Recursion Theory

Frank Stephan

January 9, 2025

Recursion theory deals with the fundamental concepts on what subsets of natural numbers (or other famous countable domains) could be defined effectively and how complex the so defined sets are. The basic concept are the recursive and recursively enumerable sets, but the world of sets investigated in recursion theory goes beyond these sets. The notions are linked to Diophantine sets, definability by functions via recursion and Turing machines. Although some of the concepts are very old, it took until Matiyasevich's great result that Diophantine and r.e. sets are the same that the picture was fully understood. This lecture gives an overview on the basic results and proof methods in recursion theory.

Frank Stephan: Rooms S17#07-04 and COM1#03-11
Departments of Mathematics and Computer Science
National University of Singapore
10 Lower Kent Ridge Road,Singapore 119076
Republic of Singapore
Telephone 6516-2759 and 6516-4246
Email fstephan@comp.nus.edu.sg
Homepage http://www.comp.nus.edu.sg/~fstephan/index.html

Thanks. The author wants to thank Chong Chi Tat and Yang Yue for discussions on the lecture "Recursion Theory" and its syllabus. Furthermore, he wants to thank Li Hongyang, Li Yanfang and Ye Nan for proofreading.

Contents

1 Foundations **3**

2 Numberings of partial-recursive functions **14**

3 Rice's Theorem and the Arithmetical Hierarchy **25**

4 Post's Problem and Reducibilities **34**

5 The Theorem of Post and Kleene **43**

6 The Fixed-Point Theorem and DNR Degrees **49**

7 A solution to Post's Problem for r.e. sets **56**

8 Maximal, r-maximal and semirecursive sets **62**

9 Permitting and Infinite Injury Priority Methods **70**

10 The Analytical Hierarchy **81**

11 Algorithmic randomness **87**

12 Inductive inference **105**

1 Foundations

The historical roots of recursion theory date back more than 2000 years, but it was not before 1930 that mathematicians started to formalize the notion of "algorithm" precisely and defined the important concepts of recursive and recursively enumerable sets. Diophantus, a mathematician from Alexandria in Egypt during the Ptolomy dynasty studied equations on natural and integer numbers named by him: namely he investigated polynomials in several variables with integer coefficients and wanted to find solutions in the integers or natural numbers. A famous example is the question whether the following equation has a solution in the natural numbers for any fixed n:

$$(x+1)^n + (y+1)^n = (z+1)^n$$

There are easily found solutions for $n = 1$ and $n = 2$, for example $3 + 4 = 7$ and $3^2+4^2 = 5^2$. Pierre de Fermat stated in notes found after his death the given equation has no solution for $n > 2$ and wrote that he would know an easy and nice proof, but it was nowhere written down. It took more than 350 years and many intermediate results until Andrew Wiles provided finally a proof that the equation above has no solution for any $n > 2$. So Diophantine equations are quite complicated and the study of this field needs a lot of imagination and intuition. Naturally mathematicians were interested in finding good methods to solve such equations. Of course one can search for zeroes by testing one possibility after the other. For example, Euler made in 1769 a conjecture which implied that

$$(v+1)^5 + (w+1)^5 + (x+1)^5 + (y+1)^5 = (z+1)^5$$

has no solution. This conjecture was open until 1966 when Lander and Parkin found the solution

$$27^5 + 84^5 + 110^5 + 133^5 = 144^5.$$

Indeed, the age of computers and the ability to test one by one all possible solutions solved a lot of open questions of this type. Although one can find solutions with a computer, the disadvantage is that the computer would search forever in the case that there is no solution. For this reason, mathematicians considered the exhaustive search strategy never as satisfying and asked, already before the age of computers, whether there would be a better method to determine the solvability of Diophantine equations on the natural numbers. In a famous address to the International Congress of Mathematicians in the year 1900, David Hilbert posed 23 fundamental mathematical problems which should be solved in the twentieth century. Among these was the problem to develop a method to check whether a Diophantine equation has a solution. One can easily define a parameterized version of this problem.

Definition 1.1. A set $A \subseteq \mathbb{N}$ is *Diophantine* iff there are a number $n \in \mathbb{N}$ and polynomials f, g with coefficients in $\mathbb{N}$ such that

$$A = \{x : \exists y_1 \ldots \exists y_n \, [f(x, y_1, \ldots, y_n) = g(x, y_1, \ldots, y_n)]\}.$$

Today it is known that one can fix without loss of generality n to the value 9.

In 1970, Matiyasevich [68, 69] showed that one cannot even solve the parameterized version. It is easily seen that a computer can enumerate the members of such a set A by exhaustive search (if one ignores the constraints of finite memory size and computation time). Matiyasevich showed that also the converse is true: every set given by such a enumeration procedure of an idealized computer can be represented as a Diophantine set. Such sets are called recursively enumerable and Matiyasevich's Theorem gives the easiest way to characterize them. This characterization uses only polynomials over $\mathbb{N}$ and avoids any formalization of machines or recursive functions.

Example 1.2. The sets of even numbers, of odd numbers, of square numbers, of nonsquare numbers and of composite numbers are Diophantine:

$$\begin{aligned}
\text{Even} &= \{x : \exists y_1 \, [x = y_1 + y_1]\}; \\
\text{Odd} &= \{x : \exists y_1 \, [x = y_1 + y_1 + 1]\}; \\
\text{Square} &= \{x : \exists y_1 \, [x = y_1 \cdot y_1]\}; \\
\text{Nonsquare} &= \{x : \exists y_1, y_2, y_3 \, [(y_1^2 + y_2 + 1)^2 + x^2 + (1 + y_2 + y_3)^2 + 4y_1^2 \\
&\qquad = 2x(y_1^2 + y_2 + 1) + 4y_1(1 + y_2 + y_3)]\}; \\
\text{Composite} &= \{x : \exists y_1, y_2 \, [x = (y_1 + 2) \cdot (y_2 + 2)]\}.
\end{aligned}$$

The same holds for the set of prime numbers. A list of Diophantine equations for this set can be found on `http://en.wikipedia.org/wiki/Formula_for_primes` and the definition is much more complicated than those above.

As the characterization is so easy, Diophantine sets are an ideal way to introduce recursively enumerable sets. But unfortunately it is a bit difficult to construct Diophantine sets having desired properties. Indeed, it took approximately 40 years from the initial definition of recursively enumerable sets until Matiyasevich discovered the method how to convert a recursively enumerable set (given by some other description) into the polynomials f, g over $\mathbb{N}$ (or the polynomial $f - g$ which then also has negative integers among its coefficients) to define the Diophantine set. The alternative ways to define recursively enumerable sets provide easier constructions and will now be presented in the upcoming paragraphs.

Recursive functions have this name because one can define them iteratively from

easier functions. So addition and multiplication can be defined in terms of the successor-function mapping x to $S(x) = x + 1$ as follows:

$$\begin{aligned} x + 0 &= x; \\ x + S(y) &= S(x + y); \\ x \cdot 0 &= 0; \\ x \cdot S(y) &= (x \cdot y) + x. \end{aligned}$$

The recursive definition goes over the second parameter and now, together with composition of functions, one can define all polynomials. Indeed, the class of functions which can be defined using recursion and certain basic concepts is the class of primitive recursive functions. Dedekind [22], Gödel [38] and Skolem [102] introduced and used this class for various purposes.

Definition 1.3. The class of *primitive recursive functions* is the smallest class **PR** of functions from $\mathbb{N}^n$ (with parameter $n \in \mathbb{N}$) to $\mathbb{N}$ such that the following conditions are satisfied:

- The function mapping any input in $\mathbb{N}^n$ to some constant m is in **PR**.
- The successor-function S given by $S(x) = x + 1$ is primitive recursive.
- For every n and every $m \in \{1, 2, \ldots, n\}$, the function mapping $(x_1, x_2, \ldots, x_n)$ to x_m is in **PR**.
- If $f : \mathbb{N}^n \to \mathbb{N}$ and $g_1, \ldots, g_n : \mathbb{N}^m \to \mathbb{N}$ are all in **PR**, so is the function mapping $(x_1, x_2, \ldots, x_m)$ to $f(g_1(x_1, x_2, ..., x_m), g_2(x_1, x_2, ..., x_m), \ldots, g_n(x_1, x_2, ..., x_m))$.
- If $g : \mathbb{N}^{n+2} \to \mathbb{N}$ and $h : \mathbb{N}^n \to \mathbb{N}$ are functions in **PR** then there is also a function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ in **PR** with $f(x_1, x_2, \ldots, x_n, 0) = h(x_1, x_2, \ldots, x_n)$ and $f(x_1, x_2, \ldots, x_n, S(x_{n+1})) = g(x_1, x_2, \ldots, x_n, x_{n+1}, f(x_1, x_2, \ldots, x_n, x_{n+1}))$,

The last step defining f from g, h is called primitive recursion.

