T be the set of those sequences $(a_0, \ldots, a_{n-1})$ which satisfy:

for all $i < n$, $a_i \leq 1$
for all $i, k < n$, if $T(1, i, i, k)$ and $U(k) \leq 1$, then $a_i = U(k)$

Convince yourself that T is a tree, and that it is recursive. By the first requirement on elements of T, T is finitely branching. In order to construct an element of T of length n, simply run through all $i, k < n$ and check whether $T(i, 1, i, k)$ and $U(k) \leq 1$. If so, put $a_i = U(k)$. This gives a finite number of constraints on the sequence we wish to construct; if there is no such constraint involving the number i, simply put $a_i = 0$. So we get a legitimate element of T of length n; hence T is infinite.
Now suppose F is a recursive path through T. Suppose $x \cdot x = 0$. Let k be such that $T(1, x, x, k)$. Let $n > \max(x, k)$. Then since $(F(0), \ldots, F(n)) \in T$ we must have $F(x) = 0$. Similarly, when $x \cdot x = 1$ we must have $F(x) = 1$. Also, F takes values in $\{0, 1\}$. We conclude that F is a recursive separation of the sets X and Y in the proof of proposition 3.2.8, contradicting that proposition. ∎

3.3 Extensional r.e. sets and effective operations

In this section you'll find three important theorems on extensional r.e. sets of indices of partial recursive functions and indices of r.e. sets, and "effective operations" on these things. Let us clarify what we mean by this.

Definition 3.3.1 A subset A of $\mathbb{N}$ is called *extensional for indices of r.e. sets* if whenever $e \in A$ and $W_e = W_f$, then $f \in A$.

Write $\mathcal{PR}$ for the set of all partial recursive functions, $\mathcal{R}$ for the set of total recursive functions, and $\mathcal{RE}$ for the set of r.e. subsets of $\mathbb{N}$. A function $H : \mathcal{PR} \to \mathcal{PR}$, $\mathcal{R} \to \mathcal{R}$ or $\mathcal{RE} \to \mathcal{RE}$ is called an *effective operation* if there is a partial recursive function F such that:

if $H : \mathcal{PR} \to \mathcal{PR}$ then F is total, and for all e, $H(\phi_e) = \phi_{F(e)}$

if $H : \mathcal{R} \to \mathcal{R}$ then $\{e \,|\, \phi_e \text{ is total}\} \subseteq \mathrm{dom}(F)$ and for all e, $H(\phi_e) = \phi_{F(e)}$

if $H : \mathcal{RE} \to \mathcal{RE}$ then F is total and for all e: $H(W_e) = W_{F(e)}$

In a similar way, one can define effective operations $\mathcal{RE} \to \mathcal{PR}$, $\mathcal{R} \to \mathbb{N}$ etc.
In this section we shall show that there exist very natural topologies on the sets $\mathcal{PR}$, $\mathcal{R}$ and $\mathcal{RE}$, such that the following statements are true:

every r.e. extensional set for indices of partial recursive functions is open in $\mathcal{PR}$

every r.e. extensional set for indices of r.e. sets is open in $\mathcal{RE}$

every effective operation is *continuous*

The topology we take on $\mathcal{R}$ is straightforward. Give $\mathbb{N}$ the discrete topology, and then the set of functions $\mathbb{N}^{\mathbb{N}}$, which is the same as the infinite profuct $\prod_{n\in\mathbb{N}} \mathbb{N}$, the product topology. A basis for this topology is given by the sets

$$\mathcal{U}_s \;=\; \{f : \mathbb{N} \to \mathbb{N} \,|\, f(0) = s_0, \ldots, f(n-1) = s_{n-1}\}$$

for every finite sequence $s = (s_0, \ldots, s_{n-1})$ of numbers.

The set $\mathcal{R}$, as subset of $\mathbb{N}^{\mathbb{N}}$, now gets the subspace topology, which has as basis the sets $\mathcal{U}_s \cap \mathcal{R}$.

For $\mathcal{RE}$, we do something more subtle. When we have numbers n and e, the problem "is n an element of W_e?" is, as we know, in general undecidable. We do have a program that, on inputs n and e, outputs 0 if $n \in W_e$ (the program searches for the first y such that $T(1, e, n, y)$ and outputs 0 if such y is found); but for $n \notin W_e$ the program will never tell us anything. We therefore choose a topology on $\mathcal{RE}$ which mirrors this fact.

Identify $\mathcal{P}(\mathbb{N})$, the powerset of $\mathbb{N}$, with the set of functions $\mathbb{N} \to \{0, 1\}$ via characteristic functions. This is also a product: $\prod_{n\in\mathbb{N}}\{0, 1\}$, but this time we do *not* give $\{0, 1\}$ the discrete topology; instead, we give it the *Sierpinski topology* which has 3 open sets: $\emptyset$, $\{0\}$ and $\{0, 1\}$. Then, endow $\{0, 1\}^{\mathbb{N}}$ with the product topology. This results in a topology on $\mathcal{P}(\mathbb{N})$ for which the following sets form a basis:

$$\mathcal{U}_A \;=\; \{X \subseteq \mathbb{N} \,|\, A \subseteq X\}$$

where $A \subset \mathbb{N}$ is *finite*.

Finally, $\mathcal{RE}$, as subset of $\mathcal{P}(\mathbb{N})$, gets the subspace topology.

Exercise 53 Let $\mathcal{V} \subseteq \mathcal{RE}$. Prove that $\mathcal{V}$ is open in $\mathcal{RE}$ if and only if the following two statements hold:

i) $\mathcal{V}$ is *upwards closed*: if $X \in \mathcal{V}$, $Y \in \mathcal{RE}$ and $X \subseteq Y$, then $Y \in \mathcal{V}$

ii) whenever $X \in \mathcal{V}$, there is a finite subset $A \subseteq X$ such that $A \in \mathcal{V}$.

For $\mathcal{PR}$ we note that this set can be seen as a subset of $\mathcal{RE}$ via the map which sends a partial recursive function to its graph (which as an r.e. set). Therefore, we give $\mathcal{PR}$ the subspace topology of $\mathcal{RE}$. A basis for the topology on $\mathcal{PR}$ is given by sets

$$\mathcal{U}_s \;=\; \{\varphi \text{ partial recursive} \,|\, \forall i < \mathrm{lh}(s)\,(\varphi(j_1(s_i)) = j_2(s_i))\}$$

where s is such that for all $i, j < \mathrm{lh}(s)$, if $j_1(s_i) = j_1(s_j)$ then $s_i = s_j$.

Again, a set of $\mathcal{V}$ of partial recursive functions is open if and only if $\mathcal{V}$ is upwards closed and satisfies the condition that whenever a function F is in $\mathcal{V}$, there is a finite subfunction G of F which is in $\mathcal{V}$.

Proposition 3.3.2

a) *A function $F : \mathcal{R} \to \mathcal{R}$ is continuous if and only if for every total recursive function f and every finite sequence s such that $F(f) \in \mathcal{U}_s$, there is a finite sequence t such that $f \in \mathcal{U}_t$ and F maps $\mathcal{U}_t$ into $\mathcal{U}_s$.*

b) *A function $F : \mathcal{RE} \to \mathcal{RE}$ or $\mathcal{PR} \to \mathcal{PR}$ is continuous if and only if the following two confitions hold:*

- *i)* *F is* monotone*: whenever $S \subseteq T$, $F(S) \subseteq F(T)$*
- *ii)* *F is* compact*: whenever $A \subseteq F(S)$ is finite, there is a finite subset B of S such that $A \subseteq F(B)$*

Exercise 54 Prove proposition 3.3.2.

3.3.1 Theorems of Myhill-Shepherdson, Rice-Shapiro and Kreisel-Lacombe-Shoenfield

In the following theorem, parts 1 and 2 are known as the *Myhill-Shepherdson Theorem*, and 3 and 4 form the *Rice-Shapiro Theorem*

Theorem 3.3.3 (Myhill-Shepherdson; Rice-Shapiro)

1) *Let R be r.e. and extensional for indices of partial recursive functions. Then the set*

$$\{\phi_e \,|\, e \in R\}$$

is open in $\mathcal{PR}$

2) *Let $F : \mathcal{PR} \to \mathcal{PR}$ be an effective operation. Then F is continuous*

3) *Let R be r.e. and extensional for indices of r.e. sets. Then the set*

$$\{W_e \,|\, e \in R\}$$

is open in $\mathcal{RE}$

4) *Let $F : \mathcal{RE} \to \mathcal{RE}$ be an effective operation. Then F is continuous*

Proof. 1). We have to show the following two things:

a) If $e \in R$ and $\phi_e \subseteq \phi_f$ then $f \in R$

b) If $e \in R$ then there is a number e' such that $\phi_{e'}$ is finite, $\phi_{e'} \subseteq \phi_e$, and $e' \in R$

Let h be an index for the r.e. set R.

For a), use the Recursion Theorem to pick an index g such that for all e, f, x the following holds:

$$g{\cdot}(e,f,x) \;\simeq\; \begin{array}{c} U(j_2(\mu z.[j_1 z = 0 \text{ and } T(1,e,x,j_2 z)] \text{ or} \\ [T(1,h,S_1^2(g,e,f),j_1 z) \text{ and } T(1,f,x,j_2 z)])) \end{array}$$

Now suppose $e \in R$. First we see that $S_1^2(g,e,f) \in R$. For if not, no z will ever satisfy the second part of the disjunction in the definition of $g{\cdot}(e,f,x)$. That means that for all x,

$$S_1^2(g,e,f){\cdot}x \simeq U(j_2(\mu z.[j_1 z = 0 \text{ and } T(1,e,x,j_2 z)])) \simeq e{\cdot}x$$

so $\phi_{S_1^2(g,e,f)} = \phi_e$. But $e \in R$, and R is extensional. This is a contradiction. Therefore, $S_1^2(g,e,f) \in R$.

Now suppose furthermore that $\phi_e \subseteq \phi_f$. We observe that $g{\cdot}(e,f,x) \simeq f{\cdot}x$ always (check!), so $\phi_{S_1^2(g,e,f)} = \phi_f$. Since $S_1^2(g,e,f) \in R$ and R is extensional, we have $f \in R$, as desired.
For b), use the Recursion Theorem to pick an index g such that for all e, x:

$$g{\cdot}(e,x) \;\simeq\; \left\{ \begin{array}{ll} e{\cdot}x & \text{if for no } k \le x,\ T(1,h,S_1^1(g,e),k) \\ \text{undefined} & \text{otherwise} \end{array} \right.$$

Assume $e \in R$. Again, we see that $S_1^1(g,e)$ must be an element of R, for otherwise we have

$$S_1^1(g,e){\cdot}x \simeq g{\cdot}(e,x) \simeq e{\cdot}x$$

so $\phi_{S_1^1(g,e)} = \phi_e$, and we get a contradiction with the extensionality of R.

Now we know that $S_1^1(g,e) \in R$, let k be the least such that $T(1,h,S_1^1(g,e),k)$. Then from the definition we see that $g{\cdot}(e,x)$ is undefined for $x \ge k$, so $S_1^1(g,e)$ is the index of a finite function. Moreover, the inclusion $\phi_{S_1^1(g,e)} \subseteq \phi_e$ is immediate from the definition.

2). We show directly that the preimage $F^{-1}(\mathcal{U}_s)$ is open in $\mathcal{PR}$, where $\mathcal{U}_s$ is a typical basis element for the topology on $\mathcal{PR}$:

$$\mathcal{U}_s \;=\; \{\varphi \text{ partial recursive} \,|\, \forall i < \mathrm{lh}(s)\,(\varphi(j_1(s_i)) = j_2(s_i))\}$$

where s is such that for all $i, j < \mathrm{lh}(s)$, if $j_1(s_i) = j_1(s_j)$ then $s_i = s_j$.

Let f be an index such that $F(\phi_e) = \phi_{f{\cdot}e}$ for all e. Then we have that $\phi_e \in F^{-1}(\mathcal{U}_s)$ if and only if

$$\forall i < \mathrm{lh}(s)\,((f{\cdot}e){\cdot}j_1(s_i) \text{ is defined and equal to } j_2(s_i))$$

We see that $\{e \,|\, \phi_e \in F^{-1}(\mathcal{U}_s)\}$ is r.e. It is also clearly extensional, because F is an effective operation. So the openness of $F^{-1}(\mathcal{U}_s)$ follows from part 1.

3). Suppose R is r.e. and extensional for indices of r.e. sets. Then R is also extensional for indices of partial recursive functions, because $\phi_e = \phi_f$ implies $W_e = W_f$. Hence, by 1), the set

$$P \;=\; \{\phi_e \,|\, e \in R\}$$

is open in $\mathcal{PR}$. Therefore, if $e \in R$ there is some $e' \in R$ such that $\phi_{e'} \subseteq \phi_e$ and $\phi_{e'}$ is finite. Then also $W_{e'}$ is finite, and $W_{e'} \subseteq W_e$.

Furthermore, if $e \in R$ and $W_e \subseteq W_f$, we have the partial recursive functions

$$\varphi^e(x) \simeq \left\{ \begin{array}{rl} 0 & \text{if } x \in W_e \\ \text{undefined} & \text{otherwise} \end{array} \right. \quad \text{and } \varphi^f(x) \simeq \left\{ \begin{array}{rl} 0 & \text{if } x \in W_f \\ \text{undefined} & \text{otherwise} \end{array} \right.$$

Any index for φ^e belongs to R since $\mathrm{dom}(\varphi^e) = W_e$; so by openness of the set P, any index for φ^f belongs to R. Hence $f \in R$.

4). We have continuous maps $I : \mathcal{RE} \to \mathcal{PR}$ and $J : \mathcal{PR} \to \mathcal{RE}$ defined by: $I(W_e)$ is the function φ^e from the proof of 3), and $J(\varphi) = \mathrm{dom}(\varphi)$. An index for φ^e can be found primitive-recursively in e: if g is an index such that

$$g{\cdot}(e,x) \simeq \begin{cases} 0 & \text{if } x \in W_e \\ \text{undefined} & \text{otherwise} \end{cases}$$

then $S^1_1(g,e)$ is an index for φ^e. Also, $J(\phi_e) = W_e$. It follows from this, that when $F : \mathcal{RE} \to \mathcal{RE}$ is an effective operation, then the composition $IFJ : \mathcal{PR} \to \mathcal{PR}$ is also an effective operation, and hence continuous by 2). Because the functions I and J are continuous, the composition $J(IFJ)I$ is a continuous map $\mathcal{RE} \to \mathcal{RE}$. But this map is just F, because the composition JI is the identity map on $\mathcal{RE}$. ∎

Note that we have actually proven a primitive-recursively uniformity result: there is a primitive recursive function $S(h,e)$ such that whenever $R = W_h$ is extensional and $e \in R$, then $S(h,e)$ is an index for a finite set (or function) which is in R and contained in W_e (ϕ_e).

Exercise 55 Conclude from Theorem 3.3.3 that there cannot exist a total recursive function F which is such that for all e: ϕ_e is constant on its domain if and only if $F(e) \in \mathcal{K}$.

Exercise 56 Conclude from Theorem 3.3.3 that there cannot exist a total recursive function F which is such that for all e: W_e is a singleton set precisely when $F(e) \in \mathcal{K}$.

Exercise 57 Conclude from Theorem 3.3.3 that neither $\{x \mid W_x \text{ is finite}\}$ nor $\{x \mid W_x \text{ is infinite}\}$ are r.e.

The situation for $\mathcal{R}$ is a bit more subtle. Recall that the basic opens are of the form $\mathcal{U}_s$ zijn, met

$$\mathcal{U}_s = \{f \in \mathcal{R} \mid \forall i < \mathrm{lh}(s)\, f(i) = (s)_i\}$$

Call a set X *extensioneel for indices of total functions*, if for every pair e, e' of indices of total functions we have:

$$\text{If } \phi_e = \phi_{e'} \text{ then } (e \in X \Leftrightarrow e' \in X)$$

The result on r.e. extensional sets being open, *fails* in this context. First of all we have:

Exercise 58 Show: if X is an r.e. set of indices of total recursive functions and X is extensional for indices of total recursive functions, then $X = \emptyset$.

But we can modify the question: suppose we do *not* require that X consist only of indices for total functions. Do we have that if X is r.e. and extensional for indices of total functions, the set

$$\{\phi_e \mid \phi_e \text{ is total and } e \in X\}$$

is open in $\mathcal{R}$? The answer is *no*, as the following proposition shows.

Proposition 3.3.4 (Friedberg, 1958) *There exists an r.e. set X which is extensional for indices of total functions, such that $\{\phi_e \mid \phi_e$ is total and $e \in X\}$ is not open in $\mathcal{R}$.*

Proof. Define sets Y and Z by

$$\begin{aligned} Y &= \{e \mid \forall x \leq e.e{\cdot}x = 0\} \\ Z &= \{e \mid \exists z[e{\cdot}z > 0 \text{ and } \forall y < z.e{\cdot}y = 0 \text{ and} \\ &\quad \exists e' < z \forall u \leq z.e'{\cdot}u = e{\cdot}u]\} \end{aligned}$$

Let $X = Y \cup Z$. Since both Y and Z are r.e., so is X. Also, note that Z is extensional for indices of total functions. In order to see that X is extensional for indices of total functions, suppose $\phi_e = \phi_{e'}$, ϕ_e and $\phi_{e'}$ are total, and $e \in X$.