An example of formal primitive recursion could be done as follows. Let $n = 0$, $h(x) = 0$ and $g(x, y) = x$. Then the predecessor function P being the inverse to S is given by $P(0) = 0$ and $P(S(x)) = x$. So P is exactly the function defined by primitive recursion from g and h. Most functions used in everyday life are primitive recursive: addition, multiplication, substraction (when suitably adapted to natural numbers), exponentiation and, of course, also the Fibonacci numbers given by

$$\begin{aligned} F(0) &= 0; \\ F(1) &= 1; \\ F(n + 2) &= F(n) + F(n + 1). \end{aligned}$$

Note that primitive recursive functions are always total. But the primitive recursive functions do not completely capture the class of functions which can be defined inductively. The most prominent member outside this class is the Ackermann function which is defined as follows:

$$\begin{aligned} A(0,n) &= n+1; \\ A(m+1,0) &= A(m,1); \\ A(m+1,n+1) &= A(m,A(m+1,n)). \end{aligned}$$

For this reason, one has to enrich the primitive recursive functions by permitting unbounded search. This was done by introducing the μ-minimalization which is defined for partial functions f as follows.

Definition 1.4. $\mu y(f(x_1,\ldots,x_n,y)=0)$ takes the value z if $f(x_1,\ldots,x_n,y)$ is defined for all $y \leq z$ and $f(x_1,\ldots,x_n,y) > 0$ for $y < z$ and $f(x_1,\ldots,x_n,z) = 0$. Furthermore, $\mu y(f(x_1,\ldots,x_n,y)=0)$ is undefined if there is no z satisfying these conditions.

This definition can also be adapted to other conditions on the value than "being positive"; the basic principle would still be to test the condition for $y = 0, 1, \ldots$ until it is found to be true for some y and one returns this y, in the case that no y is found or the test incorporates some value $f(x_1,\ldots,x_n,y)$ where this is undefined, the μ-minimalization becomes undefined as well. So μ-minimalization gives rise to partial functions and can also be applied to partial functions.

Definition 1.5. The class of *partial-recursive functions* is the smallest class of partial functions from $\mathbb{N}^n$ to $\mathbb{N}$ which satisfies the conditions given in Definition 1.3 and is in addition closed under μ-minimization. A function is *recursive* iff it is defined on the full domain $\mathbb{N}^n$ and *partial-recursive*.

A set A is *recursively enumerable* iff it is the range of a partial-recursive function. A set A is *recursive* iff there is a recursive function f with $f(x) = 1$ for $x \in A$ and $f(x) = 0$ for $x \notin A$.

As just said, Definition 1.5 permits now to state Matiyasevich's result [68, 69] formally. Let $\langle x,y\rangle$ denotes Cantor's pairing function given by

$$\langle x,y\rangle = \tfrac{1}{2}(x+y)(x+y+1)+y;$$

Cantor's pairing function is a bijection from $\mathbb{N}^2$ to $\mathbb{N}$ and permits mainly to work with partial functions from $\mathbb{N}$ to $\mathbb{N}$ instead of dealing with functions having a domain of the form $\mathbb{N}^n$. Note that $\langle x,y,z\rangle$ is written for $\langle\langle x,y\rangle,z\rangle$, $\langle u,x,y,z\rangle$ for $\langle\langle\langle u,x\rangle,y\rangle,z\rangle$ and so on.

Theorem 1.6: Matiyasevich's Theorem [68, 69]. *A set $A \subseteq \mathbb{N}$ is recursively enumerable iff it is Diophantine. A function $\psi : \mathbb{N} \to \mathbb{N}$ is partial-recursive iff the set $\{\langle x, \psi(x)\rangle : x \in \mathrm{dom}(\psi)\}$ is Diophantine. A set $A \subseteq \mathbb{N}$ is recursive iff the set $\{\langle x, A(x)\rangle : x \in \mathbb{N}\}$ is Diophantine.*

Almost everyone has already dealt with a computer in his life. For that reason, it is quite convenient to provide also the programming definition of the partial-recursive functions. The earliest algorithm was that of Euclid to compute the greatest common divisor of two natural numbers. It could be written down like this.

```
function gcd(x,y)
  { var v = x; var w = y;
    if ((v==0) || (w==0))
      { return(0); }
    while (v != w)
      { if (v<w) { w = w-v; }
        if (w<v) { v = v-w; } }
    return(v); }
```

See `http://www.comp.nus.edu.sg/~gem1501/javascriptintroduction.html` for an introduction of the programming language Java Script used to write these functions. The verbal explanation of the algorithm is the following: given two numbers, check whether both are positive. If so, keep substracting the smaller from the larger number until the remaining numbers are equal and then return this value. If not, just return 0. The following assumptions will be made from now on:

- all variables take as values natural numbers, advanced concepts like arrays are not used;
- the basic operations allowed are addition and substraction with the operation modified to $x - y = 0$ for $y > x$;
- furthermore, if-statements are permitted to do conditional operations;
- for-loops do not permit to modify the variable or bound in the body of the loop;
- while-loops can be executed as long as the corresponding condition is true.

With these restrictions, one is able to characterize the notions of primitive recursive and recursive functions as well as recursively enumerable sets.

Theorem 1.7. *Assume that a program consists of functions $f_1, \ldots, f_n$. Then the functions defined by these programs are partial-recursive. If every f_i only calls functions f_j with $j < i$ and no while-loops are used then all functions are primitive recursive. Every partial-recursive function can defined using programs and every primitive recursive function can be obtained by defining a sequence of functions by programs without while-loops such that each function f_i only calls functions f_j with $j < i$.*

In order to illustrate this characterization by examples, definitions for basic primitive recursive functions and for a function h obtained from other functions g, h by primitive recursive are given below. All these definitions respect the constraints given with the exception that the functions are not numerated, instead those defined later just call functions defined previously. Addition and substraction are denoted by $+$ and $-$, respectively. Only examples are given for function returning constants and projections onto one variable out of a tuple of variables.

```
function null(x)
  { return(0); }

function succ(x)
  { return(x+1); }

function projecttwooffive(xone,xtwo,xthree,xfour,xfive)
  { return(xtwo); }

function mult(x,y)
  { var z; var r=0;
    for (z=0;z<y;z=z+1)
      { r = r+x; }
    return(r); }

function cube(x)
  { var r;
    r = mult(mult(x,x),x);
    return(r); }

function factorial(x)
  { var z; var r = 1;
    for (z=0;z<x;z=z+1)
      { r = mult(r,z+1); }
    return(r); }
```

```
function twopower(x)
  { var z; var r = 1;
    for (z=0;z<x;z=z+1)
      { r = r+r; }
    return(r); }

function g(xone,xtwo,xthree,y)
  { .... }   // Parameter function g for primitive recursion

function h(xone,xtwo)
  { .... }   // Parameter function h for primitive recursion

function f(xone,xtwo,xthree)
  { var z; var r = h(xone,xtwo);
    for (z=0;z<xthree;z=z+1)
      { r = g(xone,xtwo,z,r); }
    return(r); }

function k(xone,xtwo,xthree,xfour)
  { .... } // Next function makes mu-minimalization of this one
           // Note that program does not return from calling
           // k(....) if k is undefined for these values.

function muk(xone,xtwo,xthree)
  { var r = 0;
    while (k(xone,xtwo,xthree,r)>0) { r = r+1; }
       // loop can take forever or call of k can take forever
    return(r); } // this is return(mu r(k(xone,xtwo,xthree,r)))
```

One can easily show by induction that every partial-recursive function can be defined by JavaScript programs, even with the here made restrictions to their syntax. On the other hand, one can also show the reverse direction of Theorem 1.7; the proofs are not difficult but lengthy.

An other model to define partial-recursive functions is that of a Turing machine, named after its inventor Alan Turing [114]. This Turing machine has an infinite tape which has cells such that every cell takes one of finitely many symbols. This tape can be viewed as a function $T : \mathbb{Z} \to F$ where F is a finite set of symbols. Furthermore, the Turing machine has a finite set S of states. For every $s \in S$ and every symbol $a \in F$ there is a unique 5-tuple (a, s, a', s', d) in a finite set R of rules where $a, a' \in F$,

$s, s' \in S$ and $d \in \{-1, 0, 1\}$. The Turing machine has two special members $s_0, s_h \in S$ which are the starting and the halting state. Now the following algorithm is run:

1. Let $p = 0$ be the initial position and $c = s_0$ be the current state. The symbols $T[1], T[2], \ldots, T[n]$ are the binary digits of the input x (where n is the number of digits of the input x) and $T[m] = \#$ for $m \notin \{1, 2, \ldots, n\}$.
2. Find the 5-tuple $(a, s, a', s', d) \in R$ with $a = T[p]$ and $c = s$.
3. Update $T[p] = a'$.
4. Update $c = s'$.
5. Update $p = p + d$.
6. If $c \neq s_h$ then goto step 2.
7. If the symbols different from # are a binary representation of a natural number $y \in \mathbb{N}$ then output y and halt.

Then the function f computed by the Turing machine is the output y computed from input x. The value $f(x)$ is undefined if either the Turing machine never halts or the symbols on the tape (between the two infinite strings of # to the left and to the right) do not represent a natural number in the binary format. One can show that also this notion leads to the same concept.

Theorem 1.8. *A function is partial-recursive iff it is computed by a Turing machine.*

Also a further variety of equivalent formulations of the concept of a partial-recursive function had been made. Church observed that these concepts all coincide with what one might call "computable in the intuitive way"; based on this observation he formulated his famous thesis.

Church's Thesis. For any reasonable formalization XY of "computable" and any partial function f, f is computable according the criterion XY iff f is partial-recursive.

It is called a thesis because "reasonable formalization" is a fuzzy and not well-defined word. Many "reasonable formalizations" had been considered like Turing machines, register machines, programming in C, programming in Java Script, programming in Fortran 77 and so on. For all these notions, it had been shown that they coincide with the framework of partial-recursive functions. The word "reasonable" is of course necessary in order to exclude mechanisms which are too powerful or too weak: for example, f is limit-recursive iff there is a two-ary primitive recursive function g such

that $f(x) = y$ iff $g(x,t) = y$ for almost all t. Such a mechanism which works in the limit and can therefore do things which a Turing machine does not succeed to do in finite time. Therefore, the word "reasonable" is used in order to exclude unreasonable model, although those models might be studied in some other context (as computations relative to oracles). For example, Burgin's "super-recursive" computations [13] encorporated processes of computing in the limit and go therefore beyond reasonable models of computation. Burgin thinks that his model refutes Church's Thesis, most other mathematicians just do not follow his approach to incorporate computations in the limit into those basic operations which are permitted to define a reasonable model of computation. The next results show that the notion of "recursively enumerable set" and "recursive set" are quite natural and have many natural characterizations.

Theorem 1.9. *A nonempty set A is recursively enumerable iff it is the range of a partial-recursive function iff it is the domain of a partial-recursive function iff it is the range of a recursive function iff it is the range of a primitive recursive function.*

A set A is recursive iff A and $\mathbb{N} - A$ are both recursively enumerable iff its characteristic function is recursive iff A is either finite or the range of a strictly monotonic increasing recursive function.

Proof. In the proof, the corresponding algorithms are written down informally. They are not worked out into the last detail in a programming language. Recall that by Definition 1.5, a set is recursively enumerable iff it is the range of a partial-recursive function.

Now it is shown that the other four formulations are equivalent. So assume that A is not empty and let a be an element of A. Furthermore, assume that A is the domain of the partial-recursive function f. This function f can be computed by some computer program which can be simulated step by step. Hence there is a primitive-recursive function g such that

$$g(x,t) = \begin{cases} 0 & \text{if the computer program for } f(x) \\ & \text{terminates within } t \text{ steps;} \\ 1 & \text{otherwise.} \end{cases}$$

Now one can define a function h such that

$$h(x,t) = \begin{cases} a & \text{if } g(x,t) = 1; \\ x & \text{if } g(x,t) = 0; \end{cases}$$

now the range of h is $A \cup \{a\}$ which, by assumption on a, is just A. As A is the range of a primitive recursive function, it is also the range of a recursive and of a partial-recursive function.

For the other way round, assume that A is the range of some partial-recursive

function $\tilde{h}$. Also $\tilde{h}$ has a computer program and one can simulate this to get the following function $\tilde{g}$:

$$\tilde{g}(x,t) = \begin{cases} \tilde{h}(x) & \text{if the computer program for } \tilde{h}(x) \\ & \text{terminates within } t \text{ steps;} \\ a & \text{otherwise.} \end{cases}$$

For the converse let $\pi_1(u)$ and $\pi_2(u)$ be the unique numbers with $\langle \pi_1(u), \pi_2(u) \rangle = u$ for all u. Now one can modify $\tilde{g}$ to

$$\tilde{f}(y) = \mu u(g(\pi_1(u), \pi_2(u)) = y).$$

The resulting function $\tilde{h}$ has exactly the domain A and this completes the proof for the characterization of the r.e. sets.

Again it is a definition that A is recursive iff its characteristic function with $A(x) = 1$ for $x \in A$ and $A(x) = 0$ for $x \notin A$ is recursive. Clearly one can introduce functions f_0, f_1 such that $f_a(x)$ is defined iff $f(x) = a$ for $a \in \{0,1\}$. Hence one obtains that A and $\mathbb{N} - A$ are the domains of partial-recursive functions and recursively enumerable.

For the converse direction assume that A and $\mathbb{N} - A$ are both recursively enumerable. If $A = \emptyset$ or $A = \mathbb{N}$ then A is clearly recursive. Otherwise there are recursive functions f_0, f_1 with range $\mathbb{N} - A$ and A, respectively. Now let

$$g(x) = \mu y(f_0(y) = x \vee f_1(y) = x).$$

Then one can build the characteristic function of A by case-distinction:

$$A(x) = \begin{cases} 0 & \text{if } f_0(g(x)) = x; \\ 1 & \text{if } f_1(g(x)) = x. \end{cases}$$

As the range of f_0, f_1 is disjoint, the two conditions cannot contradict each other.

Given A, define recursively a function h by $h(0) = \mu y(A(y) = 1)$ and $h(S(x)) = \mu y(y > h(x) \wedge A(y) = 1)$. If A is infinite then h is a recursive function with range A. For the converse direction, note that A is clearly recursive if A is finite; one could build the function by looking up in a table. So assume again that A is infinite and the range of an strictly increasing recursive function h. As $h(x) \geq x$ for all x, the set A is recursive by the condition

$$x \in A \Leftrightarrow x \in \{h(0), h(1), \ldots, h(x)\}$$

which can be checked effectively. ∎

Exercise 1.10. For the ease of notation, it is advantageous to permit integer coefficients and values for the polynomials defining Diophantine sets, but the variables still range over natural numbers. Given two disjoint Diophantine sets A and B, construct a Diophantine set C such that

$$C = \{\langle x, 0\rangle : x \in A\} \cup \{\langle x, 1\rangle : x \in B\}$$

by describing how the polynomial f^C with $C = \{x : \exists y_1, ..., y_n\, [f^C(x, y_1, \ldots, y_n) = 0]\}$ is constructed from the polynomials f^A for A and f^B for B; it is possible to introduce new variables.

Exercise 1.11. Prove that a function f is partial-recursive iff its graph $\{\langle x, f(x)\rangle : x \in \mathrm{dom}(f)\}$ is recursively enumerable.