If $\phi_{e'} = \lambda x.0$ then $e' \in Y$, hence $e' \in X$. If not, let y' the minimal x for which $\phi_{e'}(x) > 0$. We distinguish two cases:

a) $e \in Y$. Then since $\phi_e = \phi_{e'}$, we have $\forall x \leq e.e' \cdot x = 0$, so $e < y'$.
If $y' \leq e'$ then $e' \in Z$ (switch the roles of e and e' in the definition of Z, and take y' for z).
If $e' < y'$, then $e' \in Y$. We conclude that if $e \in Y$ then always $e' \in X$.

b) $e \in Z$. Then $e' \in Z$ by extensionality of Z, hence also $e' \in X$.

However, the set $T = \{\phi_e \in \mathcal{R} \mid e \in X\}$ is not open in $\mathcal{R}$: because $\lambda x.0 \in T$ we would have, if T were open, an s such that $\forall i < \mathrm{lh}(s).(s)_i = 0 \wedge \mathcal{U}_s \subseteq T$; choose such an s and let $n = \mathrm{lh}(s)$. Now let g be an index such that for all x:

$$g \cdot x \simeq \begin{cases} 0 & \text{if } x < n \\ k & \text{otherwise} \end{cases}$$

where $k \neq 0$ is determined in such a way that no partial recursive function with index $< n$ has the same values as ϕ_g on the set $\{0, \ldots, n\}$. We have: $\phi_g \in \mathcal{U}_s$, but $n \leq g$ (check!), and $g \notin X$. ∎

Exercise 59 Show that if X is r.e. and extensional for indices of partial recursive functions, then the set

$$\{\phi_e \mid \phi_e \text{ is total and } e \in X\}$$

is open in $\mathcal{R}$.

We do have the continuity result, for which the most important building block is the theorem of Kreisel-Lacombe-Shoenfield (independently also proved by the Russian Čeitin).

Theorem 3.3.5 (Kreisel, Lacombe, Shoenfield (1957)) *There is a partial recursive function K with the following property: for every effective operation $F : \mathcal{R} \to \mathbb{N}$ (effective in the sense that $F(\phi_e) = f \cdot e$ for a certain index f) and every f satisfying*

$$\{e \mid \phi_e \text{ is total}\} \subseteq \mathrm{dom}(\phi_f) \text{ and } \forall e(\text{if } \phi_e \text{ is total then } f \cdot e = F(\phi_e))$$

and for every e such that ϕ_e is total, $K(f, e)$ is defined, and the following holds:

1. $\phi_e \in \mathcal{U}_{K(f,e)}$
2. $\forall e'(\text{if } \phi_{e'} \text{ is total and } \phi_{e'} \in \mathcal{U}_{K(f,e)} \text{ then } F(\phi_e) = F(\phi_{e'}))$

Proof. Let $s^\circ = \lambda x.(s)_x$ (Recall that by convention, $(s)_x = 0$ for $x \geq \mathrm{lh}(s)$, so s° is a total function); let G be primitive recursive such that $s^\circ = \phi_{G(s)}$. We define a partial recursive function of 4 variables as follows:

$$\eta(f, y, z, x) \simeq \begin{cases} \text{undefined} & \text{if} & f \cdot y \text{ is undefined} & (1) \\ y \cdot x & \text{if} & \neg\exists w \leq x(T(1, f, z, w) \text{ and } U(w) = f \cdot y) & (2) \\ s^\circ(x) & \text{if} & w \leq x \text{ is minimal such that } T(1, f, z, w) \text{ and } U(w) = f \cdot y, & \\ & & \text{and } \langle s, t \rangle \text{ is minimal such that } \mathrm{lh}(s) > w \text{ and} & \\ & & \text{and } \forall i < w \exists v < t(T(1, y, i, v) \text{ and } U(v) = (s)_i) \text{ and} & (3) \\ & & \text{and } \exists v < t(T(1, f, G(s), v) \text{ and } U(v) \neq f \cdot y) & \\ \text{undefined} & \text{if} & \text{in (3) such a } \langle s, t \rangle \text{ does not exist} & \end{cases}$$

We apply the Recursion Theorem to find an index z such that for all f, y, x:

$$z \cdot (f, y, x) \simeq \eta(f, y, S_1^2(z, f, y), x)$$

Now suppose that F is an effective operation, and f an index such that $F(\varphi_e) = f \cdot e$ for every index e of a total recursive function, and assume ϕ_y is total. Then $f \cdot y$ is defined. Moreover we have:

i) There exists a w such that $T(1, f, S_1^2(z, f, y), w)$ and $U(w) = f \cdot y$. For if not, then $S_1^2(z, f, y) \cdot x$ equals $y \cdot x$ for all x (this follows from the definition of η, for $S_1^2(z, f, y) \cdot x \simeq \eta(f, y, S_1^2(z, f, y), x)$), i.e. $\phi_{S_1^2(z,f,y)} = \phi_y$, but then we should have $f \cdot S_1^2(z, f, y) = f \cdot y$, by the assumption on f.

ii) Let w be minimal such that $T(1, f, S^2_1(z, f, y), w)$. Then there cannot exist an s such that

$$w \leq \mathrm{lh}(s) \text{ and } \forall i < w.y{\cdot}i = (s)_i \text{ and } f{\cdot}G(s) \neq f{\cdot}y$$

For suppose such s existed. Then there would be $\langle s, t\rangle$ as in (3) of the definition of η, which would imply that $\phi_{S^2_1(z,f,y)} = \phi_{G(s)} = s^\circ$ so $f{\cdot}S^2_1(z, f, y) = f{\cdot}G(s) \neq f{\cdot}y$, which contradicts i).

iii) With w as in ii) we have: if $\phi_{y'}$ is total $\forall i < w.y'{\cdot}i = y{\cdot}i$, then $f{\cdot}y = f{\cdot}y'$; for if w' satisfies $T(1, f, S^2_1(z, f, y'), w')$ and w'' is the maximum of w and w'we have:

$$f{\cdot}y' = f{\cdot}G(\langle y'{\cdot}0, \ldots, y'{\cdot}(w' - 1)\rangle)$$

because of ii) with y' substituted for y; this also equals $f{\cdot}G(\langle y'{\cdot}0, \ldots, y'{\cdot}(w'' - 1)\rangle) = f{\cdot}y$, for $\langle y'{\cdot}0, \ldots, y'{\cdot}(w'' - 1)\rangle$ is an extension of the sequence $\langle y{\cdot}0, \ldots, y{\cdot}(w - 1)\rangle$.

So if we put:

$$K(f, y) \simeq \langle y{\cdot}0, \ldots, y{\cdot}(v - 1)\rangle$$

where $v = \mu w.T(1, f, S^2_1(z, f, y), w)$, then K has the stated properties of the theorem. Check yourself that K is partial recursive. ∎

Corollary 3.3.6 *Let $F : \mathcal{R} \to \mathcal{R}$ be an effective operation. Then F is continuous.*

Proof. This is immediate from Theorem 3.3.5: for each n, the function which sends f to $F(f)(n)$ is an effective operation $\mathcal{R} \to \mathbb{N}$, hence continuous. Therefore, by the defining property of the product topology, F is continuous as map $\mathcal{R} \to \mathbb{N}^{\mathbb{N}}$; and since it lands in $\mathcal{R}$ and $\mathcal{R}$ has the subspace topology, F is continuous as map $\mathcal{R} \to \mathcal{R}$. ∎

However, we can make a stronger statement. Call a map $F : \mathcal{R} \to \mathcal{R}$ *effectively continuous* if there is a partial recursive function φ such that for every pair (e, n) with ϕ_e total, $\varphi(e, n)$ is defined and the following holds:

$$F[\mathcal{U}_{\langle e{\cdot}0,\ldots,e{\cdot}\varphi(e,n)\rangle} \subseteq \mathcal{U}_{\langle F(\phi_e)(0),\ldots,F(\phi_e)(n)\rangle}$$

Corollary 3.3.7 *Let $F : \mathcal{R} \to \mathcal{R}$ be an effective operation. Then F is effectively continuous.*

Proof. We must show that for every $f \in \mathbb{R}$ and every s such that $F(f) \in \mathcal{U}_s$, there is a v such that $f \in \mathcal{U}_v$ and $F[\mathcal{U}_v] \subseteq \mathcal{U}_s$.
Because F is an effective operation, there is a partial recursive function ψ such that for all i and all indices of total recursive functions e:

$$\psi(e, i) = F(\varphi_e)(i)$$

Let G be primitive recursive, such that $G(i)$ is an index for $\lambda e.\psi(e, i)$; we have:

$$\phi_e = \phi_{e'} \Rightarrow G(i){\cdot}e = G(i){\cdot}e'$$

for all i and all indices for total recursive functions e, e'; therefore we can apply Theorem 3.3.5 to $G(i)$. Let $f \in \mathbb{R}$, $f = \phi_y$. Let

$$w = \max\{\mathrm{lh}(K(G(i), y)) \mid 0 \leq i < \mathrm{lh}(u)\}$$

Then the following holds:

$$\text{If } g \in \mathcal{U}_{\langle y{\cdot}0,\ldots,y{\cdot}(w-1)\rangle} \text{ then } F(g) \in \mathcal{U}_s$$

(check!) ∎

Remark 3.3.8 The number $w = w_{y,s} = \max\{\mathrm{lh}(K(G(i), y)) \mid 0 \leq i < \mathrm{lh}(s)\}$ is sometimes called a *modulus of continuity* of F at ϕ_y. However we should keep in mind that it crucially depends on the *index* y; that is, in general there is no partial recursive function N which satisfies: $N(y, u)$ is a modulus van continuity, and if $\phi_y = \phi_{y'}$ then $N(y, s) = N(y', s)$.

Exercise 60 Show that there does not exist a subset $A \subseteq \mathbb{N}$ and an effective operation $F : \mathcal{R} \to \mathbb{N}$ which satisfies for all indices for total recursive functions e:

$$0 \in \mathrm{rge}(\phi_e) \text{ if and only if } F(e) \in A$$

3.4 Strict r.e. sets in Mathematics and Logic

In this section we discuss a number of problems in Mathematics and Logic, the unsolvability of which has been demonstrated using techniques from Recursion Theory. In every example there is a suitable *coding* of the problem in terms of natural numbers.

3.4.1 Hilbert's Tenth Problem

As mentioned in the Introduction, in 1900 David Hilbert presented a list with problems for the 20th century. The tenth problem was about *Diophantine equations*, that is: equations of the form $P(X_1, \ldots, X_n) = 0$, waar $P \in \mathbb{Z}[X_1, \ldots, X_n]$ is a polynomial with integer coefficients. The problem reads:

> 10. Given a Diophantine equation in an arbitrary number of variables and with integer coefficients; design a procedure by which it can be deternmined in a finite number of steps whether the equation has a solution in the integers.

Now every polynomial in n variables can of course be coded as a sequence of $n+1$-tuples (for coefficients and powers), for example: give $4X_1^4X_2 + 5X_2^2X_3 + X_1X_3$ code

$$\langle\langle 4, 4, 1, 0\rangle, \langle 5, 0, 2, 1\rangle, \langle 1, 1, 0, 1\rangle\rangle$$

in such a way that if $\ulcorner P \urcorner$ denotes the code of the polynomial P, the function

$$\lambda x_1 \cdots x_n . P(x_1, \ldots, x_n)$$

is primitive recursive in $\ulcorner P \urcorner$ and $x_1, \ldots, x_n$. The set

$$\{u \mid u = \ulcorner P \urcorner \text{ and } \exists \vec{x} . P(\vec{x}) = 0\}$$

is then obviously r.e. A solution to Hilbert's Tenth Problem presupposes that the set of all u such that u codes a polynomial P for which the equation $P(\vec{x}) = 0$ has integer solutions, is recursive.

In 1970, after much preparatory work by Davis, Julia Robinson, and Putnam, the Russian Yuri Matyasevich proved that for every e and n there is a polynomial P in at least $n+1$ variabbles and with integer coefficients, such that

$$W_e^n = \{(x_1, \ldots, x_n) \mid \exists a_{n+1} \cdots a_{n+k} P(x_1, \ldots, x_n, a_{n+1}, \ldots, a_{n+k}) = 0\}$$

It now follows that

> the set W_e^n is nonempty, precisely when the corresponding polynomial $P(X_1, \ldots, X_{n+k})$ has integer zeroes

and therefore, a solution of Hilbert's Tenth Problem in full generality would yield a decision procedure for the problem: is $W_e^n = \emptyset$? But we have seen, that this problem is unsolvable.

A proof of Matyasevich's Theorem can be found in the very accessible book [24]; there is also a proof in [37].

Interestingly, this theorem initiated a development that is still going on today. Number-theorists got interested in the question: what if one replaces (in the statement of Hilbert's problem) $\mathbb{Z}$ by $\mathbb{Q}$? Or, if one replaces $\mathbb{Z}$ by the ring of integers of a number field? These problems are more complicated than the original one, and involve study of elliptic curves and algebraic geometry. For a survey of results and open questions, see [28].

3.4.2 Word Problems: Groups

Word problems arise with every type of mathematical structure which is defined by generators and relations. As an example, I sketch the word problem for groups.

Let $A = \{a_1, \ldots, a_n\}$ be a finite set. Let $f : A \to B$ be a bijection, where B is disjoint from A. We shall write A^{-1} for B and a^{-1} for $f(a)$.

A *word* on A is a sequence of elements of $A \cup A^{-1}$ in which never two elements a_i, a_i^{-1} (or a_i^{-1}, a_i appear consecutively. We also consider the empty sequence as a word. Let $[A]$ be the collection of words on A

We can concatenate two words: if $\sigma = b_1 \cdots b_m$ and $\tau = c_1 \cdots c_l$ are words, let $\sigma * \tau = b_1 \cdots b_m c_1 \cdots c_l$. This need not be a word, but by successively removing adjacent elements a_i, a_i^{-1} or a_i^{-1}, a_i from it, we do get a word, which we denote by $\sigma\tau$.

Proposition 3.4.1 *With the operation $\sigma, \tau \mapsto \sigma\tau$, $[A]$ is a group.*

Exercise 61 Prove proposition 3.4.1.

$[A]$ is called the *free group* generated by the set A. It has the following universal property: for every group G and every function $f : A \to G$ there is a unique group homomorphism $F_f : [A] \to G$ which sends each one-element sequence (a) to $f(a)$.

A *relation* on A is an expression $X = Y$, with $X, Y \in [A]$. If R is a set of relations on A, we denote by $[A; R]$ the quotient group $[A]/K$ where K is the least normal subgroup of $[A]$ which contains all elements $X^{-1}Y$ for X, Y such that $X = Y$ is in R.

A *finite presentation* of a group G is a pair (A, R) with A a finite set and R a finite set of relations on A, which satisfies $G \cong [A; R]$. We say that G is *finitely presented*, if it has a finite presentation.

Let $G = [A; R]$ and $\phi_R : [A] \to G$ be the quotient map. We say that the *word problem for G is solvable*, if there is an algorithm which determines for each word X on A whether $\phi(X)$ is the unit element of G.

At first glance, the solvability of the word problem for G appears to depend on the presentation; however, it is not too hard to see that in fact it is independent of it.

Novikov's Theorem (1955) states that there exists a finitely presented group with an unsolvable word problem.

For a proof, see the appendix of [34], or the book [23].

3.4.3 Theorems by Church and Trakhtenbrot

In section 2.1.2 we mentioned that all primitive recursive functions are provably total in $I\Sigma_1$. In particular, there is a formula $\varphi^T(x, y, z, w)$ of the language of ordered rings, which is such that for every quadruple k, l, m, n of elements of $\mathbb{N}$ we have:

$$I\Sigma_1 \models \varphi^T(\overline{k}, \overline{l}, \overline{m}, \overline{n}) \text{ if and only if } T(k, l, m, n)$$

where T denotes the Kleene T-predicate. Let $\psi(x, y)$ be the formula $\varphi^T(\overline{1}, x, x, y)$.

Exercise 62 Prove that for every natural number n the following holds:

$$I\Sigma_1 \models \exists y \psi(\overline{n}, y) \text{ if and only if } n \in \mathcal{K}$$

It can be shown that $I\Sigma_1$ is *finitely axiomatized*: it is equivalent to a theory in the same language, with only finitely many axioms. Denote the conjunction of these axioms by $\mathcal{A}$.
Hilbert's *Entscheidungsproblem* called for an algorithm which would determine, for an arbitrary formula in first-order logic, whether or not this formula is valid. *Church's Theorem* states that it is impossible to have such an algorithm, and this is why: from the remarks and the exercise above it folows that for all n the following statements are equivalent:

a) $n \in \mathcal{K}$

b) $I\Sigma_1 \models \exists y \psi(\overline{n}, y)$

c) the formula $\mathcal{A} \to \exists y \psi(\overline{n}, y)$ is valid

Hence, an algorithm as asked for by Hilbert would solve the problem $n \in \mathcal{K}?$, which we know is unsolvable.

A stronger statement is possible. It is possible to code terms and formulas in first-order logic (let us assume countably infinite sets of function symbols, of all arities, relation symbols of all arities, and constants) as numbers in such a way that elementary properties (such as for example: n is the code of a variable v and m is the code of a formula φ, and v occurs freely in φ) are given by primitive recursive relations on the codes. It can then be shown that the set of those numbers n which are codes of provable sentences, is r.e. Let us call this set $\mathcal{V}$ (of "valid sentences").

One can also code finite structures for finite languages: a finite structure consists of a number n and for every k-ary function symbol of the language, a function $\{0, \ldots, n\}^k \to \{0, \ldots, n\}$ etc. This can be done in such a way that the set P of those pairs (n, m) where n codes a sentence ψ in first-order logic, m codes a finite structure M for the language of ψ, and ψ is *false* in M, is recursive. Then the set $\mathcal{FR}$ of "finitely refutable" sentences is the set of those numbers n for which there exists a number m such that $(n, m) \in P$, is also r.e.

Trakhtenbrot's Theorem states that $\mathcal{V}$ and $\mathcal{FR}$ are recursively inseparable r.e. sets (we shall see later in this course that this is stronger than just "not recursive"). For more information: see [37].

Chapter 4

Reduction and Classification

This chapter is about *definable* subsets of $\mathbb{N}^k$. A subset $A \subseteq \mathbb{N}^k$ is definable if the assertion "$\vec{x} \in A$" is equivalent to a statement that can be written down using only primitive recursive relations, propositional logical symbols as $\wedge$, $\rightarrow$ etc., and quantifiers $\forall x$, $\exists x$. Equivalently, the class of definable sets is the least class $\mathcal{D}$ such that:

Every primitive recursive subset of $\mathbb{N}^k$ is in $\mathcal{D}$;

if $D \subseteq \mathbb{N}^k$ is in $\mathcal{D}$, then $\mathbb{N}^k - D$ is in $\mathcal{D}$;

if $D_1, D_2 \subseteq \mathbb{N}^k$ are in $\mathcal{D}$ then $D_1 \cap D_2$ is in $\mathcal{D}$;

if $D \subseteq \mathbb{N}^{k+1}$ is in $\mathcal{D}$ and $\pi : \mathbb{N}^{k+1} \rightarrow \mathbb{N}^k$ is the projection on the first k coordinates, then $\pi[D]$ is in $\mathcal{D}$.

Clearly, every r.e. set is definable, since it is the projection of a primitive recursive set (3.2.2). The definable sets can be classified according to the complexity of the simplest possible definition. We get a layered structure which is called the *arithmetical hierarchy*. An important instrument for determining the precise position of a set within the arithmetical hierarchy is the notion of 'reduction' we already saw in chapter 3, and which we shall study further in this chapter. Reducibility also serves to refine the structure of the hierarchy; we shall see a bit of this in section 4.3 for r.e. sets.

4.1 Many-one reducibility

Definition 4.1.1 A *preorder* on a set is a reflexive and transitive relation.

We usually denote the preorder relation by $\leq$. Note that if $\leq$ denotes a preorder on X, then the relation $\equiv$ defined by:

$$x \equiv y \text{ if and only if } x \leq y \text{ and } y \leq x$$

is an equivalence relation on X. The preorder $\leq$ then induces a partial order on the set of equivalence classes of $\equiv$. We call x and y *isomorphic* if $x \equiv y$.

Definition 4.1.2 Let $\leq$ be a preorder on a set X, and assume $x, y, z \in X$. We say that z is a *join* (or *least upper bound*) of x and y, if for every $w \in X$ we have:

$$z \leq w \text{ if and only if } x \leq w \text{ and } y \leq w$$

In a preorder, a join of two elements (if it exists) is unique up to isomorphism.

A preordered set $(X, \leq)$ with the property that every two elements of X have a join, is called an *upper pre-semilattice*.

We now repeat the definition of chapter 3 and define a preorder on $\mathcal{P}(\mathbb{N})$:

Definition 4.1.3 Let X and Y be subsets of $\mathbb{N}$. We call X *many-one reducible* to Y, and write $X \leq_m Y$, if there is a total recursive function F such that

$$X = F^{-1}(Y) = \{x \mid F(x) \in Y\}$$

Given such F, we also say that X is many-one reducible to Y *via* F.
We write $X \equiv_m Y$ for the corresponding equivalence relation: $X \equiv_m Y$ if $X \leq_m Y$ and $Y \leq_m X$.

The subscript m (for "many-one") emphasizes that the function F need not be one-to-one.

We summarize the most important features of the relation $\leq_m$ in an exercise:

Exercise 63

a) Prove that $\leq_m$ is a preorder on $\mathcal{P}(\mathbb{N})$.

b) Prove that if $X \leq_m Y$ then $\mathbb{N} - X \leq_m \mathbb{N} - Y$.

c) Prove that for all subsets X of $\mathbb{N}$ the following hold:

$$\begin{array}{lll} \mathbb{N} \leq_m X & \text{if and only if} & X \neq \emptyset \\ \emptyset \leq_m X & \text{if and only if} & X \neq \mathbb{N} \end{array}$$

d) Suppose $X \neq \emptyset$, $X \neq \mathbb{N}$ and X is recursive. Prove that X is a join of $\emptyset$ and $\mathbb{N}$.

e) Let X, Y be subsets of $\mathbb{N}$. Define the set $X \sqcup Y$ by:

$$X \sqcup Y = \{2x \mid x \in X\} \cup \{2x+1 \mid x \in Y\}$$

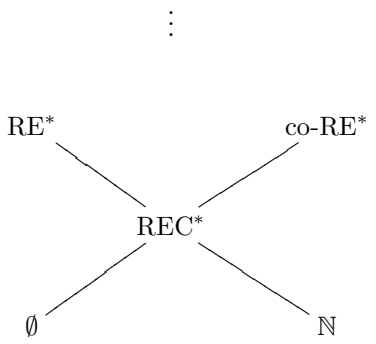
Prove that $X \sqcup Y$ is a join of X and Y.

f) Suppose $X \leq_m Y$. Prove: if Y is recursive, so is X; and if Y is r.e., so is X.

g) Prove: if X is r.e. and not recursive, then $X \not\leq_m \mathbb{N} - X$.

h) Prove: if $Y \neq \emptyset$ and $Y \neq \mathbb{N}$ and X is recursive, then $X \leq_m Y$.

Among other things, we see from this exercise that $\mathcal{P}(\mathbb{N})$ with the preorder $\leq_m$ is an upper pre-semilattice. The "bottom side" of the preorder $\leq_m$ looks like this (at least, if we restrict ourselves to the definable subsets of $\mathbb{N}$:



where:

$$\begin{array}{rcl} \text{REC}^* & = & \{X \mid X \neq \emptyset, X \neq \mathbb{N}, X \text{ is recursive}\} \\ \text{RE}^* & = & \{X \mid X \text{ is r.e., but not recursive}\} \\ \text{co-RE}^* & = & \{X \mid \mathbb{N} - X \text{ is r.e., but not recursive}\} \end{array}$$

Furthermore, REC* forms a *clique* w.r.t. $\leq_m$: that means that for all $X, Y \in \text{REC}^*$ we have $X \equiv_m Y$. Later (in section 4.3) we shall see that this is *not* the case for RE*!

The following exercise lets you pratice a bit with the relation $\leq_m$.

Exercise 64 Prove:

$$\{x \mid W_x \text{ is infinite}\} \equiv_m \{x \mid \phi_x \text{ is total}\} \equiv_m \{x \mid \mathrm{rge}(\phi_x) \text{ is infinite}\}$$

Let $\mathcal{X}$ be a collection of subsets of $\mathbb{N}$. An element $X \in \mathcal{X}$ is called *m-complete* in $\mathcal{X}$, if for every $Y \in \mathcal{X}$ we have $Y \leq_m X$ (X is "the largest element" van $\mathcal{X}$).

Exercise 65 Prove that $\mathcal{K}$ is m-complete in RE^*.

By coding of sequences we extend the relation $\leq_m$ to sets of k- and n-tuples: if $X \subseteq \mathbb{N}^k$ and $Y \subseteq \mathbb{N}^n$ we say that $X \leq_m Y$ if

$$\{j^k(\vec{x}) \mid \vec{x} \in X\} \leq_m \{j^n(\vec{y}) \mid \vec{y} \in Y\}$$

Equivalently: there are total recursive functions $F_1, \ldots, F_n : \mathbb{N}^k \to \mathbb{N}$, satisfying

$$X = \{\vec{x} \mid (F_1(\vec{x}), \ldots, F_n(\vec{x})) \in Y\}$$

4.2 The Arithmetical Hierarchy

Definition 4.2.1 The classes of Σ_n-, Π_n- and Δ_n-relations $R \subseteq \mathbb{N}^k$ (for all k) are defined as follows:

i) $R \subseteq \mathbb{N}^k$ is a Σ_0-relation if and only if R is a Π_0-relation, if and only if R is primitive recursive;

ii) R is a Σ_{n+1}-relation if and only if there is a Π_n-relation R' such that

$$R = \{\vec{x} \mid \exists y((\vec{x}, y) \in R')\}$$

iii) R is a Π_{n+1}-relation if and only if there is a Σ_n-relation R' such that

$$R = \{\vec{x} \mid \forall y((\vec{x}, y) \in R')\}$$

iv) R is a Δ_n-relation if and only if R is both a Σ_n-relation and a Π_n-relation.

We also write Σ_n (Π_n, Δ_n) for the class of all Σ_n (c.q. Π_n-,Δ_n-)-relations; and the notations "R is a Σ_n-relation" and "$R \in \Sigma_n$" have the same meaning.

Note the following:

Δ_1 is the class of recursive relations;

Σ_1 is the class of r.e. relations;

Π_1 is the class of complements of r.e. rlations.

Proposition 4.2.2 $\Sigma_n \cup \Pi_n \subseteq \Delta_{n+1}$

Proof. This is based on the logical trick of "dummy variables": for $S \subseteq \mathbb{N}^m$, let $S \times \mathbb{N} = \{(x_1, \ldots, x_{m+1}) \mid (x_1, \ldots, x_m) \in S\}$. Then

$$S = \{\vec{x} \mid \forall x_{m+1}((\vec{x}, x_{m+1}) \in S \times \mathbb{N})\} = \{\vec{x} \mid \exists x_{m+1}((\vec{x}, x_{m+1}) \in S \times \mathbb{N})\}$$

By induction one easily verifies: if $S \in \Sigma_n(\Pi_n)$ then $S \times \mathbb{N} \in \Sigma_n(\Pi_n)$. The statement now follows by induction:

$\Sigma_0 = \Sigma_0 \cup \Pi_0 \subseteq \Delta_1$ since all primitive recursive relations are recursive.

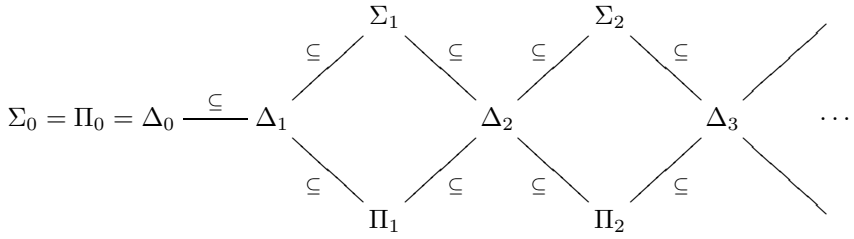
Suppose $S \in \Sigma_{n+1}$, so $S = \{\vec{x} \mid \exists y((\vec{x}, y) \in S')\}$ for some $S' \in \Pi_n$. Then

$$S = \{\vec{x} \mid \forall z \exists y((\vec{x}, y, z) \in S' \times \mathbb{N})\} \in \Pi_{n+2}$$

and $S \in \Sigma_{n+2}$ is clear, for $S' \in \Pi_{n+1}$ by induction hypothesis.

The argument for $S \in \Pi_{n+1}$ is identical. ∎

We get the following picture:



Our first main objective is now, to show that all these inclusions are *strict* (that is, in fact we could have replaced $\subseteq$ by $\subsetneq$); this is the so-called *Hierarchy Theorem* (4.2.10). In particular, this implies that $\Sigma_n \neq \Pi_n$ (why?).

Examples.

$$\{x \mid \phi_x \text{ is total}\} = \{x \mid \forall y \exists z T(1, x, y, z)\} \in \Pi_2$$

$$\{x \mid W_x \text{ is finite}\} = \{x \mid \exists n \forall z (T(1, x, j_1(z), j_2(z)) \Rightarrow j_1(z) \leq n)\} \in \Sigma_2$$

Proposition 4.2.3 *If $R \in \Sigma_n$ and $S \leq_m R$ via a primitive recursive function, then $S \in \Sigma_n$.*

Exercise 66

a) Prove proposition 4.2.5. Use induction.

b) Prove that proposition 4.2.5 also holds with Σ_n replaced by Π_n.

Proposition 4.2.4

i) *The classes Σ_n and Π_n are closed under binary unions and intersections.*

ii) *The classes Σ_n and Π_n are closed under bounded quantification*

iii) *($n > 0$) If $R \subseteq \mathbb{N}^{k+1}$ is a Σ_n-relation, then so is $\{\vec{x} \mid \exists w(\vec{x}, w) \in R\}$.*
If $R \subseteq \mathbb{N}^{k+1}$ is a Π_n-relation then so is $\{\vec{x} \mid \forall w(\vec{x}, w) \in R\}$.

Proof. We prove the properties i)–iii) simultaneously by induction on n. For $n = 0$, i) and ii) follow from proposition 2.1.2 and for iii) there is nothing to prove.
We do the induction step:

i) Suppose $R, S \in \Sigma_{n+1}$; there exist $R', S' \in \Pi_n$ such that

$$R = \{\vec{x} \mid \exists y(\vec{x}, y) \in R'\},\ S = \{\vec{x} \mid \exists w(\vec{x}, w) \in S'\}$$

so

$$R \cap S = \{\vec{x} \mid \exists y \exists w((\vec{x}, y) \in R' \text{ and } (\vec{x}, w) \in S')\}$$

Now $\{(\vec{x}, y, w) \mid (\vec{x}, y) \in R' \wedge (\vec{x}, w) \in S'\} \in \Pi_n$ by induction hypothesis, so by proposition 4.2.5 we have

$$\{(\vec{x}, z) \mid (\vec{x}, j_1(z)) \in R' \text{ and } (\vec{x}, j_2(z)) \in S'\} \in \Pi_n$$

whence

$$R \cap S = \{\vec{x} \mid \exists z((\vec{x}, j_1(z)) \in R' \text{ and } (\vec{x}, j_2(z)) \in S')\} \in \Sigma_{n+1}$$

And $R \cup S = \{\vec{x} \mid \exists y((\vec{x}, y) \in R' \cup S')\} \in \Sigma_{n+1}$, for $R' \cup S' \in \Pi_n$ by induction hypothesis.

For $R, S \in \Pi_{n+1}$ we apply De Morgan duality. For, $R \in \Sigma_n$ if and only if the complement of R is in Π_n; and $R \cap S$ is the complement of the union of their complements.

ii) Suppose $R \in \Sigma_{n+1}$. Then for some $R' \in \Pi_n$ we have that $R = \{\vec{x} \mid \exists y(\vec{x}, y) \in R'\}$. It follows that

$$\begin{array}{l}\{(\vec{x}, y) \mid \forall w \leq y(\vec{x}, w) \in R\} = \\ \{(\vec{x}, y) \mid \forall w \leq y \exists z(\vec{x}, w, z) \in R'\} = \\ \{(\vec{x}, y) \mid \exists \sigma(\mathrm{lh}(\sigma) = y + 1 \text{ and } \forall i \leq y(\vec{x}, i, (\sigma)_i) \in R')\}\end{array}$$

Now apply the induction hypothesis and proposition 4.2.5.

iii) Use coding of pairs and proposition 4.2.5:

$$\{\vec{x} \mid \exists y \exists w(\vec{x}, y, w) \in R\} \;=\; \{\vec{x} \mid \exists y(\vec{x}, j_1(y), j_2(y)) \in R\}$$

∎

Proposition 4.2.5 *Let $n > 0$. Then proposition 4.2.3 also holds without the restriction to reducibility via primitive recursive functions: if $S \in \Sigma_n$ and $R \leq_m S$, then $R \in \Sigma_n$.*

Exercise 67

i) Prove proposition 4.2.5. Hint: if $R \leq_m S$ via ϕ_f, then the statement "$x \in R$" is equivalent to both of the following statements:

$$\begin{array}{l}\exists y\,(T(1, f, x, y) \wedge U(y) \in S) \\ \forall y(T(1, f, x, y) \rightarrow U(y) \in S)\end{array}$$

ii) Show that the proposition also holds with Σ_n replaced by Π_n or Δ_n.

iii) Why is the condition $n > 0$ necessary?