Exercise 1.12. Which of the following sets B, C, D, E, F derived from a set A are recursively enumerable whenever A is:

$$\begin{aligned}
B &= \{x : \exists y\, [\langle x, y\rangle \in A]\};\\
C &= \{x : \forall y\, [\langle x, y\rangle \in A]\};\\
D &= \{x : \exists y\, [\langle x, y\rangle \notin A]\};\\
E &= \{x : \forall y\, [\langle x, y\rangle \notin A]\};\\
F &= \{x : \exists y, z\, [y < z \wedge \langle x, y\rangle, \langle x, z\rangle \in A]\}.
\end{aligned}$$

Exercise 1.13. Match the following Diophantine sets to the below verbal descriptions.

$$\begin{aligned}
G &= \{x : \exists y_1, y_2, y_3\, [x = (y_1 + 2) \cdot (y_2 + 2) \cdot (y_3 + 2)]\};\\
H &= \{x : \exists y_1\, [x + x = y_1 \cdot (y_1 + 1)]\};\\
I &= \{x : \exists y_1, y_2\, [y_1 \cdot y_1 + y_2 \cdot y_2 = x]\};\\
J &= \{x : \exists y_1, y_2\, [(y_1 + y_1 + 3) \cdot (y_2 + y_2 + 3) = x]\};\\
K &= \{x : \exists y_1\, [x = y_1 + y_1 + 4]\}.
\end{aligned}$$

The descriptions are:

- the set of all sums of two squares;
- the set of all odd non-prime numbers;
- the set of all even non-prime numbers;
- the set of all sums of the form $0 + 1 + 2 + \ldots + y$;
- the set of all numbers with at least three prime factors.

2 Numberings of partial-recursive functions

One fundamental result of recursion-theory is that there are effective numberings of all recursively enumerable sets as well as of all partial-recursive functions. Here a numbering is defined as follows.

Definition 2.1. A *numbering* is (depending on the context) either a list of subsets $L_0, L_1, L_2, \ldots$ of $\mathbb{N}$ such that $\{\langle e, x\rangle : x \in L_e\}$ is recursively enumerable or a list of partial functions $\phi_0, \phi_1, \phi_2, \ldots$ from $\mathbb{N}$ to $\mathbb{N}$ such that $\langle e, x\rangle \mapsto \phi_e(x)$ is partial-recursive. A numbering is *universal* iff it lists all r.e. sets or partial-recursive functions, respectively.

The existence of universal numberings of all r.e. sets and all partial-recursive functions are based on the following observations.

- There is a quite straight-forward enumeration of all possible program-texts; for example, each program is represented by a file in a computer consisting of bytes (numbers 0 to 255) where without loss of generality the first byte is different from 0. So one could interpret each file as a number in the system with base 256, so the file consisting of the bytes $(35, 38, 66, 32)$ would give the number $35 \cdot 256^3 + 38 \cdot 256^2 + 66 \cdot 256 + 32$. Conversely one could reverse the mapping to assign to every number a content of a file which could then be interpreted as a program text. Let T_e be the text assigned to number e.

- One can check programs for syntactic correctness and also check whether they satisfy the constraints before and in Theorem 1.7 and as well check whether the function defined in the program has exactly one input variable. If this is true for program T_e, let $\varphi_e(x)$ take as value the output y produced by the program on input x. If the program does not halt and the simulation therefore also does not halt then $\varphi_e(x)$ is undefined. Furthermore, $\varphi_e(x)$ is undefined whenever T_e is not of the adequate form (as described above).

- Note that all these operations can be done effectively in e and x. That is, the function $\langle e, x\rangle \mapsto \varphi_e(x)$ is a partial-recursive function and $\varphi_0, \varphi_1, \varphi_2, \ldots$ is a numbering of all partial-recursive functions.

- One can also simulate the computations for a number s of computation steps. Then the function $\varphi_{e,s}(x)$ is defined iff the computation halts in s steps, otherwise $\varphi_{e,s}(x)$ is undefined. Note that this can be checked effectively, that is, the set $\{\langle e, x, s\rangle : \varphi_{e,s}(x) \text{ is defined}\}$ is primitive-recursive.

- The set W_e can be defined from φ_e. Following a common convention, W_e is the domain of φ_e.
- One can define approximations $W_{e,s}$ to W_e; without loss of generality, $W_{e,s} \subseteq \{0, 1, \ldots, s\}$ for all s. The set $\{\langle e, x, s\rangle : x \in W_{e,s}\}$ is primitive recursive.

Furthermore, numberings of subclasses of the r.e. sets or partial-recursive functions are also considered. Note that notions like "number of steps" depend on the underlying machine model. In the case that one does not like this unprecise definition, one can take Matiyasevich's characterization and use the following alternative but more precise definition.

Remark 2.2. Let φ_e and W_e as above. The set $A = \{\langle e, x, y\rangle : \varphi_e(x) = y\}$ is recursively enumerable and thus Diophantine. So there are polynomials f, g in $n+1$ variables with coefficients in $\mathbb{N}$ such that

$$A = \{u : \exists z_1 \ldots \exists z_n \; [f(u, z_1, \ldots, z_n) = g(u, z_1, \ldots, z_n)]\}$$

and this set is the union of all sets A_s defined as

$$A_s = \{u : u < s \wedge \exists z_1 < s \ldots \exists z_n < s \; [f(u, z_1, \ldots, z_n) = g(u, z_1, \ldots, z_n)]\}.$$

Now $\varphi_{e,s} = y$ iff $\langle e, x, y\rangle \in A_s$. Similarly $W_{e,s}$ is the set of all x such that there is an $y < s$ with $\langle e, x, y\rangle \in A_s$.

Besides Diophantine sets, also Turing machines are popular because for them things like the space and time used can also be defined in a very natural way. So each update-cycle just corresponds to one time step and the computation-space used by a Turing machine is just the number of different fields it has accessed from the begin of the computation until it reaches the halting state.

Blum [9] introduced an abstract concept of complexity measure. So $\Phi_e(x)$ would be the complexity assigned to the computation of $\varphi_e(x)$ and the complexity measure has to follow the following axioms:

- $\Phi_e(x) < \infty$ if $\varphi_e(x)$ is defined and $\Phi_e(x) = \infty$ if $\varphi_e(x)$ is undefined. So $\Phi_e(x) = \infty$ means that either program T_e with input x either runs forever or T_e is syntactically incorrect.
- The set $\{\langle e, x, s\rangle : \Phi_e(x) \leq s\}$ is recursive.

A possible choice for $\Phi_e(x)$ would be the time the program T_e with input x is running. But this is not the only choice which can be made. Another famous measure is the space complexity used by a Turing machine.

An important property of programs is that one can systematically change and translate them. For example, one can make a program which translates T_e into a new text $T_{e'}$ such that $\varphi_{e'}(x) = \varphi_e(x)+1$ whenever $\varphi_e(x)$ is defined. This leads to the following properties of φ, recall that $\langle x_1, \ldots, x_n \rangle$ is formed by iteratively making pairs. The result is known as "s-m-n Theorem" or "Substitution-Theorem".

Theorem 2.3. *For all m, n, a partial function $f(e_1, \ldots, e_m, x_1, \ldots, x_n)$ is partial-recursive iff there is a recursive function g such that*

$$\forall e_1, \ldots, e_m, x_1, \ldots, x_n \; [f(e_1, \ldots, e_m, x_1, \ldots, x_n) = \varphi_{g(e_1,\ldots,e_m)}(\langle x_1, \ldots, x_n \rangle)].$$

Here "=" means that either both sides are defined and equal or both sides are undefined.

The programming environment can be used to show one direction. Given a program for f, one can easily construct a function g which appends a further function h to the program such that the resulting program $g(e_1, e_2, \ldots, e_m)$ does the required job. In this example, $m = 3$ and $n = 2$ and the given function is this one.

```
function f(eone, etwo, ethree, xone, xtwo)
  { return(312+eone+etwo+ethree+xone+xone+xtwo+xtwo)}
```

Now the function g copies the program `function f` and then adds a new part which is called `function h` and computes $\varphi_{g(e_1,e_2,e_3)}(\langle x_1, x_2 \rangle)$. Note that this part encodes the values of e_1, e_2, e_3 and has to decode the pair `xpair` which is the given input instead of x_1, x_2 themselves. In this example, $e_1 = 17$, $e_2 = 4$ and $e_3 = 42$.

```
function f(eone, etwo, ethree, xone, xtwo)
  { return(312+eone+etwo+ethree+xone+xone+xtwo+xtwo)}

function h(xpair)
  { var eone = 17; var etwo = 4; var ethree = 42;
    var xone = 0; var xtwo = 0;
    var search = 0;
      // search represents (xone+xtwo)*(xone+xtwo+1)/2+xtwo
    while (search != xpair)
      { search = search+1;
        if (xone > 0)
          { xtwo = xtwo+1; xone = xone-1; }
        else
          { xone = xtwo+1; xtwo = 0; } }
    return(f(eone,etwo,ethree,xone,xtwo)); }
```

Note that only the initialization of the variables `eone`, `etwo` and `ethree` depends on the inputs of g, the remaining parts are always the same. Thus it is easy to write a program for the function g which produces programs for f and h such that in h the initial values for `eone`, `etwo` and `ethree` are taken according to the input variables of g.

A consequence of this is that every further numbering of partial-recursive functions can be translated into the numbering φ. This property of a numbering is called "acceptable".

Definition 2.4. A numbering ψ is called an *acceptable numbering* or a *Gödel numbering* iff for every further numbering ϑ there is a recursive function f with $\vartheta_e = \psi_{f(e)}$ for all e. Here $\vartheta_e = \psi_{f(e)}$ means that for all x either both $\vartheta_e(x)$ and $\psi_{f(e)}(x)$ are undefined or both $\vartheta_e(x)$ and $\psi_{f(e)}(x)$ are defined and take the same value.

One can show that the numbering φ is acceptable. Furthermore, the question is whether every numbering is acceptable. The answer is that some numberings are not acceptable.

Theorem 2.5. *The numbering φ is acceptable. There is a further numbering ψ such that ψ is not acceptable.*

Proof. To see that φ is acceptable, consider any numbering ϑ and the partial-recursive function f given as $f(e,x) = \vartheta_e(x)$ for all e, x. By Theorem 2.3, there exists a function g with $\varphi_{g(e)}(x) = \vartheta_e(x)$ for all e, x. It follows that $\varphi_{g(e)} = \vartheta_e$ for all e. Thus φ is acceptable.

For the construction of ψ, let $A = \{\langle i,j\rangle : \exists d, e \leq i\ [\varphi_e(d) = \langle i,j\rangle]\}$. The set A recursively enumerable and let A_s be the se of elements enumerated into A by step s. Note that for each i there are only finitely many j with $\langle i,j\rangle \in A$. Now define ψ as follows:

$$\psi_{\langle i,j\rangle}(x) = \begin{cases} \varphi_i(x) & \text{if } \langle i,j\rangle \notin A_x; \\ \uparrow & \text{if } \langle i,j\rangle \in A_x. \end{cases}$$

Here $\uparrow$ means that $\psi_{\langle i,j\rangle}(x)$ remains undefined. The numbering ψ covers all functions as for each i there are infinitely many j with $\langle i,j\rangle \notin A$; for these j it then holds that $\psi_{\langle i,j\rangle} = \varphi_i$. Now assume by way of contradiction that there is a recursive function φ_e with $\psi_{\varphi_e(d)} = \varphi_d$ for all d. Then choose d to be the least index such that φ_d is total and $\varphi_d \neq \varphi_0, \varphi_d \neq \varphi_1, \ldots, \varphi_d \neq \varphi_e$. Let i, j be the numbers with $\langle i,j\rangle = \varphi_e(d)$.

If $i \geq d$ then $\langle i,j\rangle \in A$ and $\varphi_d \neq \psi_{\langle i,j\rangle}$ as the latter function is almost everywhere undefined. If $i < d$ and $\psi_{\langle i,j\rangle}$ is total then $\psi_{\langle i,j\rangle} = \varphi_i$ and $\varphi_i \neq \varphi_d$ by choice of d; hence $\psi_{\langle i,j\rangle} \neq \varphi_d$. If $i < d$ and $\psi_{\langle i,j\rangle}$ is partial then $\varphi_d \neq \psi_{\langle i,j\rangle}$ again. This contradicts the assumption on φ_e and therefore ψ is not an acceptable numbering. ∎

One might ask which other classes of functions have numberings. As undefined places are unpleasant, one would like to find a numbering of all recursive functions. Somehow, this does not exist. As Turing put it: "If a machine is infallible then it cannot also be intelligent." In other words, if a numbering (produced by some Turing machine) contains only total functions then it is not intelligent enough to contain all total recursive functions. The argument to prove this is diagonalization which can be traced back to Cantor [16]. The idea is the following: Given a numbering ψ of total recursive functions, consider the function f with $f(x) = \psi_x(x) + 1$ for all x; this function f is recursive, total and different from the functions in the numbering ψ.

Theorem 2.6. *There is no numbering of all total recursive functions.*

This result has an important application: it shows that there is a recursively enumerable set which is not recursive. Namely this set is diagonal halting problem.

Theorem 2.7. *The set $\mathbb{K} = \{x : \varphi_x(x)$ is defined$\}$ is recursively enumerable but not recursive.*

The set is recursively enumerable because it is the domain of the partial-recursive function γ mapping x to $\varphi_x(x)$. But $\mathbb{K}$ cannot be recursive as the following function f differs from all functions φ_e and is hence not recursive:

$$f(x) = \begin{cases} 0 & \text{if } \gamma(x) \text{ is undefined;} \\ \gamma(x) + 1 & \text{if } \gamma(x) \text{ is defined.} \end{cases}$$

This is most easily seen in the programming-paradigm, so assume that `gamma` is a program for γ and `domgamma` is a program for the domain $\mathbb{K}$ of γ. Then the program for f would be the following.

```
function gamma(x)
  { var y; ........; return(y); }

function domgamma(x)
  { var y; ........; return(y); }

function f(x)
  { if (domgamma(x)==1)
      { return(gamma(x)+1); }
    else
      { return(0); } }
```

The most important thing is that this program clearly does not run the subprogram for $\gamma(x)$ in the case that $\gamma(x)$ is undefined. Hence the function f is total.

Mathematicians have identified many other sets which are not recursive but recursively enumerable. Here some examples:

- The set of all x such that $W_x \neq \emptyset$;
- The set of all x such that $8 \in W_x$;
- The set of all x such that $|W_x| > x$, here $|W_x|$ is the cardinality of W_x;
- The set of all x such that $\varphi_x(0)$ halts and outputs 256;
- The set of all x such that there are $e, d < \sqrt{x} - 2$ with $\varphi_e(d) = x$.

One can also formulate such sets with other notions of mathematics, for example the set of all x such that the x-th Diophantine equation has a solution (for a suitable coding of the Diophantine equations). This list is now not extended here as it would need to introduce the concepts which were used to define them; these concepts might need more explanations than the list gives insight.

One topic studied in recursion theory is the question on whether a given class of r.e. sets has a numbering (with exactly the sets of this class in the numbering) and whether it has a one-one numbering (in which every set occurs once). Clearly there are classes without a numbering, the reason is that there are countably many numberings but uncountably many classes of r.e. sets. Furthermore, one can consider the following numbering:

$$\begin{aligned} L_{2x} &= \begin{cases} \{2x\} & \text{if } x \notin \mathbb{K}; \\ \{2x, 2x+1\} & \text{if } x \in \mathbb{K}; \end{cases} \\ L_{2x+1} &= \begin{cases} \{2x+1\} & \text{if } x \notin \mathbb{K}; \\ \{2x, 2x+1\} & \text{if } x \in \mathbb{K}. \end{cases} \end{aligned}$$

The numbering $L_0, L_1, L_2, \ldots$ is not one-one as for every set $\{2x, 2x+1\}$ with $x \in \mathbb{K}$ there are two indices. If $H_0, H_1, H_2, \ldots$ would be a one-one numbering of this class, then $\mathbb{K}$ would be recursive, a contradiction: Let f be the recursive function finding the unique index e with $y \in H_e$; now it holds that $x \in \mathbb{K} \Leftrightarrow f(2x) = f(2x+1)$. Kummer [56, 57] showed the following theorem which permits to construct Friedberg numberings for many classes.