In order to find some n for which a given relation is in Σ_n or Π_n, we apply the so-called *Tarski-Kuratowski algorithm*: first write R as a combination of primitive recursive relations, using the logical symbols $\wedge, \vee, \neg, \rightarrow, \exists, \forall$. Then move the quantifiers to the front of the definition using the familiar logical laws:

i) $\neg\forall x\phi \leftrightarrow \exists x\neg\phi$
$\neg\exists x\phi \leftrightarrow \forall x\neg\phi$

ii) $(\forall x\phi) \wedge \psi \leftrightarrow \forall x(\phi \wedge \psi)$
$(\forall x\phi) \vee \psi \leftrightarrow \forall x(\phi \vee \psi)$
if the variable x does not occur in ψ; similarly for $\exists$;

iii) $((\forall x\phi) \rightarrow \psi) \leftrightarrow \exists x(\phi \rightarrow \psi)$
$((\exists x\phi) \rightarrow \psi) \leftrightarrow \forall x(\phi \rightarrow \psi)$
if x does not occur in ψ.

The conditions in ii) and iii) make it sometimes necessary to "rename bound variables" (i.e., write $\forall y\phi(y)$ instead of $\forall x\phi(x)$). The result is what logicians call a *prenex normal form*: all quantifiers are at the front. Now one can say:

if the formula starts with $\exists$ and there are n quantifier alternations, the relation is Σ_{n+1};

if the formula starts with $\forall$ and there are n quantifier alternations, the relation is Π_{n+1};

if the formula is quantifier-free, the relation is Σ_0.

Example. Let $R = \{x \mid \phi_x \text{ is total and bounded}\} =$

$$\begin{array}{ll} = & \{x \mid \forall y \exists z T(1, x, y, z) \wedge \exists n \forall y \forall z(T(1, x, y, z) \rightarrow U(z) \leq n)\} \\ = & \text{(rename bound variables)} \\ & \{x \mid \forall y \exists z T(1, x, y, z) \wedge \exists n \forall v \forall w(T(1, x, v, w) \rightarrow U(w) \leq n)\} \\ = & \{x \mid \forall y \exists z(T(1, x, y, z) \wedge \exists n \forall v \forall w(\cdots))\} \\ = & \{x \mid \forall y \exists z \exists n \forall v \forall w(\cdots)\}\end{array}$$

with $(\cdots)$ primitive recursive; so R is a Π_3-relation. But after renaming the bound variables, we could also have started by moving the quantifiers from the second part to the front. Then we get:

$$\{x \mid \exists n \forall v \forall w \forall y \exists z (\cdots)\}$$

and we conclude that R is also a Σ_3-relation. We see that this R is a Δ_3-relation.
The Tarski-Kuratowski algorithm always gives us some n such that our relation is Σ_n or Π_n (provided our relation can be defined in logical terms), but seldom the most 'economical'; and it does not tell us anything about Δ_n-relations (as in the example).
In order to prove that a given relation R (about which one knows, for example, that it is a Δ_{n+1}-relation) is *not* Σ_n or Π_n, the algorithm offers no help, and we have to get to work.

Example 4.2.6 The relation $R = \{x \mid \phi_x \text{ is total and bounded}\}$ is not a Π_2-relation.

Proof. Suppose for a contradiction that $R \in \Pi_2$. Then we would have for some primitive recursive relation $S(x, y, z)$:

$$\phi_x \text{ is total and bounded} \leftrightarrow \forall y \exists z S(x, y, z)$$

Using the Recursion Theorem, pick an index e satisfying, for all y:

$$e{\cdot}y \simeq \max(\{w \leq y \mid \forall v \leq w \exists z \leq y S(e, v, z)\})$$

where, by convention, $\max(\emptyset) = 0$. Then ϕ_e is total. However, by choice of e one can check that

$$\phi_e \text{ is bounded} \leftrightarrow \exists y \forall z \neg S(e, y, z)$$

On the other hand, from the assumed property of S it follows that

$$\exists y \forall z \neg S(e, y, z) \leftrightarrow \phi_e \text{ is not total and bounded}$$

hence, we have obtained the desired contradiction. ∎

Exercise 68 Prove that $\{x \mid \phi_x \text{ is total and bounded}\}$ is also not Σ_2.
[Hint: the definition of "total and bounded" consists of a Σ_2-part and a Π_2-part. In the Example, in order to prove that the relation is not Π_2, the Σ_2-part is exploited. In this exercise, you must work with the Π_2-component]

The given example shows us, that the inclusions $\Pi_2 \subseteq \Delta_3$ and $\Sigma_2 \subseteq \Delta_3$ are strict. We shall now see that *all* inclusions $\Pi_n \subseteq \Delta_{n+1}$, $\Sigma_n \subseteq \Delta_{n+1}$, $\Delta_n \subseteq \Sigma_n$, $\Delta_n \subseteq \Pi_n$ are strict.

Definition 4.2.7 Define for all $n, k \geq 1$ a Σ_n-relation $E_n^{(k)} \subseteq \mathbb{N}^{k+1}$ as follows:

$$E_n^{(k)} = \{(z, x_1, \ldots, x_k) \mid \exists y_1 \forall y_2 \cdots \exists y_n T(n + k - 1, z, j^{n+k-1}(x_1, \ldots, x_k, y_1, \ldots, y_{n-1}), y_n)\}$$

if n is odd, and

$$E_n^{(k)} = \{(z, x_1, \ldots, x_k) \mid \exists y_1 \forall y_2 \cdots \forall y_n \neg T(n + k - 1, z, j^{n+k-1}(x_1, \ldots, x_k, y_1, \ldots, y_{n-1}), y_n)\}$$

if n is even.

Theorem 4.2.8 (Kleene Normal Form Theorem) *Let $n > 0$.*
For every Σ_n-relation $R \subseteq \mathbb{N}^k$ there is a number z such that

$$R = \{\vec{x} \in \mathbb{N}^k \mid (z, \vec{x}) \in E_n^{(k)}\}$$

and for every Π_n-relation $S \subseteq \mathbb{N}^k$ there is a z such that

$$S = \{\vec{x} \in \mathbb{N}^k \mid (z, \vec{x}) \notin E_n^{(k)}\}$$

If $R \subseteq \mathbb{N}^k$ is a Σ_n-relation, then the representation of R as $\{\vec{x} \mid (z, \vec{x}) \in E_n^{(k)}\}$ is called a Σ_n-*normal form* of R, and z is a Σ_n-*index* for R. Analogously, we have Π_n-normal forms and indices.

Proof. The statement about the Π_n-relations follows immediately from the one about Σ_n-relations. We prove this one by induction on $n \geq 1$:
Every Σ_1-relation is $W_e^{(k)}$ for a certain e; in other words, it can be written as $\{\vec{x} \mid \exists y T(k, e, j^k(\vec{x}), y)\}$. Suppose R is a Σ_{n+1}-relation. Then $R = \{\vec{x} \mid \exists y S(\vec{x}, y)\}$ for some Π_n-relation $S \subseteq \mathbb{N}^{k+1}$; hence $\mathbb{N}^{k+1} - S$ is Σ_n and by induction hypothesis we have $(\vec{x}, y) \notin S \leftrightarrow (z, \vec{x}, y) \in E_n^{(k+1)}$ for some z. Now assume that n is odd (the other case is proved in the same way). We get:

$$\begin{array}{ll}
\vec{x} \in R & \leftrightarrow \\
\exists y[(z, \vec{x}, y) \notin E_n^{(k+1)}] & \leftrightarrow \\
\exists y \neg \exists y_1 \forall y_2 \cdots \exists y_n T(k+n, z, j^{k+n}(\vec{x}, y, y_1, \ldots, y_{n-1}), y_n) & \leftrightarrow \\
\exists y \forall y_1 \cdots \forall y_n \neg T(k+n, z, j^{k+n}(\vec{x}, y, y_1, \ldots, y_{n-1}), y_n) & \leftrightarrow \\
(z, \vec{x}) \in E_{n+1}^{(k)} &
\end{array}$$

∎

Proposition 4.2.9 *The relation $E_n^{(k)}$ is Σ_n but not Π_n.*

Proof. We do this for $k = 1$. Suppose $E_n^{(1)} \in \Pi_n$, then also $\{x \mid (x, x) \in E_n^{(1)}\} \in \Pi_n$, and by the Normal Form Theorem there is a z such that for all x:

$$(x, x) \in E_n^{(1)} \text{ if and only if } (z, x) \notin E_n^{(1)}$$

But of course, this yields a contradiction for $x = z$. ∎

Corollary 4.2.10 (Hierarchy Theorem) *All inclusions in the arithmetical hierarchy are strict.*

Proof. Indeed, the fact that $E_n^{(k)} \in \Sigma_n - \Pi_n$ implies that $\Delta_n \subsetneq \Sigma_n$ and that $\Pi_n \subsetneq \Delta_{n+1}$. And from the fact that $\mathbb{N}^{k+1} - E_n^{(k)}) \in \Pi_n - \Sigma_n$ it follows that $\Delta_n \subsetneq \Pi_n$ and $\Sigma_n \subsetneq \Delta_{n+1}$. ∎

Exercise 69 Prove: if R is m-complete in Σ_n, then $R \notin \Pi_n$.

Exercise 70 Prove: if A is m-complete in Σ_n and B is m-complete in Π_n, then $A \sqcup B \in \Delta_{n+1} \setminus \Sigma_n \cup \Pi_n$.

Proposition 4.2.11 *The relations $E_n^{(k)}$ are m-complete in Σ_n en their complements are therefore m-complete in Π_n.*

Proof. This is immediate from the Normal Form Theorem (4.2.8). ∎

The *classification* of a set $X \subseteq \mathbb{N}^k$ is determining its exact position in the arithmetical hierarchy. A straightforward way to proceed is as follows: by the Tarski-Kuratowski algorithm one tries to find a smallest possible n such that $X \in \Sigma_n$ or Π_n of Δ_n. If one knows that $X \in \Sigma_n$ and doesn't succeed in proving that $X \in \Pi_n$, then one tries to prove that for appropriate k, $E_n^{(k)} \leq_m X$ holds.

Example 4.2.12 The set $\{x \mid \phi_x \text{ is total}\}$ is strictly Π_2.

Proof. Let us call this set Tot. We shall show that $\mathbb{N}^2 - (E_2^{(1)}) \leq_m X$.
Recall that $\mathbb{N}^2 - (E_2^{(1)}) = \{(z, x) \mid \forall y_1 \exists y_2 T(2, z, j(x, y_1), y_2)\}$. Let g be an index which satisfies for all z, x, y_1:

$$g{\cdot}(z, x, y_1) \simeq z{\cdot}(x, y_1)$$

We now have:

$$\begin{array}{ll} (w_1, w_2) \notin E_2^{(1)} & \text{if and only if} \\ \forall y_1 \exists y_2 T(2, w_1, j(w_2, y_1), y_2) & \text{if and only if} \\ \forall y_1 \exists y_2 T(1, S_1^2(g, w_1, w_2), y_1, y_2) & \text{if and only if} \\ \phi_{S_1^2(g,w_1,w_2)} \text{ is total} & \end{array}$$

Dus $\mathbb{N}^2 - (E_2^{(1)}) \leq_m X$, as desired. ∎

Exercise 71 Show that $\{e \mid W_e \text{ is a singleton set}\}$ is strictly Δ_2.

Exercise 72 Find for each of the following relations an n, as small as you can, such that they are in Σ_n, Π_n or Δ_n:

i) $\{e \mid W_e \text{ is finite}\}$

ii) $\{e \mid \text{rge}(\phi_e) \text{ is infinite}\}$

iii) $\{e \mid \phi_e \text{ is constant (possibly partial)}\} = \{e \mid \phi_e \text{ has at most one value}\}$

iv) $\{j(e, f) \mid W_e \leq_m W_f\}$

v) $\{e \mid W_e \text{ is } m\text{-complete in } \Sigma_1\}$

Then, classify the first three of these *completely*, by showing that they are m-complete in the class you found.

4.2.1 Some exact classifications

In this section I treat two examples of exact classifications, which are not so immediate. There is extensive theory about obtaining such classifications: see [38]. However, this goes beyond the scope of this course. Therefore, the classifications below are done with "bare hands".

Functions with no total extension

Our first example is the set

$$X = \{x \mid \phi_x \text{ cannot be extended to a total recursive function}\}$$

By exercise 51, the set X is nonempty.

Exercise 73 Prove that X is not Σ_1:

a) using the Myhill-Shepherdson Theorem (3.3.3)

b) without using the Myhill-Shepherdson Theorem

By the Tarski-Kuratowski algorithm we easily find out that X is a Π_3-set: we have that $e \in X$ if and only if the following condition holds:

$$\forall f(\text{if } \phi_f \text{ is total then } \phi_e \text{ is not a subfunction of } \phi_f)$$

We can rewrite this as

$$\forall f(\forall x \exists y T(1, f, x, y) \rightarrow \exists uvw(T(1, e, u, v) \wedge T(1, f, u, w) \wedge U(v) \neq U(w)))$$

which we bring in prenex normal form:

$$\forall f \exists uvwx \forall y(T(1, f, x, y) \rightarrow (T(1, e, u, v) \wedge T(1, f, u, w) \wedge U(v) \neq U(w)))$$

hence $X \in \Pi_3$.

Proposition 4.2.13 *X is m-complete in Π_3.*

Proof. Let A be an arbitrary Π_3-set:

$$A = \{f \mid \forall x \exists y \forall z R(f, x, y, z)\}$$

where R is some primitive recursive predicate. We have to find a total recursive function G such that $A = G^{-1}(X)$.
By the Smn-theorem there is a primitive recursive function G such that for all f and z, $G(f)\cdot z$ is given by the following instructions:

Find the least W such that either (1) or (2) below holds:

1) $\exists x \leq j_1(z) \forall y \leq j_2(z) \exists w \leq W \neg R(f, x, y, w)$

2) $\forall y \leq j_2(z) \exists w \leq W T(1, j_1(z), j(j_1(z), y), w)$

Let $G(f)\cdot z$ be undefined if such a W is not found.

If W has been found, put $G(f)\cdot z = W$ if (1) holds.

If (1) does not hold, find the least w satisfying $T(1, j_1(z), z, w)$, and put $G(f)\cdot z = U(w) + 1$.

Now suppose $f \in A$ so $\forall x \exists y \forall z R(f, x, y, z)$, and suppose ϕ_u is a total function.
For every x let $n_x = \max(\{m \mid \forall y < m \exists z \neg R(f, x, y, z)\})$; and define

$$N(u) = \max(\{n_x \mid x \leq u\}) + 1$$

Then $G(f)\cdot j(u, N(u))$ is always defined (the required W is always found by (2), because ϕ_u is total), but the obtained W can never satisfy (1) (check this yourself); so $G(f)\cdot j(u, N(u)) = u\cdot j(u, N(u)) + 1$, and therefore ϕ_u is not a total function which extends $\phi_{G(f)}$. This holds for every index u such that ϕ_u is total, and hence $G(f) \in X$.

If $f \notin A$ then $\exists x \forall y \exists z \neg R(f, x, y, z)$; let N be minimal such that $\forall y \exists z \neg R(f, N, y, z)$. Then we have:

If $j_1(z) \geq N$, then $G(f)\cdot z$ is defined (the required W will be found because (1) will occur).

If $j_1(z) < N$ then the partial function $\phi_{G(f)}$ depends on $j_1(z)$:

The function $\lambda n.G(f)\cdot j(j_1(z), n)$ is total if the function $\lambda x.j_1(z)\cdot j(j_1(z), x)$ is total, and has finite domain if $\lambda x.j_1(z)\cdot j(j_1(z), x)$ is not total. In both cases, the function $\lambda n.G(f)\cdot j(j_1(z), n)$ has a total recursive extension $F_{j_1(z)}$.

Now define

$$H(z) = \begin{cases} F_0(j_2(z)) & \text{if } j_1(z) = 0 \\ \vdots & \vdots \\ F_{N-1}(j_2(z)) & \text{if } j_1(z) = N - 1 \\ G(f)\cdot z & \text{otherwise} \end{cases}$$

Then H is a total recursive function which extends $\phi_{G(f)}$. ∎

Primitive recursive functions

Our second example is the set

$$\text{PRIM} = \{e \mid \phi_e \text{ is primitive recursive}\}$$

Proposition 4.2.14 *The set* PRIM *is strictly Δ_3.*

Proof. In the proof of 2.1.11 we have defined codes for definitions of primitive recursive functions. From this definition it is not hard to see, that the set of such codes, let's call it C, is recursive (in fact, primitive recursive).

Moreover, since the partial recursive functions are closed under double recursion (2.4.7), there is a partial recursive function ψ such that for all $n \in C$ and m arbitrary, $\psi(n, m) = F_n(m)$, where F_n denotes the primitive recursive function whose definition is coded by n.