Theorem 2.8. *Let $L_0, L_1, L_2, \ldots$ and $H_0, H_1, H_2, \ldots$ be two numberings such that*

- *$L_a \neq H_b$ for all a, b (that is, the numberings are disjoint);*

- $H_a \neq H_b$ *whenever* $a \neq b$ (*that is,* $H_0, H_1, H_2, \ldots$ *is a one-one numbering*);
- $\forall a \forall$ *finite* $D \subseteq L_a \exists^\infty b [D \subseteq H_b]$.

Then there is a one-one numbering $F_0, F_1, F_2, \ldots$ *for the class* $\{L_0, L_1, L_2, \ldots\} \cup \{H_0, H_1, H_2, \ldots\}$.

Proof. The construction uses two auxiliary sets A and B, A is defined before and B during the construction of the sets. The set A mainly stores information on which indices in the numbering $L_0, L_1, L_2, \ldots$ are minimal indices and which not; the set B_s stores all the b for which at stage s already an index in the new enumeration $F_0, F_1, F_2, \ldots$ has been found. Let $L_{a,s}$ be the set of elements enumerated into L_a within s steps, similarly one defines $H_{b,s}$. Now one defines

$$\langle a, x\rangle \in A \Leftrightarrow \exists b < a \exists s > x \; [L_{a,s} \cap \{0, 1, \ldots, x\} = L_{b,s} \cap \{0, 1, \ldots, x\}].$$

It is easy to see that $\{x : \langle a, x\rangle \in A\} = \mathbb{N}$ if there is a $b < a$ with $L_b = L_a$. Furthermore, assume now the case that $L_b \neq L_a$ for all $b < a$. Then there is for each $b < a$ a number y_b with $L_b(y_b) \neq L_a(y_b)$. Furthermore, there is a number t such that $t > y_b$ and $L_{b,t}(y_b) = L(y_b) \neq L_a(y_b) = L_{a,t}(y_b)$ for all $b < a$. Then, for all $x > t$, for all $s > x$ and all $b < a$ it holds that $x > y_b$ and $L_{b,s}(y_b) \neq L_{a,s}(y_b)$. Hence, $\langle a, x\rangle \notin A$ for all $x > t$. So the set $\{x : \langle a, x\rangle \in A\}$ is finite. There is a one-one enumeration $\langle a_0, x_0\rangle, \langle a_1, x_1\rangle, \langle a_2, x_2\rangle, \ldots$ of the set

$$\{\langle a, x\rangle : \forall y < x \; [\langle a, y\rangle \in A]\}.$$

Now one constructs now together with the desired numbering $F_0, F_1, F_2, \ldots$ using the set B for book-keeping purposes. The idea of the construction is the following: F_{2c} becomes L_{a_c} if $\langle a_c, x_c\rangle \notin A$; in the case that $\langle a_c, x_c\rangle$ is enumerated into A, one stops the enumeration of F_{2c} and searches a set H_{d_c} not yet assigned to some $F_{c'}$ which contains the data enumerated so far into F_{2c} and redefines $F_{2c} = H_{d_c}$. The sets F_{2c+1} are always taken to be the first member of $H_0, H_1, H_2, \ldots$ not already covered by the enumeration $F_0, F_1, F_2, \ldots$ at stage where this set is defined. The set B is used to do the bookkeeping for this algorithm and contains at stage s the indices of all $H_0, H_1, H_2, \ldots$ which have already been assigned a partner in the enumeration of the $F_0, F_1, F_2, \ldots$ and which will therefore not be used again.

More formally, the construction is done in stages s and in each stage s, finitely many elements (perhaps none) are put into some set F_a. $B_0 = \emptyset$ and $F_{c,0} = \emptyset$ for all c. In stage s, the versions with index $s + 1$ are built.

- Stage $s = \langle c, 0\rangle$.
  Let $b_c = \min(\mathbb{N} - B_s)$.

Let $F_{2c,s+1} = L_{a_c,0}$.
Let $F_{2c+1,s+1} = H_{b_c,0}$.
Let $B_{s+1} = B_s \cup \{b_c\}$.

- Stage $s = \langle c, t\rangle$ with $t > 0$. Let $F_{2c+1,s} = H_{b_c,t}$.
  Now check whether $\langle a_c, x_c\rangle \in A_t$.
  There are three cases.

  - Case $\langle a_c, x_c\rangle \notin A_t$.
    Let $F_{2c,s+1} = L_{a_c,t}$.
    Let $B_{s+1} = B_s$.
  - Case $t = \min\{t' : \langle a_c, x_c\rangle \in A_{t'}\}$.
    Let $\langle d_c, u_c\rangle$ be the first pair such that $d_c \notin B_s$ and $F_{2c,s} \subseteq H_{d_c,u_c}$.
    Let $F_{2c,s+1} = H_{d_c,\max(\{u_c,t\})}$.
    Let $B_{s+1} = B_s \cup \{d_c\}$.
  - Case $t > \min\{t' : \langle a_c, x_c\rangle \in A_{t'}\}$.
    Let d_c, u_c as defined in some previous step.
    Let $F_{2c,s+1} = H_{d_c,\max(\{u_c,t\})}$.
    Let $B_{s+1} = B_s$.

  Let $F_{c',s+1} = F_{c',s}$ for all $c' \notin \{2c, 2c+1\}$.

Now it is verified that the construction succeeds. This is done by verifying the following facts.

Every set H_b occurs in the numbering exactly once. More precisely, all $b \in B$ are indices of sets H_b occurring in the numbering $F_0, F_1, F_2, \ldots$; at each stage $s = \langle c, 0\rangle$ the lowest number outside B_s is moved inside B_{s+1}; a set F_e is made equal to H_b exactly in that stage s where b goes into B, hence this happens for every b exactly once.

Let a be any index. Then there is a smallest number a' with $L_{a'} = L_a$. Furthermore, there is a maximal x' with $\langle a', x'\rangle = \langle a_c, x_c\rangle$ for some c.

$\langle a_c, x_c\rangle \notin A$ for this c and hence $F_{2c} = L_a$. Furthermore, for all $c' \neq c$ with $a_{c'} = a_c$ it holds that $\langle a_{c'}, x_{c'}\rangle \in A$. There is a first t with $\langle a_{c'}, xc'\rangle \in A_t$ and the set $F_{2c',s}$ with $s = \langle c', t\rangle$ is a finite subset of L_a. By the third condition in the statement of the theorem, the search for $d_{c'}$ terminates and $F_{a_{c'},x_{c'}} = H_{d_{c'}}$, so $F_{a_{c'},x_{c'}} \notin \{L_0, L_1, L_2, \ldots\}$. So L_a has exactly one index in the numbering $F_0, F_1, F_2, \ldots$, namely $F_{2c} = L_a$.

From these considerations, it follows that $F_0, F_1, F_2, \ldots$ is a one-one numbering of the class $\{L_0, L_1, L_2, \ldots\} \cup \{H_0, H_1, H_2, \ldots\}$. This completes the proof. ∎

Friedberg [35] discovered that the class of all r.e. sets has a one-one numbering.

Theorem 2.9: Friedberg Numberings [35]. *There is a one-one numbering of all r.e. sets.*

Proof. Note that there is a one-one numbering $H_0, H_1, H_2, \ldots$ of all finite sets with odd cardinality. The idea would be to determine for every e the binary representation $e = 2^{x_1} + 2^{x_2} + \ldots + 2^{x_n}$ with $x_1, x_2, \ldots, x_n$ being different natural numbers and to let

$$H_e = \begin{cases} \{x_1 + 1, x_2 + 1, \ldots, x_n + 1\} & \text{if } n \text{ is odd;} \\ \{0, x_1 + 1, x_2 + 1, \ldots, x_n + 1\} & \text{if } n \text{ is even.} \end{cases}$$

Then $H_0 = \{0\}$, $H_1 = \{1\}$, $H_{37} = \{1, 3, 6\}$ and $H_{39} = \{0, 1, 2, 3, 6\}$ as $1 = 2^0$, $37 = 2^0 + 2^2 + 2^5$ and $39 = 2^0 + 2^1 + 2^2 + 2^5$. So $H_0, H_1, H_2, \ldots$ is a one-one enumeration of all finite sets of odd cardinality.