Define a total recursive function E as follows:

$$E(n,m) \;=\; \begin{cases} 0 & \text{if } n \notin C \\ F_n(m)+1 & \text{if } n \in C \end{cases}$$

Now we can write the set PRIM as intersection:

$$\text{PRIM} \;=\; \begin{cases} \{e \,|\, \phi_e \text{ is total}\} \\ \cap \\ \{e \,|\, \exists n \forall m \forall k (T(1,e,m,k) \rightarrow E(n,m) = U(k)+1)\} \end{cases}$$

We see that PRIM is the intersection of a Π_2-set and a Σ_2-set. Hence, by proposition 4.2.4, PRIM is in Δ_3 (check!). We need to show that PRIM is neither a Σ_2, nor a Π_s-set.

We show that PRIM is not Σ_2 by showing that every Π_2-set is many-one reducible to PRIM. Let A be an arbitrary Π_2-set, so $A \;=\; \{e \,|\, \forall x \exists y R(e,x,y)\}$ with R recursive. Choose an index g such that

$$g{\cdot}(e,x) \simeq \begin{cases} 0 & \text{if } \exists y R(e,x,y) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Then clearly, since the function $\phi_{S^1_1(g,e)}$ is constant 0 on its domain, it is primitive recursive if and only if it is total. Hence we have: $S^1_1(g,e) \in \text{PRIM}$ if and only if $\forall x \exists y R(e,x,y)$, in other words: if and only if $e \in A$. So the function $e \mapsto S^1_1(g,e)$ reduces A to PRIM.

Similarly, we show that PRIM is not Π_2 by showing that every Σ_2-set is many-one reducible to PRIM. We use the recursive function $\psi(n,m) = F_n(m)$ from above. Suppose $A = \{e \,|\, \exists x \forall y R(e,x,y)\}$ with R recursive.

Let, for any number n, $M(n)$ be the least $m \leq n$ such that for all $k \leq n$, $R(e,m,k)$ holds. Of course, $M(n)$ need not exist, but this can be decided recursively. Let g be an index of the following total recursive function:

$$\phi_g(e,n) \;=\; \begin{cases} 0 & \text{if } n = 0 \\ 0 & \text{if } n > 0,\ M(n) \text{ and } M(n-1) \text{ both} \\ & \text{exist, and } M(n) = M(n-1) \\ F_k(n)+1 & \text{otherwise, where } k \text{ is such that } n \text{ is} \\ & \text{the } k+1\text{-st number for which the first clause fails} \end{cases}$$

Now suppose $e \in A$. Let N be the least number such that $\forall y R(e,N,y)$. Then for all $i < N$ there is a least number y_i such that $\neg R(e,i,y_i)$. Therefore if $m > \max\{y_i \,|\, i < N\} + 1$, then always $M(m) = M(m-1)$ will hold, so $\phi_g(e,m) = 0$. We see that $\phi_{S^1_1(g,e)}(m)$ will be 0 for all but finitely many m, hence $\phi_{S^1_1(g,e)}$ is primitive recursive: $S^1_1(g,e) \in \text{PRIM}$.

Conversely, suppose $e \notin A$. Then the first clause in the definition og $\phi_g(e,n)$ will fail infinitely often. Tha means that for all k there will be an n such that $\phi_{S^1_1(g,e)}(n)$ differs from $F_k(n)$. Therefore $\phi_{S^1_1(g,e)}$ cannot be primitive recursive. We conclude that the function $e \mapsto S^1_1(g,e)$ reduces A to PRIM, as desired. ∎

4.3 R.e. sets again: recursive, m-complete, and in-between

In this section we shall see that the concept of many-one reducibility gives a nontrivial structure on the class of all r.e. sets.

First we consider r.e. sets which are extensional for indices of r.e. sets (see section 3.3). These sets will turn out to be either recursive or m-complete in Σ_1.

Then we characterize the m-complete r.e. sets: these are "recursively isomorphic" to $\mathcal{K}$, the standard set. Moreover, they are precisely the creative sets (see Exercise 52): this is Corollary 4.3.6 below.

Finally we show that there exist r.e. sets which are neither recursive, nor m-complete. The structure of the preordered set $(\mathcal{RE}, \leq_m)$ is quite complicated, and topic of ongoing research.

4.3.1 Extensional r.e. sets

Exercise 74 Show the following corollary of Rice's Theorem: if $X \subseteq \mathbb{N}$ is recursive and extensional for indices of r.e. sets, then $X = \emptyset$ or $X = \mathbb{N}$.

Proposition 4.3.1 *If $X \subseteq \mathbb{N}$ is r.e. and extensional for indices of r.e. sets, then X is either recursive, or m-complete in Σ_1.*

Proof. Suppose X satisfies the assumptions of the proposition, and suppose also that $X \neq \emptyset$, $X \neq \mathbb{N}$. Then by Exercise 74 we have that X is not recursive, so we show that X is m-complete in Σ_1.

The Rice-Shapiro Theorem implies that if $n \in X$ for some n such that $W_n = \emptyset$, then $X = \mathbb{N}$. So we may assume that indices for the empty set are not elements of X. On the other hand we have assumed X nonempty; choose $e \in X$. Let $g \in \mathbb{N}$ be arbitrary.

By the Smn-theorem there exists a primitive recursive function F satisfying for all x, y:

$$F(x) \cdot y \simeq \mu z.[T(1, e, y, j_1(z)) \wedge T(1, g, x, j_2(z))]$$

Then the following holds:

$$W_{F(x)} = \begin{cases} W_e & \text{if } x \in W_g \\ \emptyset & \text{otherwise} \end{cases}$$

In other words, by the assumptions on X we have for all x: $x \in W_g$ if and only if $F(x) \in X$. So $W_g \leq_m X$ via F, and since g was arbitrary we have proved that X is m-complete in Σ_1. ∎

4.3.2 m-Complete r.e. sets

We recall from Exercise 52: a set $X \subseteq \mathbb{N}$ is called *creative* if X is r.e. and there is a partial recursive function ψ with the following property:

$$\text{Whenever } W_e \cap X = \emptyset \text{ then } \psi(e) \text{ is defined and } \psi(e) \notin W_e \cup X$$

(X is said to be creative *via* ψ)

Exercise 75 Prove that $\mathcal{K}$ is creative.

In this section we shall show that an r.e. set is m-complete precisely when it is creative.

Definition 4.3.2 By the notation $X \leq_1 Y$ we mean that X is many-one reducible to Y via an *injective* total recursive function. We say X is 1-complete in a class of sets, if for every set Y in that class we have $Y \leq_1 X$.

We call subsets X, Y of $\mathbb{N}$ *recursively isomorphic* if there is a total recursive bijective function $h : \mathbb{N} \to \mathbb{N}$ with $h[X] = Y$.

The following theorem is sometimes called the "recursive analogue of the Cantor-Bernstein Theorem" (in set theory, the Cantor-Bernstein Theorem says that whenever there are injective functions $X \to Y$ and $Y \to X$, there is also a bijection from X to Y).

Theorem 4.3.3 (Myhill's Isomorphism Theorem) *For any two sets $A, B \subseteq \mathbb{N}$ we have: if $A \leq_1 B$ and $B \leq_1 A$ then A and B are recursively isomorphic.*

Proof. Suppose $A \leq_1 B$ via f and $B \leq_1 A$ via g, so f and g are total recursive injective functions satisfying $A = f^{-1}(B)$ and $B = g^{-1}(A)$. We now construct, by primitive recursion, a sequence $(h_s)_{s\geq 0}$ of coded sequences, such that for every $i < k < \mathrm{lh}(h_s)$ we have: if $j_1((h_s)_i) = j_1((h_s)_{i'})$ then $(h_s)_i = (h_s)_{i'}$, *and* $j_2((h_s)_i) = j_2((h_s)_{i'})$ then $(h_s)_i = (h_s)_{i'}$. We can therefore think of the sequences h_s as graphs of finite injective functions; we shall use the notation h_s also to denote these functions. Note, that we can recursively from h_s, obtain its domain, range and inverse:

$$\begin{array}{rcl} x \in \mathrm{dom}(h_s) & \text{if and only if} & \exists i < \mathrm{lh}(h_s)\, j_1((h_s)_i) = x \\ x \in \mathrm{rge}(h_s) & \text{if and only if} & \exists i < \mathrm{lh}(h_s)\, j_2((h_s)_i) = x \\ (h_s)^{-1}(y) & = & \mu x < h_s . \exists i < \mathrm{lh}(h_s)\, (h_s)_i = j(x,y) \end{array}$$

In the end, the finite injective functions h_s will be pieced together to form a bijection $h : \mathbb{N} \to \mathbb{N}$.

Let h_0 be the code of the empty sequence. Now suppose h_s is given, such that h_s is the graph of a finite injective function, and for every $x \in \mathrm{dom}(h_s)$ we have $x \in A$ if and only if $h_s(x) \in B$.

If $s+1 = 2x+1$ we define $h(x)$. If already $x \in \mathrm{dom}(h_s)$ we put $h_{s+1} = h_s$. Otherwise, we compute the sequence $f(x), f(h_s^{-1}(f(x))), \ldots, f(h_s^{-1}f)^n(x), \ldots$, until the first element y is found which satisfies $y \notin \mathrm{rge}(h_s)$. Such a y must exist because f and h_s are injective and $x \notin \mathrm{dom}(h_s)$. Put $h_{s+1} = h_s * \langle j(x,y)\rangle$. Check yourself that h_{s+1} is still injective and still satisfies for $n \in \mathrm{dom}(h_{s+1})$: $n \in A$ if and only if $h_{s+1}(n) \in B$.

If $s+1 = 2x+2$, we define $h^{-1}(x)$. This is done in the same way, with f, h_s, dom en rge replaced by (respectively) g, h_s^{-1}, rge en dom. ∎

Proposition 4.3.4 *If a set X is creative, then there is an injective total recursive function P such that X is creative via P.*

Proof. Let X be creative via ϕ. Check that there is a primitive recursive function G is satisfying

$$W_{G(x)} = \begin{cases} W_x & \text{if } \phi(x) \text{ is defined} \\ \emptyset & \text{otherwise} \end{cases}$$

Define Q by: $Q(x) = \phi(x)$ or $Q(x) = \phi(G(x))$, whichever computation terminates first. Then the function Q is total, for if $\phi(x)$ is undefined, then $W_{G(x)} = \emptyset$, hence $\phi(G(x))$ is defined because X is creative via ϕ.

We also have that if $W_x \cap X = \emptyset$ then $\phi(x)$ is defined, hence $W_x = W_{G(x)}$, so in this case $\phi(x) \notin X \cup W_x$ and $\phi(G(x)) \notin X \cup W_x$. This means that X is also creative via Q, and Q is total. It remains to modify Q to an injective function P.

Let H be a primitive recursive function satisfying for all x:

$$W_{H(x)} = W_x \cup \{Q(x)\}$$

It now folows that:

$$W_{H^{n+1}(x)} = W_x \cup \{Q(x), QH(x), \ldots, QH^n(x)\}$$

Define the function P as follows:

- $P(0) = Q(0)$
- Suppose $P(0), \ldots, P(x)$ have been defined. Now enumerate the sequence $Q(x+1), QH(x+1), \ldots, QH^n(x+1), \ldots$, until a) or b) occurs:
  a) For some n, $QH^n(x+1) \notin \{P(0), \ldots, P(x)\}$. Put $P(x+1) = QH^n(x+1)$.
  b) A repitition occurs: for some n, we have $QH^{n+1}(x+1) \in \{Q(x+1), \ldots, QH^n(x+1)\}$. Then we must have $W_{x+1} \cap X \neq \emptyset$ (check); simply let $P(x+1) = \mu y.[y \notin \{P(0), \ldots, P(x)\}]$.

Convince yourself that P is injective and total, and that X is creative via P. ∎

Proposition 4.3.5 (Myhill) *If a set X is creative, then $\mathcal{K} \leq_1 X$*

Proof. Choose (by proposition 4.3.4) an injective and total recursive function P such that X is creative via P. Let F be a primitive recursive function satisfying

$$W_{F(x,y)} = \begin{cases} \{P(x)\} & \text{if } y \in \mathcal{K} \\ \emptyset & \text{otherwise} \end{cases}$$

By the Recursion Theorem, there is a primitive recursive, injective function N satisfying for all y:

$$W_{N(y)} = W_{F(N(y),y)} = \begin{cases} \{P(N(y))\} & \text{if } y \in \mathcal{K} \\ \emptyset & \text{otherwise} \end{cases}$$

(Choose by the Recursion Theorem an n such that for all y, x, $n{\cdot}(y,x) \simeq F(S_1^1(n,y),y){\cdot}x$. Let $N(y) = S_1^1(n,y)$. Then $W_{N(y)} = W_{F(N(y),y)}$ and N can be assumed to be injective because it is an Smn-function (Exercise 29))
The following hold:

$$\text{If } y \in \mathcal{K} \text{ then } W_{N(y)} = \{P(N(y))\} \text{ hence } W_{N(y)} \cap X \neq \emptyset \text{ hence } P(N(y)) \in X$$

$$\text{If } y \notin \mathcal{K} \text{ then } W_{N(y)} = \emptyset \text{ hence } W_{N(y)} \cap X = \emptyset \text{ hence } P(N(y)) \notin X$$

so we conclude that $\mathcal{K} \leq_1 X$ via the composition PN. ∎

Exercise 76 Prove:

i) If X is creative then X is strictly r.e.

ii) If X is creative, then there is an infinite r.e. set which is disjoint from X

iii) If X is creative and Y is r.e. and $X \leq_m Y$, then Y is creative

Corollary 4.3.6 *For a set $X \subseteq \mathbb{N}$ the following assertions are equivalent:*

i) X is creative

ii) X is 1-complete in Σ_1;

iii) X is m-complete in Σ_1;

iv) There is a total recursive bijective function h such that $h[X] = \mathcal{K}$

Exercise 77 Prove Corollary 4.3.6. Hint: first prove that $\mathcal{K}$ is 1-complete in Σ_1. Use Exercise 76, proposition 4.3.5 and Theorem 4.3.3.

Exercise 78 Call a disjoint pair (A, B) of r.e. sets *effectively inseparable* if there exists a partial recursive function ψ which is such that for all x and y: if $A \subseteq W_x$ and $B \subseteq W_y$ and $W_x \cap W_y = \emptyset$, then $\psi(\langle x, y\rangle)$ is defined and $\psi(\langle x, y\rangle) \notin W_x \cup W_y$.

a) Show: we may assume that ψ is total and injective.

b) Show: the sets $A = \{x \mid x{\cdot}x = 0\}$ and $B = \{x \mid x{\cdot}x = 1\}$ are effectively inseparable.

c) Show: if (A, B) is effectively inseparable then A and B are creative.

Exercise 79 Show that there exist disjoint r.e. sets $A, B \subseteq \mathbb{N}$ such that (A, B) is recursively inseparable, but not effectively inseparable.

4.3.3 Simple r.e. sets: neither recursive, nor m-complete

Definition 4.3.7 An r.e. set A is called *simple* if $\mathbb{N} - A$ is infinite but does not contain an infinite r.e. subset (in other words: A intersects every infinite r.e. set).

Exercise 80 Show: if A is a simple r.e. set, then A is neither recursive, nor creative.

Since we know from Corollary 4.3.6 that the m-complete sets are precisely the creative sets, it suffices for us to show that simple sets exist.

Theorem 4.3.8 (Post, 1944) *There exists a simple set.*

Proof. Definne the partial recursive function ψ by:

$$\psi(e) \simeq j_1(\mu z.[T(1, e, j_1 z, j_2 z) \wedge j_1 z > 2e])$$

Let $S = \text{rge}(\psi)$. Then S is certainly r.e. Because always $\psi(e) > 2e$ (if $\psi(e)$ is defined), the set S contains, for each e, at most e elements from the set $\{0, 1, \ldots, 2e\}$; and therefore the set $\mathbb{N} - S$ is infinite.
Moreover, if $B = W_e$ is an infinite r.e. set, then $\psi(e)$ is defined, and $\psi(e) \in S \cap W_e$ (check this!). So S intersects every infinite r.e. set. This implies that $\mathbb{N} - S$, being infinite and disjoint from S, cannot be r.e.; so S is not recursive.
We conclude that S is simple, as claimed. ∎

4.3.4 Non-Arithmetical Sets; Tarski's Theorem

In section 2.1.2 we introduced the language of ordered rings, also called the *language of arithmetic* by logicians. We formulated the theory $I\Sigma_1$ in this language, and stated that for every k-ary primitive recursive function F there is a formula $\phi(x_1, \ldots, x_{k+1})$ in the language, such that the sentences:

$$\begin{array}{c} \psi(\overline{n_1}, \ldots, \overline{n_k}, \overline{F(n_1, \ldots, n_k)}) \qquad \text{for all } n_1, \ldots, n_k \\ \forall \vec{x} y z\, (\psi(\vec{x}, y) \wedge \psi(\vec{x}, z) \rightarrow y = z) \\ \forall \vec{x} \exists y \psi(\vec{x}, y) \end{array}$$

are all consequences of $I\Sigma_1$ (i.e., true in every model of $I\Sigma_1$).

This means, that for every set $A \subseteq \mathbb{N}^k$ in the Arithmetical Hierarchy, there is a formula $\psi_A(x_1, \ldots, x_k)$ in the language of arithmetic, such that for every k-tuple $n_1, \ldots, n_k$ of natural numbers the following holds:

$$(n_1, \ldots, n_k) \in A \;\Leftrightarrow\; \mathbb{N} \models \psi_A(\overline{n_1}, \ldots, \overline{n_k})$$

where $\mathbb{N}$ is the obvious structure for the language of arithmetic.

In the course of proving his famous Incompleteness Theorems, Gödel constructed a *coding* of formulas of the arithmetical language; assigning to every formula φ a number $\ulcorner \varphi \urcorner$. This coding was done in such a way that he was able to prove the following lemma (we present a simplified version):

Lemma 4.3.9 (Diagonalization Lemma; Gödel, 1930) *For every formula $\varphi(x)$ in one free variable x there exists a sentence ψ such that the sentence*

$$\psi \;\leftrightarrow\; \varphi(\overline{\ulcorner \psi \urcorner})$$

is a consequence of $I\Sigma_1$.

The logician Alfred Tarski extracted the following consequence of this lemma:

Theorem 4.3.10 (Tarski's Theorem, 1936) *The set*

$$T \;=\; \{\ulcorner \psi \urcorner \mid \psi \text{ is an arithmetical sentence such that } \mathbb{N} \models \psi\}$$

is not arithmetical (i.e., does not belong to the Arithmetical Hierarchy).

Proof. Suppose T were arithmetical. Then there would be a formula $\chi_T(x)$ in one variable, such that for any number n we would have: $n \in T$ if and only if $\chi_T(\overline{n})$ is true in the model $\mathbb{N}$.

But now apply the Diagonalization Lemma 4.3.9 to the formula $\neg\chi_T$. We obtain a sentence ψ such that the equivalence $\psi \leftrightarrow \neg\chi_T(\overline{\ulcorner\psi\urcorner})$ is a consequence of $I\Sigma_1$, and therefore true in $\mathbb{N}$. We obtain:

$$\begin{array}{ll} \mathbb{N} \models \psi & \Leftrightarrow \\ \ulcorner\psi\urcorner \in T & \Leftrightarrow \\ \mathbb{N} \models \chi_T(\overline{\ulcorner\psi\urcorner}) & \Leftrightarrow \\ \mathbb{N} \models \neg\psi & \end{array}$$

which is a clear contradiction. ∎

Chapter 5

Relative Computability and Turing Reducibility

The stratification of subsets of $\mathbb{N}$ in $\equiv_m$-classes was important because of the close connection with the Arithmetical Hierarchy.

However, for other important issues in Computability Theory we need a different type of reduction. The question we shall deal with in this chapter is: which functions are computable, if a certain function F is "given" to me as computable, for example by some infinite database?

In other words, which functions can be computed partial-recursively from F?

5.1 Functions partial recursive in F

Definition 5.1.1 The class of functions *primitive recursive in* a given function F is the least class of functions which contains the basic functions from Definition 2.1.1, as well as F, and is closed under definition by composition and primitive recursion.

Exercise 81 Show that the function $\lambda x.F^{F(0)}(x)$ is primitive recursive in F.

Definition 5.1.2 The class of functions *partial recursive in* F is defined as in Definition 2.2.1, with 'primitive recursive' replaced by 'primitive recursive in F'.

We can also introduce the notion 'computable in F', by extending the concept of Register Machine. We now think of a machine which is able to produce the values of F in a 'magical' way, which explains the name "oracle" in the next definition.

Definition 5.1.3 The programming language for the *Register Machine with oracle* F has, in addition to the commands $r_i^+ \Rightarrow j$ and $r_i^- \Rightarrow j, k$ of section 1.1, also commands of the form

$$r_i = F(r_j) \Rightarrow k$$

which are read as follows: replace the content of the i-th register by $F(r_j)$ (leaving all other register contents unchanged) and move to the k-th command.

If we code the command $r_i = F(r_j) \Rightarrow k$ by $\langle i, j, k, 0\rangle$ then we can rewrite section 2.4 as follows:

> There is a primitive recursive set Prog' of codes of programs for the RM with oracle: $e \in \mathrm{Prog}'$ if and only if
>
> $$\mathrm{lh}(e) > 0 \wedge \forall i < \mathrm{lh}(e)[\mathrm{lh}((e)_i) \in \{2,3\} \vee (\mathrm{lh}((e)_i) = 4 \wedge ((e)_i)_3 = 0)]$$
>
> By $\phi^F_{m,e}$ we denote the m-ary partial function which is computed by the program with code e and with oracle F.

There is a ternary partial function Φ^F defined by

$$\Phi^F(m,e,x) \simeq \begin{cases} \text{undefined} & \text{if } \neg\text{Prog}'(e) \\ \phi^F_{m,e}(j^m_1(x),\ldots,j^m_m(x)) & \text{otherwise} \end{cases}$$

In full analogy to 2.3.1 we have:

Proposition 5.1.4 *Φ^F is partial recursive in F. Moreover, for every k-ary partial function G which is partial recursive in F there is a number e such that*

$$G(x_1,\ldots,x_k) \simeq \Phi^F(k,e,j^k(x_1,\ldots,x_k))$$

for all $x_1,\ldots,x_k$.

The proof constructs a relation $T^F(m,e,j^m(x_1,\ldots,x_m),y)$ which holds if and only if y codes a computation on the RM with oracle F, with a program with code e and input $x_1,\ldots,x_m$.

The relation T^F is not simply primitive recursive, but primitive recursive *in* F, because in order to check whether it holds, we must verify that the commands $r_i = F(r_j) \Rightarrow k$ have been correctly executed.

The *output function* $U(y) = ((y)_{\text{lh}(y)-1})_1$ remains the same, and is primitive recursive.

Corollary 5.1.5 *The class of partial functions of the form $\phi^F_{m,e}$ coincides with the class of functions partial recursive in F.*

We also have the *Smn*-theorem and the Recursion Theorem in this more general context:

Proposition 5.1.6 (*Smn*-theorem) *There are primitive recursive functions S^m_n, satisfying*

$$\phi^F_{n,S^m_n(e,x_1,\ldots,x_m)}(y_1,\ldots,y_n) \simeq \phi^F_{m+n,e}(x_1,\ldots,x_m,y_1,\ldots,y_n)$$

Proposition 5.1.7 (Recursion Theorem, primitive-recursively uniform version) *There exists a primitive recursive function T satisfying, for all f and $x_1,\ldots,x_m$:*

$$\phi^F_{m,T(f)}(x_1,\ldots,x_m) \simeq \phi^F_{m+1,f}(x_1,\ldots,x_m,T(f))$$

The reason that the functions S^m_n (and the function T in the Recursion Theorem) are primitive recursive (you might have expected: primitive recursive in F) is that they manipulate codes of programs and don't act on computations; and the code for a program for the RM with oracle F does not depend on F. And the function T in the Recursion Theorem is a composition of *Smn*-functions (see the proof of theorem 2.4.3).

Exercise 82 Use the fact (which is known as the *Use Principle*) that every computation of $\phi^F_{m,e}(x_1,\ldots,x_m)$ uses only finitely many values of F, in order to show that the partial function

$$\Phi_{e,x_1,\ldots,x_m} : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$$

defined by $\Phi_{e,x_1,\ldots,x_m}(F) \simeq \phi^F_{m,e}(x_1,\ldots,x_m)$, is continuous on its domain (where we give $\mathbb{N}^{\mathbb{N}}$ the same topology as in section 3.3).

When one defines functions partial recursive in F one writes the same sort of equations as with ordinary partial recursive functions; the only difference is that one can use F as if it were recursive.

The following consideration is also useful: define the relation

$$T^\sigma(m,e,j^m(x_1,\ldots,x_m),y)$$

as: y codes a computation "with oracle σ" for a coded finite sequence σ; that is, every time a command $r_i = F(r_j)$ appears we have $r_j < \text{lh}(\sigma)$ and r_i is replaced by $(\sigma)_{r_j}$.

This relation is of course primitive recursive. Now in any computation on the RM with oracle F we can separate the partial recursive part from the oracle:

Proposition 5.1.8 *The statement* $\phi^F_{m,e}(x_1, \ldots, x_m) = y$ *is equivalent to the statement*

$$\exists \sigma[\forall i < \mathrm{lh}(\sigma)(\sigma)_i = F(i) \wedge \exists w(T^\sigma(m, e, j^m(x_1, \ldots, x_m), w) \wedge U(w) = y)]$$

So apart from the quantifier $\exists \sigma$ we have a conjunction of something primitive recursive in F and something recursively enumerable.

Definition 5.1.9 For total functions F and G we say that F is *Turing reducible to* G, and we write $F \leq_T G$, if F is recursive in G. For subsets A and B of $\mathbb{N}$ ($\mathbb{N}^k$) we say that A is Turing reducible to B (and write $A \leq_T B$) if $\chi_A \leq_T \chi_B$.

Exercise 83 Prove that the relation $\leq_T$ is a preorder on $\mathbb{N}^{\mathbb{N}}$.

Exercise 84 Prove:

$$\text{If } A \leq_m B \text{ then } A \leq_T B$$
$$(\mathbb{N} - A) \leq_T A$$
$$A \leq_T B \text{ does not imply } A \leq_m B$$

Just as with $\leq_m$, we can divide $\mathcal{P}(\mathbb{N})$ in $\equiv_T$-equivalence classes. The $\equiv_T$-class of a subset $A \subseteq \mathbb{N}$ is called the *degree of unsolvability* (or *Turing degree*) of A.

Exercise 85 Prove that every pair $A, B \subseteq \mathbb{N}$ has a join w.r.t. $\leq_T$.

Exercise 86 Prove that the following does *not* hold: if $A \leq_T B$ and B is r.e., then so is A.

Exercise 87 Given sets A and B, prove that the following assertions are equivalent:

i) $B \leq_T A$

ii) There exist total recursive functions F and G such that the following holds:

$$\begin{array}{lll} x \in B & \text{if and only if} & \exists \sigma(\sigma \in W_{F(x)} \wedge \forall i < \mathrm{lh}(\sigma)(\sigma)_i = \chi_A(i)) \\ x \notin B & \text{if and only if} & \exists \sigma(\sigma \in W_{G(x)} \wedge \forall i < \mathrm{lh}(\sigma)(\sigma)_i = \chi_A(i)) \end{array}$$

(Hint: use proposition 5.1.8)

5.2 Sets r.e. in F; the jump operation

Also a considerable part of Chapter 3 can be rewritten in the context of "recursive in F". In particular, we have:

Proposition 5.2.1 *Let* $\mathcal{K}^F = \{e \mid \phi^F_{1,e}(e)$ *is defined*$\}$. *Then* $\mathcal{K}^F$ *is not recursive in* F.

The proof is analogous to the standard case.

Definition 5.2.2 A set A is called *r.e. in* F if $A = \mathrm{dom}(\phi^F_{1,e})$ for some e (equivalently: one of the other conditions in proposition 3.2.2, suitably relativized to F). The set A is called r.e. in $B \subseteq \mathbb{N}$ if A is r.e. in χ_B.

Definition 5.2.3 For a set A, the set $\mathcal{K}^{\chi_A}$ is called the *jump* of A; we write A' for this set.

The operation $A \mapsto A'$ preserves $\equiv_T$-equivalence, as we shall see in proposition 5.2.4 below, and induces therefore an operation on $\equiv_T$-equivalence classes. It is a tool for the study of the partial order on Turing degrees.

Proposition 5.2.4 (Jump Theorem)

i) For every set A, A' *is r.e. in* A

ii) A' *is not recursive in* A

iii) B *r.e. in* A *if and only if* $B \leq_m A'$

iv) If A *is r.e. in* B *and* $B \leq_T C$ *then* A *is r.e. in* C

v) $B \leq_T A$ *if and only if* $B' \leq_m A'$

vi) A *is r.e. in* B *if and only if* A *is r.e. in* $\mathbb{N} - B$

Proof. i) is evident.
ii) This is proposition 5.2.1.
iii) 'Only if' follows by a suitable relativization of the statement that $\mathcal{K}$ is m-complete in Σ_1, and 'if' follows from i) and the statement that if A is r.e. in C and $B \leq_m A$, then B is r.e. in C.
iv) If A is of the form $\{x \mid \exists y P(x,y)\}$ with P recursive in B, then P is also recursive in C, so A is r.e. in C.
v) 'Only if' follows from iii), for $B \leq_T A$ implies that B' is r.e. in A. For the 'if'-direction, relativize proposition 3.2.5i) to: $B \leq_T A$ holds precisely if both B and $\mathbb{N} - B$ are r.e. in A. From iii) we can deduce that $B \leq_m B'$ and $(\mathbb{N} - B) \leq_m B'$. We conclude: if $B' \leq_m A'$ then B is r.e. in A and $\mathbb{N} - B$ is r.e. in A, so $B \leq_T A$.
vi) This follows from Exercise 84. ∎

Proposition 5.2.4 not only connects the notions $\leq_m$ and $\leq_T$; we also have as a direct consequence of v) and Exercise 84 that $B \leq_T A$ implies $B' \leq_T A'$ (hence also: if $B \equiv_T A$ then $B' \equiv_T A'$). As stated before proposition 5.2.4, the mapping $(-)'$ is well-defined on Turing degrees, and is *order-preserving*

We shall indicate Turing degrees by lower-case Greek letters: $\alpha, \beta, \ldots$

The $\equiv_T$-class of $\emptyset$ is written $\mathbb{O}$; this class consists precisely of the recursive sets. It is obvious that $\mathbb{O}$ is the least element in the ordering of Turing degrees. The structure of the partial ordering of $\equiv_T$-classes is quite complicated; much advanced Computability Theory (so-called *Degree Theory*) studies this partial order.

Exercise 88 Prove that $\mathcal{K}' \in \Sigma_2$.

If we write $A^{(n)}$ for the n-fold jump of A, then the following holds in general: $\emptyset^{(n)}$ is m-complete in Σ_n.

In the early 1940's, E. Post whether every r.e. set is either recursive, or $\equiv_T$-equivalent to $\mathcal{K}$; in other words, whether every r.e. set belongs to $\mathbb{O} \cup \mathbb{O}'$.

For some time this was referred to as *Post's Problem*. In 1956/57, Friedberg and Muchnik independently answered this question in the negative: there exists an r.e. set A such that $\emptyset <_T A <_T \mathcal{K}$ (here $<_T$ means: $\leq_T$ and $\not\equiv_T$). There also exist r.e. sets A and B whose degrees are incomparable w.r.t. $\leq_T$. For such A and B we must then also have $\emptyset <_T A <_T \mathcal{K}$ and $\emptyset <_T B <_T \mathcal{K}$.

Exercise 89 Prove this last sentence.

Since we know that $A \leq_m B$ implies $A \leq_T B$, this gives us another proof that there exist r.e. sets which are neither recursive, nor m-complete in Σ_1. However, the Friedberg-Muchnik result is substantially stronger (and harder) than Theorem 4.3.8.

We also conclude from the Friedberg-Muchnik result that there exist Turing degrees α which are not of the form β': the jump operation is not surjective.

We conclude this section with a theorem which says that the jump operation *is* surjective on degrees $\geq \mathbb{O}'$:

Theorem 5.2.5 (Friedberg) *Suppose* $\beta \geq \mathbb{O}'$. *Then there is an* α *satisfying* $\alpha' = \alpha \sqcup \mathbb{O}' = \beta$.

Proof. Choose $B \in \beta$. Because $\beta \geq \mathbb{O}'$ every r.e. set is recursive in B. We now define a B-recursive sequence of coded sequences $(f_s | s \in \mathbb{N})$, such that, whenever $s \leq t$, then the sequence coded by f_s is an initial segment of the sequence coded by f_t (which we denote by $f_s \preceq f_t$). This is done as follows: $f_0 = \langle\rangle$, the code of the empty sequence. Suppose f_s has been defined.

- If s is even, $s = 2e$, determine whether

$$\exists \sigma \exists t [f_s \preceq \sigma \wedge T^\sigma(1, e, e, t) \wedge \forall i < \mathrm{lh}(\sigma)(\sigma)_i \in \{0, 1\}] \quad (*)$$

  If there exists a σ satisfying $(*)$, let f_{s+1} be the minimal such σ; otherwise, put $f_{s+1} = f_s$.

- If s is odd, $s = 2e + 1$, put $f_{s+1} = f_s \star \langle \chi_B(e) \rangle$.

Observe that $f_s \preceq f_{s+1}$ always holds and that the length of f_s increases at every odd step, so the collection of all f_s determines a total function f (which takes values in $\{0, 1\}$).

Because condition $(*)$ is r.e. and therefore recursive in $\mathbb{O}'$, hence recursive in B, and at odd steps we use only χ_B, the resulting function f is recursive in B.