Without loss of generality, assume that $|W_{e,s+1} - W_{e,s}| \leq 1$ and $W_{e,0} = \emptyset$ for all e, s. Now let $L_{e,0} = \emptyset$ and $L_{e,s+1} = W_{e,s+1}$ if $W_{e,s+1}$ has an even number of elements and $L_{e,s+1} = L_{e,s}$ otherwise. One can easily prove by induction that $L_{e,s}$ has always an even number of elements and that $|W_{e,s} - L_{e,s}| \leq 1$ for all e, s. Let $L_e = \cup_s L_{e,s}$. The numbering $L_0, L_1, L_2, \ldots$ contains all r.e. finite sets of even cardinality and all r.e. infinite sets. Hence $\{H_0, H_1, H_2, \ldots\} \cap \{L_0, L_1, L_2, \ldots\} = \emptyset$.

As $L_{e,s}$ has at every stage an finite and even number of elements, each set $L_{e,s} \cup \{\max(L_{e,s}) + 1 + d\}$ occurs in the list $H_0, H_1, H_2, \ldots$; hence every finite subset of a set L_a has infinitely many supersets H_b.

So these two classes satisfy the three conditions of Theorem 2.8 and that the union of these two classes is the class of all r.e. sets. Hence, the class of all r.e. sets has a one-one numbering. ∎

Exercise 2.10. Adapt the above proof to show that there is a one-one numbering of all partial-recursive functions.

Comprehensive Exercise 2.11. As seen, some classes of r.e. sets have one-one numberings and others do not. The task of this exercise is to show that there is a class where every set occurs infinitely often in each numbering of the class. The proof should use the following steps.

- Define the mapping $f : \langle x, y \rangle \to \frac{x+1}{y+1}$ from $\mathbb{N}$ into the positive rational numbers. Show that this function is onto.

- Define a family of r.e. sets as follows: For parameter e, search for two numbers x, y such that $f(x) < f(y) < f(x) + 2^{-e}$ and $\varphi_e(x), \varphi_e(y)$ are both defined and different; if x, y are found then let $V_e = \{z : f(x) \leq f(z) \leq f(y)\}$ else let $V_e = \emptyset$. Prove that $\{\langle e, z \rangle : z \in V_e\}$ is recursively enumerable.

- Let $x \sim_0 y$ if there is an e with $x, y \in V_e$ and $x \sim_{n+1} y$ iff $\exists z\ [x \sim_n z \wedge z \sim_n y]$. Define $L_x = \{y : \exists n\ [x \sim_n y]\}$ for each set. Prove that $L_0, L_1, L_2, \ldots$ is a numbering of r.e. sets.

- Prove that no L_x is recursive. To see this, first fix x. Then prove that L_x contains x but $L_x \neq \mathbb{N}$. Furthermore, prove that either $L_x = L_y$ or $L_x \cap L_y = \emptyset$ for all $y \neq x$. Then show that no φ_e is the characteristic function of L_x.

- Assume that $H_0, H_1, H_2, \ldots$ is some numbering for the class $\{L_0, L_1, L_2, \ldots\}$. Furthermore, assume that some L_x occurs in the numbering $H_0, H_1, H_2, \ldots$ only finitely often. Show that then this set L_x is recursive in contradiction to what has been shown previously.

Exercise 2.12. Let A be a co-infinite recursively enumerable set, $L_x = A$ for $x \in A$ and $L_x = \{x\}$ for $x \notin A$. Show that the class $\{L_0, L_1, L_2, \ldots\}$ has a one-one numbering iff A is recursive.

Theorem 2.13. *No Friedberg numbering is a Gödel numbering.*

Proof. Assume that $H_0, H_1, H_2, \ldots$ is a Friedberg numbering, that is, a one-one numbering of all r.e. sets. Consider the numbering $L_0, L_1, L_2, \ldots$ from Exercise 2.12 with $A = \mathbb{K}$; that is, let $L_x = \mathbb{K}$ for $x \in \mathbb{K}$ and let $L_x = \{x\}$ for $x \notin \mathbb{K}$. If $H_0, H_1, H_2, \ldots$ would be an acceptable numbering then there would be a recursive function f with $L_x = H_{f(x)}$ for all x. As $H_0, H_1, H_2, \ldots$ is repetition-free, there is a unique index e with $H_e = \mathbb{K}$. Then it would follow that $x \in \mathbb{K} \Rightarrow L_x = \mathbb{K} \Rightarrow f(x) = e$ and $x \notin \mathbb{K} \Rightarrow L_x = \{x\} \Rightarrow f(x) \neq e$, a contradiction to $\mathbb{K}$ not being recursive. ∎

Exercise 2.14. Prove that there is no numbering of all infinite r.e. sets. To see this, assume by way of contradiction that $L_0, L_1, L_2, \ldots$ would be such a numbering and define inductively a recursive function f such that $f(x) > f(y) + 1$ and $f(x) + 1 \in L_x$ for all x and all $y < x$. Show how such a function can be defined and use a uniform enumeration $L_{x,s}$ of all elements in L_x enumerated into this set within s steps for the search. Note that $\{\langle x, y, s\rangle : y \in L_{x,s}\}$ is uniformly recursive. Once having defined f, show that the range of f is infinite and different from all sets L_x.

Exercise 2.15. Let φ be an acceptable numbering and define ψ from φ using the following equation:

$$\psi_{\langle i,j\rangle}(x) = \begin{cases} \varphi_i(x) & \text{if } x > 0; \\ j - 1 & \text{if } j > 0 \text{ and } x = 0; \\ \uparrow & \text{if } j = 0 \text{ and } x = 0. \end{cases}$$

Show that ψ is a universal numbering in the sense that every partial-recursive function is equal to some $\psi_{\langle i,j\rangle}$. After that, prove that ψ is not a Gödel numbering by first showing that there is a recursive function g satisfying $\varphi_{g(e)}(x) = \varphi_e(e)$ for all e. Then consider any function h such that $\psi_{h(d)} = \varphi_d$ for all d. Let f be the function which assigns to e the second component j of the pair $\langle i, j\rangle$ defined as $\langle i, j\rangle = h(g(e))$. Show that $f(e) \neq \varphi_e(e)$ and hence $f \neq \varphi_e$ for all e. So f cannot be recursive. As g and the projection on the second component of the pair are both recursive, conclude that the function h cannot be recursive and hence ψ is not a Gödel numbering.

Exercise 2.16. Let $L_e = \{e, e+1, e+2, \ldots\}$. Construct a numbering containing all sets L_e plus all sets $L_e - W_e$ where W_e is finite. If W_e is infinite, $L_e - W_e$ should not be added into the family.

Exercise 2.17. Show that the class $\{\{x\} : x \notin \mathbb{K}\}$ has no numbering but that the class $\{\mathbb{N}\} \cup \{\{x\} : x \notin \mathbb{K}\}$ has a numbering. Show that in the latter case this numbering cannot be one-one.

Exercise 2.18. Construct a family $L_0, L_1, L_2, \ldots$ such that

- for every i there are at most two other indices j, k with $L_i = L_j = L_k$;
- for every numbering $H_0, H_1, H_2, \ldots$ of the same family there are three different indices i, j, k such that $H_i = H_j = H_k$.

The construction can be carried out by carefully selecting some finite sets.

Exercise 2.19. Construct a numbering $L_0, L_1, L_2, \ldots$ of all recursive sets. Note that it is only required that $\{\langle e, x\rangle : x \in L_e\}$ is recursively enumerable; it is impossible to make a uniformly recursive numbering of all recursive sets.

Exercise 2.20. Construct a numbering $L_0, L_1, L_2, \ldots$ of sets such that the following conditions hold for all x and y:

- L_x is not empty;
- if $y = \min(L_x)$ and $|W_y| < \infty$ then $|\{y, y+1, y+2, \ldots\} - L_x| = |W_y|$;
- if $H \subseteq \{y, y+1, y+2, \ldots\}$ is a nonempty r.e. set with minimum y, $|W_y| < \infty$ and $|\{y, y+1, y+2, \ldots\} - H| = |W_y|$ then there is an x with $L_x = H$;
- if $y = \min(L_x)$ and $|W_y| = \infty$ then $L_x = \{y, y+1, y+2, \ldots\}$.