Let $A = \{x \mid f(x) = 0\}$ and α the $\equiv_T$-class of A. Then $A \leq_T B$, so the inequality $\alpha \sqcup \mathbb{O}' \leq_T \beta$ is clear. We also always have $\alpha \sqcup \mathbb{O}' \leq_T \alpha'$ (as you can check for yourself). Therefore, it remains to show:

1) $\alpha' \leq_T \beta$

2) $\beta \leq_T \alpha \sqcup \mathbb{O}'$

For 1): in order to decide whether $e \in A' = \{e \mid \phi^A_{1,e}(e) \text{ is defined}\}$, determine whether $(*)$ holds for $s = 2e$. If $(*)$ holds, then there is a minimal 0-1 sequence σ such that $f_{2e} \preceq \sigma$ and $T^\sigma(1, e, e, t)$ for some t; but then we have $e \in A'$ by definition.

If $(*)$ does not hold for $s = 2e$ then clearly $e \notin A'$. Because condition $(*)$ is recursive in $\mathbb{O}'$ and $(f_s)_{s \geq 0}$ is recursive in B, it follows that $A' \leq_T B$, hence $\alpha' \leq \beta$.

For 2): $\chi_B(e)$ is the last element of the sequence coded by f_{2e+2}, so B is recursive in the sequence $(f_s | s \in \mathbb{N})$.

Now f_{2e+1} is obtained recursively in $\mathbb{O}'$ from f_{2e}, and f_{2e+2} is obtained recursively in A from f_{2e+1} (for $f = \chi_A$, so $f_{2e+2} = f_{2e+1} \star \langle \chi_A(\mathrm{lh}(f_{2e+1})) \rangle$), so the enumeration $(f_s | s \in \mathbb{N})$ is recursive in $A \sqcup \emptyset'$, in other words $B \leq_T A \sqcup \emptyset'$, so $\beta \leq \alpha \sqcup \mathbb{O}'$. ∎

5.3 The Relativized Arithmetical Hierrarchy

There are completely straightforward definitions of Σ^F_n, Π^F_n: in the definition of Σ_n and Π_n, replace 'recursive' by 'recursive in F', and the T-predicate by T^F.

All theorems of section 4.2 generalize: we have Kleene Normaal Forms for Σ^F_n, which are m-complete in Σ^F_n; the hierarchy does not collapse.

We call a set A *arithmetical in* F if $A \in \bigcup_{n \in \mathbb{N}} \Sigma^F_n$. We can represent the subsets of $\mathbb{N}$ which are arithmetical in F, also in the following way: augment the language of arithmetic or ordered rings (see section 2.1.2) by a new function symbol $\overline{F}$, and make the standard model $\mathcal{N}$ into a structure for the new language by interpreting $\overline{F}$ as F. Again we have: A is arithmetical in F precisely if there exists a formula $\varphi(x)$ in one free variable in the extended language, such that

$$A = \{n \in \mathbb{N} \mid \mathcal{N} \models \varphi(\overline{n})\}$$

One last exercise:

Exercise 90 Prove that for all $n \geq 1$:

$$X \in \Sigma^Y_n \text{ if and only if } X \leq_m Y^{(n)}$$

Hint: use induction.

Chapter 6

A Glimpse Beyond the Arithmetical Hierarchy

In this final chapter of these notes, we shall have a look at sets which can be defined in "second-order logic"; that is, logic which has, in addition to variables for numbers (and quantifiers over these variables) also variables and quantifiers for functions.

We have a hierarchy which extends the Arithmetical Hierarchy and which is called the *Analytical Hierarchy* because traditionally, logicians refer to second order arithmetic as "analysis".

6.1 Partial Recursive Functionals

We extend the concept of an *RM with oracle* of definition 5.1.3 to one with multiple oracles $F_1, \ldots, F_k$: we have extra commands

$$r_i = F_m(r_j) \Rightarrow n \quad (1 \leq m \leq k)$$

whose intended meaning is: replace r_i by $F_m(r_j)$ and move to command n.
We can code this command as $\langle i, j, n, k\rangle$ and again, we have a primitive recursive set of codes of programs, and we have partial functions

$$\phi_{l,e}^{F_1,\ldots,F_m}(x_1, \ldots, x_l)$$

which we call *partial recursive in* $F_1, \ldots, F_m$. This extension is completely straightforward; what is new in this chapter is that we regard the expression $\phi_{l,e}^{F_1,\ldots,F_m}(x_1, \ldots, x_l)$ as a partial function also *of the variables* $F_1, \ldots, F_m$.

Definition 6.1.1 A *partial recursive* k, l*-ary functional* is a partial function $(\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N}^l \to \mathbb{N}$ of the form

$$(\vec{F}, \vec{x}) \mapsto \phi_{l,e}^{F_1,\ldots,F_k}(x_1, \ldots, x_l)$$

A partial recursive k, l-ary functional into $\mathbb{N}^{\mathbb{N}}$ is a partial function $\Phi : (\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N}^l \to \mathbb{N}^{\mathbb{N}}$ such that for some partial recursive functional $\Psi : (\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N}^{l+1} \to \mathbb{N}$ we have:

a) $\Phi(\vec{F}, \vec{x})$ is defined if and only if for all y, $\Psi(\vec{F}, \vec{x}, y)$ is defined, and

b) If $\Phi(\vec{F}, \vec{x})$ is defined, then for all y we have

$$\Phi(\vec{F}, \vec{x})(y) = \Psi(\vec{F}, \vec{x}, y)$$

The word "functional" is often used for a function some of whose arguments are functions.
Without developing an elaborate theory of these functionals, we state the following (which should be intuitively clear) without proof:

Proposition 6.1.2 *The partial recursive functionals enjoy the following closure under composition: if*

$$\begin{aligned}\Psi_1,\ldots,\Psi_k &: (\mathbb{N}^{\mathbb{N}})^m \times \mathbb{N}^n \to \mathbb{N}^{\mathbb{N}}\\ \Phi_1,\ldots,\Phi_l &: (\mathbb{N}^{\mathbb{N}})^m \times \mathbb{N}^n \to \mathbb{N}\\ \Xi &: (\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N}^l \to \mathbb{N}\end{aligned}$$

are partial recursive functionals and $\Psi_1,\ldots,\Psi_k$ are total, then the map

$$(\vec{F},\vec{x}) \mapsto \Xi(\Psi_1(\vec{F},\vec{x}),\ldots,\Psi_k(\vec{F},\vec{x}),\Phi_1(\vec{F},\vec{x}),\ldots,\Phi_l(\vec{F},\vec{x}))$$

is a partial recursive functional.

Example 6.1.3 Without the requirement that $\Psi_1,\ldots,\Psi_k$ be total in proposition 6.1.2, the result is no longer true. The following example is instructive: let Φ be the partial recursive function

$$\Phi(e,x) \simeq \begin{cases} 0 & \text{if } \forall y \le x \neg T(1,e,e,y)\\ \text{undefined} & \text{otherwise}\end{cases}$$

and let $\Psi : \mathbb{N} \to \mathbb{N}^{\mathbb{N}}$ be the partial recursive functional defined by $\Psi(e) = \lambda x.\Phi(e,x)$. Note that the domain of Ψ is $\{e \mid e \notin \mathcal{K}\}$, so if we compose Ψ with the (total) functional $F \mapsto F(0)$, we get a partial function $\mathbb{N} \to \mathbb{N}$ with domain $\mathbb{N} - \mathcal{K}$. This is not a partial recursive functional.

If we have an expression $\varphi(\vec{F},\vec{x})$ for a number or a function, we say that it is *recursive in* $\vec{F},\vec{x}$ if it is the result of applying a partial recursive functional to $\vec{F},\vec{x}$

Exercise 91 For a function F and number u, define $F_u(x) = F(j(u,x))$. Show that F_u is recursive in F,u.
For F,u let $\langle u\rangle * F$ be the function such that $(\langle u\rangle * F)(0) = u$ and $(\langle u\rangle * F)(n+1) = F(n)$. Show that $\langle u\rangle * F$ is recursive in F,u.
For functions F,G let $j(F,G)$ be the function such that $j(F,G)(n) = j(F(n),G(n))$. Show that $j(F,G)$ is recursive in F,G.

Definition 6.1.4 A subset of $(\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N}^l$ is called *recursive* if its characteristic function is a (total) recursive functional.

From proposition 6.1.2 we immediately derive the following consequence:

Corollary 6.1.5 *If $X \subseteq (\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N}^l$ is recursive and*

$$\begin{aligned}\Psi_1,\ldots,\Psi_k &: (\mathbb{N}^{\mathbb{N}})^m \times \mathbb{N}^n \to \mathbb{N}^{\mathbb{N}}\\ \Phi_1,\ldots,\Phi_l &: (\mathbb{N}^{\mathbb{N}})^m \times \mathbb{N}^n \to \mathbb{N}\end{aligned}$$

are total recursive functionals, then the set

$$\{(\vec{F},\vec{x}) \mid (\Psi_1(\vec{F},\vec{x}),\ldots,\Psi_k(\vec{F},\vec{x}),\Phi_1(\vec{F},\vec{x}),\ldots,\Phi_l(\vec{F},\vec{x})) \in X\}$$

is a recursive subset of $(\mathbb{N}^{\mathbb{N}})^m \times \mathbb{N}^n$.

Exercise 92 Let us say that a function $F : \mathbb{N} \to \mathbb{N}$ is *torsion-free* if for all $n > 0$ and x, we have $F^n(x) \neq x$ ($F^n(x)$ is F applied n times).
Show that the set $\{F \mid F \text{ is torsion-free}\}$ is not recursive. Hint: use Exercise 82.

6.2 The Analytical Hierarchy

We repeat definition 4.2.1 for the context of sets of functions and numbers.

Definition 6.2.1 Let $n \geq 1$. A subset $X \subseteq (\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N}^l$ is in Σ^0_n if there is a recursive subset Y of $(\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N}^{l+n}$ such that:

either n is odd and

$$X = \{(\vec{F}, \vec{x}) \,|\, \exists y_1 \forall y_2 \cdots \forall y_{n-1} \exists y_n \, (\vec{F}, \vec{x}, \vec{y}) \in Y\}$$

or n is even and

$$X = \{(\vec{F}, \vec{x}) \,|\, \exists y_1 \forall y_2 \cdots \exists y_{n-1} \forall y_n \, (\vec{F}, \vec{x}, \vec{y}) \in Y\}$$

A set is in Π^0_n if its complement is in Σ^0_n; and a set is in Δ^0_n if it is both in Σ^0_n and in Π^0_n.
A set is *arithmetical* if it is in Σ^0_n for some n.

Exercise 93

i) Show that X is in Δ^0_1 if and only if X is recursive.

ii) Show that the set $\{F \,|\, F \text{ is torsion-free}\}$ of Exercise 92 is in Π^0_1.

iii) Show that the set $\{F \,|\, F \text{ is recursive}\}$ is in Σ^0_3.

iv) Show that the set $\{F \,|\, \mathrm{rge}(F) \text{ is infinite}\}$ is in Π^0_2.

Definition 6.2.2 (Analytical Hierarchy) A subset $X \subseteq (\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N}^l$ is in Σ^1_0 if it is arithmetical. We put $\Sigma^1_0 = \Pi^1_0 = \Delta^1_0$.
A set $X \subseteq (\mathbb{N}^{\mathbb{N}})^k \times \mathbb{N}^l$ is in Σ^1_{n+1} if there is a subset $Y \subseteq (\mathbb{N}^{\mathbb{N}})^{k+1} \times \mathbb{N}^l$ such that Y is in Π^1_n and

$$X = \{(\vec{F}, \vec{x}) \,|\, \exists G \, (\vec{F}, G, \vec{x}) \in Y\}$$

A set is in Π^1_{n+1} if its complement is in Σ^1_{n+1}, and it is in Δ^1_{n+1} if it is both in Σ^1_{n+1} and in Π^1_{n+1}.

The hierarchy of classes Σ^1_n, Π^1_n (the *Analytical Hierarchy*) is very much in analogy with the Arithmetical Hierarchy. We have the inclusions $\Sigma^1_n \subset \Delta^1_{n+1}$ which are strict (so the hierarchy does not collapse); we have that if X is in Σ^1_n and $Y \leq_m X$ then Y is in Σ^1_n; we have m-complete sets at every level.

In this chapter, we limit the discussion to the levels Π^1_1 and Δ^1_1.
The *Tarski-Kuratowski algorithm* is a bit less trivial than in the first-order (arithmetical) case. We have the following rewriting rules.

Proposition 6.2.3 (Tarski-Kuratowski for sets of functions)

a) *The set* $\{(\vec{F}, \vec{x}) \,|\, \forall y_1 \cdots y_n \exists u \, (\vec{F}, \vec{x}, \vec{y}, u) \in X\}$ *is equal to the set*

$$\{(\vec{F}, \vec{x}) \,|\, \exists G \forall y_1 \cdots y_n \, (\vec{F}, \vec{x}, \vec{y}, G(j^n(y_1, \ldots, y_n))) \in X\}$$

(Here X is an arbitrary set)

b) *The set* $\{(\vec{F}, \vec{x}) \,|\, \forall G, H \, (\vec{F}, G, H, \vec{x}) \in X$ *is equal to the set*

$$\{(\vec{F}, \vec{x}) \,|\, \forall K \, (\vec{F}, \lambda u.j_1(K(u)), \lambda u.j_2(K(u)), \vec{x}) \in X\}$$

c) *The set* $\{(\vec{F}, \vec{x}) \,|\, \forall y \exists G \, (\vec{F}, G, \vec{x}, y) \in X\}$ *is equal to the set*

$$\{(\vec{F}, \vec{x}) \,|\, \exists H \forall y \, (\vec{F}, H_y, \vec{x}, y) \in X\}$$

where the notation H_y is as in Exercise 91

d) *The set* $\{(\vec{F},\vec{x})\,|\,\forall y\forall G\,(\vec{F},G,\vec{x},y)\in X\}$ *is equal to the set*

$$\{(\vec{F},\vec{x})\,|\,\forall G\,(\vec{F},\lambda x.G(x+1),\vec{x},G(0))\in X\}$$

We note that all the expressions used in the rewritings of proposition 6.2.3 are recursive in their arguments. Therefore, we can conclude from b) of 6.2.3 that if a set X is in Π^1_n, the set

$$\{(\vec{F},\vec{x})\,|\,\forall G\,(\vec{F},G,\vec{x})\in X\}$$

is also in Π^1_n.
From c) of 6.2.3 we conclude that if X is in Σ^1_n, the set

$$\{(\vec{F},\vec{x})\,|\,\forall y\,(\vec{F},\vec{x},y)\in X\}$$

is in Σ^1_n, too; therefore, by taking complements, that if X is in Π^1_n, so is the set

$$\{(\vec{F},\vec{x})\,|\,\exists y\,(\vec{F},\vec{x},y)\in X\}$$

Proposition 6.2.4 *Every* Π^1_1*-set is of the form*

$$\{(\vec{F},\vec{x})\,|\,\forall G\exists y\,(\vec{F},G,\vec{x},y)\in X\}$$

with X *recursive. Moreover, the* Π^1_1*-sets are closed under numerical quantification* $\exists y,\forall y$*.*

Proof. Let Y be a Π^1_1-set. By definition, Y has the form

$$\{(\vec{F},\vec{x})\,|\,\forall G\,(\vec{F},G,\vec{x})\in X\}$$

with X arithmetical; we employ induction on the level of X in the arithmetical hierarchy.
If X is in Σ^0_1, we are done. If X is in Π^0_1 then

$$Y=\{(\vec{F},\vec{x})\,|\,\forall G\forall y\,(\vec{F},\vec{x},y)\in X'\}$$

for some recursive X'; we use rewriting d) of 6.2.3 to contract the quantifier combination $\forall G\forall y$ (which is equivalent to $\forall y\forall G$) to one quantifier $\forall G$ and we are done.

Now suppose X is in Σ^0_n with $n\geq 2$. So

$$Y\;=\;\{(\vec{F},\vec{x})\,|\,\forall G\exists y_1\forall y_2\,(\vec{F},\vec{x},y_1,y_2)\in X'\}$$

with X' in Σ^0_{n-2}. We use rewriting a) of 6.2.3, taking complements, in order to rewrite the quantifier combination $\exists y_1\forall y_2$ to $\forall H\exists y_1$:

$$Y\;=\;\{(\vec{F},\vec{x})\,|\,\forall G\forall H\exists y_1\,(\vec{F},\vec{x},y_1,H(y_1))\in X'\}$$

Then we contract the two function quantifiers into one, using b) of 6.2.3, and apply the induction hypothesis.

The second statement is a direct application of rewritings c) and d) (and taking complements) of 6.2.3. ∎

6.3 Well-founded trees: an m-complete Π^1_1-set of numbers

We modify the definition (in section 3.2.1) of a *tree*: a tree is now a set of *coded* sequences closed under initial segments. Also, recall from there the notion of a *path through* a tree: this is a function F such that for every n, $\langle F(0),\ldots,F(n-1)\rangle$ is an element of the tree.

Definition 6.3.1 A tree T is *well-founded* if there is no path through T.

Definition 6.3.2 *Define the set WfRec by*

$$\mathit{WfRec} \;=\; \{x \,|\, \phi_x \text{ is the characteristic function of a well-founded tree}\}$$

Theorem 6.3.3 *The set* WfRec *is in* Π^1_1 *and every* Π^1_1*-subset of* $\mathbb{N}$ *is many-one reducible to* WfRec.

Proof. The statement $x \in \mathrm{WfRec}$ is equivalent to the conjunction of the following three assertions:

ϕ_x is total and takes values in $\{0,1\}$

$\forall\sigma\tau($ if $\phi_x(\sigma) = 0$ and $\tau \preceq \sigma$ then $\phi_x(\tau) = 0)$

(where $\tau \preceq \sigma$ abbreviates $\mathrm{lh}(\tau) \leq \mathrm{lh}(\sigma) \wedge \forall i < \mathrm{lh}(\tau)\,(\tau)_i = (\sigma)_i$))

$\forall F \exists n\,(\phi_x(\langle F(0), \ldots, F(n-1)\rangle) \neq 0)$

You can check for yourself that the first of these statements is Π^0_2 and the second is Π^0_1. The third statement is Π^1_1. That the conjunction is in Π^1_1 now follows by the rewritings of 6.2.3. This proves the first statement.

Now let $X \subseteq \mathbb{N}$ be an arbitrary Π^1_1-set. By 6.2.4, we may assume that

$$X \;=\; \{n \in \mathbb{N} \,|\, \forall F \exists y\,(F, n, y) \in P\}$$

where $P \subseteq \mathbb{N}^{\mathbb{N}} \times \mathbb{N}^2$ is a recursive subset. Since the characteristic function of P is a recursive functional, we may write X as

$$\{n \,|\, \forall F \exists y w (T^F(2, e, j(n,y), w) \wedge U(w) = 0)\}$$

for some number e.
By proposition 5.1.8, the condition $T^F(2, e, j(n,y), w) \wedge U(w) = 0$ is equivalent to

$$\exists\sigma[(\forall i < \mathrm{lh}(\sigma)\,(\sigma)_i = F(i)) \wedge T^\sigma(2, e, j(n,y), w) \wedge U(w) = 0]$$

Now clearly, if $T^\sigma(2, e, j(n,y), w)$ holds, then $T^\tau(2, e, j(n,y), w)$ will also hold whenever the sequence coded by σ is an initial segment of the sequence coded by τ. Hence, by taking larger and larger initial segments of F we see that the statement

$$\forall F \exists y w (T^F(2, e, j(n,y), w) \wedge U(w) = 0)$$

is equivalent to

$$\forall F \exists\sigma[(\forall i < \mathrm{lh}(\sigma)(\sigma)_i = F(i)) \wedge \exists y w < \mathrm{lh}(\sigma)(T^\sigma(2, e, j(n,y), w) \wedge U(w) = 0)]$$

in other words, to

$$\forall F \exists\sigma[(\forall i < \mathrm{lh}(\sigma)(\sigma)_i = F(i)) \wedge Q(\sigma, n)]$$

where Q is some *recursive* set. Moreover, it is clear that if $Q(\sigma, n)$ fails to hold, then $Q(\tau, n)$ will also fail to hold whenever τ codes an initial segment of the sequence coded by σ. Therefore, for every n,

$$\{\sigma \,|\, \neg Q(\sigma, n)\}$$

is a recursive tree. Clearly, this tree is recursively obtained from the original set P and has therefore a characteristic function whose index is $H(e, n)$ for some primitive recursive function H. Finally, we have $H(e, n) \in \mathrm{WfRec}$ if and only if $n \in X$ (as I leave for you to check), so the function $n \mapsto H(e, n)$ reduces X to WfRec. ∎

6.4 Hyperarithmetical Sets and Functions

Hyperarithmetical is another word for: being in Δ^1_1. A function $f : \mathbb{N} \to \mathbb{N}$ is hyperarithmetical if its graph (the set $\{\langle x, y\rangle \mid f(x) = y\}$) is a hyperarithmetical set.
The level Δ^1_1 is the lowest non-arithmetical level. In this section we shall see: a set which is hyperarithmetical, but not arithmetical; a structure theorem (the famous Suslin-Kleene Theorem below) for Δ^1_1-sets and an analogon of Theorem 3.2.10 for hyperarithmetical functions.

For our example of a set in $\Delta^1_1 - \Sigma^1_0$, we take the set of Theorem 4.3.10.

Theorem 6.4.1 *The set of codes of true sentences in the language of arithmetic is in* Δ^1_1.

Proof. Note, that in 4.3.10 we already saw that this set is not arithmetical.
Although the details of coding are irrelevant (and certainly, by now you should be able to define such a coding yourself), for the purpose of the proof it is good to have a specific coding in mind. So let us assume that the sentences of arithmetic are built up from: constants 0,1; function symbols $+, \times$; a relation symbol $<$; the equality sign $=$; variables $(v_i)_{i\geq 0}$; and logical symbols $\neg, \vee, \exists$. We set up a dictionary:

0	1	+	×	<	=	v_i	¬	∨	∃
0	1	2	3	4	5	6	7	8	9

We define codes of terms: the constant 0 gets code $\ulcorner 0 \urcorner = \langle 0\rangle$; $\ulcorner 1 \urcorner = \langle 1\rangle$; $\ulcorner v_i \urcorner = \langle 6, i\rangle$; if t and s are terms then $\ulcorner t + s \urcorner = \langle 2, \ulcorner t \urcorner, \ulcorner s \urcorner\rangle$ and $\ulcorner t \times s \urcorner = \langle 3, \ulcorner t \urcorner, \ulcorner s \urcorner\rangle$.
Next we define codes of formulas: if t and s are terms then $\ulcorner t = s \urcorner = \langle 5, \ulcorner t \urcorner, \ulcorner s \urcorner\rangle$ and $\ulcorner t < s \urcorner = \langle 4, \ulcorner t \urcorner, \ulcorner s \urcorner\rangle$. If φ and ψ are formulas then $\ulcorner \varphi \vee \psi \urcorner = \langle 8, \ulcorner \varphi \urcorner, \ulcorner \psi \urcorner\rangle$, $\ulcorner \neg \varphi \urcorner = \langle 7, \ulcorner \varphi \urcorner\rangle$ and $\ulcorner \exists v_i \varphi \urcorner = \langle 9, i, \ulcorner \varphi \urcorner\rangle$.
You can check for yourself that properties like "x is the code of a term", "x is the code of an atomic formula", "x is the code of a sentence", etcetera, are all primitive recursive. We also have a primitive recursive function $S(x, i, n)$ such that whenever $x = \ulcorner \varphi \urcorner$ then $S(x, i, n) = \ulcorner \varphi[\overline{n}/v_i] \urcorner$ (the code of the formula resulting from substituting the term $\overline{n}$ for v_i in φ). Moreover, from a code of a closed term we can primitive-recursively obtain its interpretation in the standard model $\mathbb{N}$ and therefore we can primitive-recursively check whether an atomic sentence is true in $\mathbb{N}$.
Now let P be the set of those functions F satisfying the following conditions:

> For all x, $F(x) \leq 1$ if and only if x is the code of a sentence
>
> For all x, if x is de code of an atomic sentence φ then $F(x) = 0$ if and only if φ is true in $\mathbb{N}$
>
> For all x, if $x = \langle 7, y\rangle$ and $F(y) \leq 1$ then $F(x) = 1 - F(y)$
>
> for all x, if $x = \langle 8, y, z\rangle$ and $F(y), F(z) \leq 1$ then $F(x) = F(y) \times F(z)$
>
> for all x, if $x = \langle 9, i, y\rangle$ and y is the code of a formula with at most the variable v_i free, then $F(x) = 0$ if and only if for some n, $F(S(y, i, n)) = 0$

We see that P is arithmetical; in fact, P is a Σ^0_2-set. Moreover, the following statements are equivalent:

> The number x is the code of a true sentence in the language of arithmetic
>
> For all $F \in P$, $F(x) = 0$
>
> There exists an $F \in P$ such that $F(x) = 0$

This shows that our set is in Δ^1_1, as claimed. ∎

Without proof, I now give two theorems about hyperarithmetical sets and functions. The first is an analogue of our Theorem 3.2.10, stating that there exists a recursive, infinite, finitely branching tree which does not have a recursive path.

Theorem 6.4.2 *There exists a recursive, infinite, finitely branching tree which does not have a hyperarithmetical path.*

Our second theorem allows one sometimes to prove that all hyperarithmetical sets have a certain property.

Definition 6.4.3 *Let $C \subset \mathbb{N}$ be a set and let for every $e \in C$ a set $C_e \subseteq \mathbb{N}$ be given. The system $\mathcal{C} = \{C_e \mid e \in C\}$ is called an* SK-class *([26]) or an* effective σ-ring *([25]) if there exist partial recursive functions ι, μ, ϵ satisfying the following conditions:*

For all n, $\iota(n)$ is defined and $\iota(n) \in C$ and $C_{\iota(n)} = \{n\}$

For all $e \in C$, $\mu(e)$ is defined and $\mu(e) \in C$ and $C_{\mu(e)} = \mathbb{N} - C_e$

For every index e of a total recursive function ϕ_e such that $\mathrm{rge}(\phi_e) \subseteq C$, *$\epsilon(e)$ is defined and $\epsilon(e) \in C$ and $C_{\epsilon(e)} = \bigcup_n C_{\phi_e(n)}$*

Exercise 94

i) Prove that every SK-class $\mathcal{C} = \{C_e \mid e \in C\}$ contains every arithmetical subset of $\mathbb{N}$. Hint: use induction and the Kleene Normal Form Theorem 4.2.8.

ii) Let $\mathcal{C} = \{C_e \mid e \in C\}$ be an SK-class. Prove that $\mathcal{C}$ contains a non-arithmetical set. Hint: prove that there is a total recursive function ψ such that for all n, $\psi(n) \in C$ and

$$C_{\psi(n)} = \{\langle n, x, y\rangle \mid (x, y) \in E^{(1)}_n\}$$

where $E^{(1)}_n$ is the standard m-complete Σ_n-subset of $\mathbb{N}^2$ of 4.2.11.

Theorem 6.4.4 (Suslin-Kleene Theorem) *There exists an SK-class $\mathcal{D} = \{D_d \mid d \in D\}$ with partial recursive functions κ, ν, ζ, such that the following hold:*

a) The set $\{D_d \mid d \in D\}$ coincides with the set of all hyperarithmetical subsets of $\mathbb{N}$

b) For every SK-class $\mathcal{C} = \{C_e \mid e \in C\}$ with partial recursive functions ι, μ, ϵ there is a recursive function $\psi : D \to C$ satisfying $\psi(\kappa(n)) = \iota(n)$ for all n, $\psi(\nu(d)) = \mu(\psi(d))$ for all $d \in D$, and $\psi(\zeta(e)) = \epsilon(e')$ for all indices e of total recursive functions into D; here e' is a standard index for the function $n \mapsto \psi(\phi_e(n))$.

Note that Theorem 6.4.4 implies that the collection of hyperarithmetical subsets of $\mathbb{N}$ is contained in every SK-class. Therefore, if $\mathcal{P}$ is a countable collection of subsets of $\mathbb{N}$, one can prove that it contains all hyperarithmetical sets if one can endow $\mathcal{P}$ with the structure of an SK-class.

Bibliography

[1] W. Ackermann. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen*, 99:118–133, 1928.

[2] H.P. Barendregt. *The Lambda Calculus*, volume 103 of *Studies in Logic*. North-Holland, 1984. second edition.

[3] G. Boolos, J.P. Burgess, and R.C. Jeffrey. *Computability and Logic*. Cambridge University Press, 2007. Fifth edition.

[4] Samuel R. Buss. First-order Proof Theory of Arithmetic. In S.R. Buss, editor, *Handbook of Proof Theory*. Elsevier, 1998.

[5] Alonzo Church. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, 58:345–363, 1936. Reprinted in [8].

[6] N.J. Cutland. *Computability*. Cambridge University Press, 1980.

[7] M. Detlefsen. On an Alleged Refutation of Hilbert's Program using Gödel's First Incompleteness Theorem. *Journal of Philosophical Logic*, 19(4):343–377, 1990.

[8] Martin Davis (ed). *The Undecidable - Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Dover, 2004. Reprint of 1965 edition by Raven Press Books.

[9] R. Gandy. The Confluence of Ideas in 1936. In R. Herken, editor, *The Universal Turing Machine, a Half-Century Survey*, pages 55–111. Oxford University Press, 1988.

[10] A. Heyting, editor. *Constructivity in Mathematics*. North-Holland Publishing Company, 1959.

[11] D. Hilbert and W. Ackermann. *Grundzüge der theoretischen Logik*. Springer Verlag, 1928.

[12] D. Hilbert and P. Bernays. *Grundlagen der Mathematik I*. Springer Verlag, 1934.

[13] P. Hinman. *Recursion-Theoretic Hierarchies*, volume 9 of *Perspectives in Mathematical Logic*. Springer, 1978.

[14] A. Hodges. *Alan Turing: the enigma*. Random House, London, 1992.

[15] J.M.E. Hyland. The effective topos. In A.S. Troelstra and D. Van Dalen, editors, *The L.E.J. Brouwer Centenary Symposium*, pages 165–216. North Holland Publishing Company, 1982.

[16] R. Kaye. *Models of Peano Arithmetic*, volume 15 of *Oxford Logic Guides*. Oxford University Press, Oxford, 1991.

[17] A. Kechris and Y. Moschovakis. Recursion in Higher Types. In *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic*. North-Holland, 1977.

[18] S. C. Kleene. Realizability. In *Summaries of Talks presented at the Summer Institute for Symbolic Logic*, pages 100–104. Institute for Defense Analyses, Communications Research Division, Princeton, 1957. Also in *[10]*, pp. 285–289. Errata in *[22]*, page 192.

[19] S.C. Kleene. General Recursive Functions of Natural Numbers. *Math. Annalen*, 112(5):727–742, 1936. Reprinted in [8].

[20] S.C. Kleene. Recursive functionals and quantifiers of finite types I. *Trans. Amer. Math. Soc*, 91, 1959.

[21] S.C. Kleene. Recursive functionals and quantifiers of finite types II. *Trans. Amer. Math. Soc*, 108, 1963.

[22] S.C. Kleene and R.E. Vesley. *The Foundations of Intuitionistic Mathematics, especially in relation to recursive functions*. North-Holland Publishing Company, 1965.

[23] R.C. Lyndon and P.E. Schupp. *Combimatorial Group Theory*. Springer (Classics in Mathematics), 1977. reprinted 2001.

[24] Y. Matyasevich. *Hilbert's Tenth Problem*. MIT Press, 1993.

[25] Y. Moschovakis. *Elementary Induction on Abstract Structures*, volume 77 of *Studies in Logic*. North-Holland, Amsterdam, 1974. Reprinted by Dover, 2008.

[26] P. Odifreddi. *Classical Recursion Theory*, volume 125 of *Studies in Logic*. North-Holland, 1989.

[27] P. Odifreddi. *Classical Recursion Theory II*, volume 143 of *Studies in Logic*. North-Holland, 1999.

[28] B. Poonen. Undecidability in Number Theory. *Notices of the American Mathematical Society*, 55 (3):344–350, 2008.

[29] E.L. Post. Finite Combinatory Processes. Formulation I. *Journal of Symbolic Logic*, 1:103–105, 1936. Reprinted in [8].

[30] M.B. Pour-El and J.Ian Richards. *Computability in Analysis and Physics*. Springer-Verlag, 1989.

[31] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. (reprinted by MIT Press, Cambridge MA, 1987).

[32] H.E. Rose. *Subrecursion – Functions and Hierarchies*. Clarendon Press, 1984.

[33] L.E. Sanchis. *Recursive Functionals*, volume 131 of *Studies in Logic*. North-Holland, 1992.

[34] J.R. Shoenfield. *Mathematical Logic*. Addison-Wesley, 1967. Reprinted by ASL, 2000.

[35] J.R. Shoenfield. *Recursion Theory*, volume 1 of *Lecture Notes in Logic*. Springer, 1993.

[36] C. Smoryński. Hilbert's Programme. *CWI Quarterly*, 1(4):3–59, 1988.

[37] Craig Smoryński. *Logical Number Theory I - An Introduction*. Springer-Verlag, 1991.

[38] Robert I. Soare. *Recursively Enumerable Sets and Degrees*. Perspectives in Mathematical Logic. Springer-Verlag, 1987.

[39] G. Sudan. Sur le nombre transfini ω^ω. *Bulletin Mathématique de la Société Roumaine des Sciences*, 30:11–30, 1927.

[40] A. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings London Math. Soc, ser. 2*, 42:230–265, 1936. reprinted in [8].

[41] J. van Oosten. *Realizability: an Introduction to its Categorical Side*, volume 152 of *Studies in Logic*. North-Holland, 2008.

Index