3 Rice's Theorem and the Arithmetical Hierarchy

One central question of recursion theory is whether two sets have the same difficulty or whether one is more complicated than the other. This of course needs ways to express that "A is at most as complicated as B" formally. The idea is to say that "A can be translated into B" or "A can be reduced to B" which is the more technical way to express it. The first reducibility considered is many-one reduction.

Definition 3.1. A set A is many-one reducible to B iff there is a recursive function f with $A(x) = B(f(x))$ for all x, that is, with $x \in A \Leftrightarrow f(x) \in B$. One writes $A \leq_m B$ for A is many-one reducible to B.

Furthermore, $A \equiv_m B$ iff $A \leq_m B$ and $B \leq_m A$. The class $\{B : B \equiv_m A\}$ is called the many-one degree of A.

The recursive function f in the definition of $A \leq_m B$ is called a many-one reduction from A to B. Note that $\leq_m$ is a transitive partial preordering on the subsets of $\mathbb{N}$ and that $\equiv_m$ is an equivalence relation. Furthermore, $\leq_m$ induces a partial ordering on the class of many-one degrees.

The next result shows that $\mathbb{K}$ is the most complicated recursively enumerable set which exists.

Theorem 3.2. *A set A is many-one reducible to $\mathbb{K}$ iff A is recursively enumerable.*

Proof. As $\mathbb{K}$ is recursively enumerable, it is the domain of a partial-recursive function g. In the case that $A \leq_m \mathbb{K}$ via a recursive function f, then A is the domain of the partial-recursive function $x \mapsto g(f(x))$. So A is recursively enumerable.

If A is recursively enumerable then one can define the following two-place partial-recursive function f:

$$f(e,x) = \begin{cases} 1 & \text{if } e \in A; \\ \uparrow & \text{if } e \notin A. \end{cases}$$

By Theorem 2.3 there is a recursive function g such that $\varphi_{g(e)}(x) = f(e,x)$ for all e, x. It follows that $\varphi_{g(e)}(g(e))$ is defined iff $e \in A$. Hence $e \in A \Leftrightarrow g(e) \in \mathbb{K}$ and $A \leq_m \mathbb{K}$ via the many-one reduction g. ∎

So the recursively enumerable sets are just the sets in the many-one degrees below $\mathbb{K}$. Besides the many-one degree of $\mathbb{K}$ itself, there are some further many-one degrees. Those of $\emptyset$ and $\mathbb{N}$ consist only of these sets, all further recursive sets form a third recursive many-one degree: if $a \in A$ and $b \in \mathbb{N} - A$ then every further recursive set B is many-one reducible to A via a recursive function which sends every element of B to a and every non-element of B to b. On the other hand, nonrecursive sets are

not many-one reducible to recursive ones, hence the many-one degree of $\mathbb{K}$ does not contain any recursive set.

The question is whether the sets in the many-one degree of $\mathbb{K}$ can be characterised. The notion "creative" was introduced by Post [84] in 1944 and the characterization was obtained by Myhill [75] eleven years later. Only the direction "many-one complete $\Rightarrow$ creative" is shown here, the direction "creative $\Rightarrow$ many-one complete" is given in Theorem 6.6 as the proof makes use of Kleene's fixed-point theorem which will be introduced in Section 6.

Definition 3.3 [75]**.** A set A is called *creative* if it is recursively enumerable and there is a recursive function f such that for all e, $W_e \subseteq \mathbb{N} - A \Rightarrow f(e) \in \mathbb{N} - A - W_e$.

Theorem 3.4: Myhill's Characterization of Many-One Complete sets [75]**.** *A set A is creative iff $A \equiv_m \mathbb{K}$.*

Proof of one direction. Let A be given such that $A \equiv_m \mathbb{K}$. Hence there is a recursive function g witnessing that $\mathbb{K} \leq_m A$; that is, $x \in \mathbb{K} \Leftrightarrow g(x) \in A$ for all x. There is a recursive function h such that

$$W_{h(x)} = \{y : g(y) \in W_x\}$$

for all x; so $W_{h(x)}$ is the preimage of W_x under g. Now consider two cases:

Case $h(x) \in W_{h(x)}$. Then $h(x) \in \mathbb{K}$, $g(h(x)) \in A$ by the definition of g and $g(h(x)) \in W_x$ by the definition of h.

Case $h(x) \notin W_{h(x)}$. Then $h(x) \notin \mathbb{K}$, $g(h(x)) \notin A$ by the definition of g and $g(h(x)) \notin W_x$ by the definition of h.

Assume now that W_x is disjoint from A. Then the first case does not hold as $g(h(x))$ cannot be a member of $W_x \cap A$. So the second case holds and $g(h(x)) \notin A \cup W_x$. Hence A is creative. ∎

Remark 3.5: Productive Sets, Recursively Inseparable Pairs and the Padding Lemma. Dekker [27, 28] has generalized the property of being creative as follows: A set A is *productive* if there is a recursive function f such that $f(x) \in A - W_x$ whenever $W_x \subseteq A$. In particular, productive sets are not recursively enumerable. Myhill [75] shows that A is productive iff $\mathbb{N} - \mathbb{K} \leq_m A$. There are sets A such that both, A and $\mathbb{N} - A$, are productive. An example is the set $\{e : W_e \text{ is finite}\}$.

Two disjoint r.e. sets A, B are called *recursively inseparable* iff there is no recursive function f with $f(x) = 0$ for all $x \in A$ and $f(x) = 1$ for all $x \in B$. An example would be $A = \{e : \varphi_e(e)\!\downarrow > 0\}$ and $B = \{e : \varphi_e(e)\!\downarrow = 0\}$. Now let f be any total function with $f(x) = 0$ for $x \in A$ and $f(x) = 1$ for $x \in B$; in order to show that A, B are recursively inseparable one has to show that f is not recursive. So consider any e. If

$\varphi_e(e)$ is undefined then $\varphi_e \neq f$. If $\varphi_e(e)\downarrow > 0$ then $e \in A$ and $f(e) = 0$, again $\varphi_e \neq f$. If $\varphi_e(e)\downarrow = 0$ then $e \in B$ and $f(e) = 1$, again $\varphi_e \neq f$. Hence f is not recursive and A, B are a recursively inseparable pair; keep this pair fixed from now on.

The *padding lemma* says that there is a recursive function *pad* satisfying $\varphi_{pad(e)} = \varphi_e$ and $pad(e) > e$ for all e. One can use the inseparable pair A, B to show this lemma. To do this, consider the following numbering:

$$\psi_{\langle i,j\rangle}(x) = \begin{cases} j & \text{if } j \in A; \\ \varphi_i(x) & \text{if } j \in B; \\ \uparrow & \text{if } j \notin A \cup B. \end{cases}$$

As $\varphi_0, \varphi_1, \varphi_2, \ldots$ is an acceptable numbering, there is a recursive function f with $\varphi_{f(\langle i,j\rangle)} = \psi_{\langle i,j\rangle}$ for all i, j and the sets

$$E_i = \{f(\langle i, j\rangle) : j \in B\}$$

are uniformly recursively enumerable. In order to see that all sets E_i are infinite, consider a fixed but arbitrary i. Let

$$F_i = \{j : f(\langle i, j\rangle) \in E_i\}.$$

Note that this set consists only of numbers j with $\psi_{\langle i,j\rangle} = \varphi_i$. As all functions $\psi_{\langle i,j\rangle}$ with $j \in A$ are pairwise different, there is at most one number $k \in A$ with $\psi_{\langle i,k\rangle} = \varphi_i$; if such a k does not exist in A, let k just be any number outside $A \cup B$. Now all $j \in A - \{k\}$ satisfy $\psi_{\langle i,j\rangle} \neq \varphi_i$ and $f(\langle i, j\rangle) \notin E_i$. Hence the characteristic function of $F_i - \{k\}$ separates A from B. As A, B is a recursively inseparable pair, the set F_i is not recursive. It follows that the set E_i is infinite.

Now let $pad(i)$ to be the first element enumerated into E_i which is larger than i. As the E_i are infinite and uniformly recursively enumerable, *pad* is a total recursive function. Furthermore, $pad(i) > i$ for all i and $\varphi_{pad(i)} = \varphi_i$ as $pad(i) \in E_i$ for all i.

Definition 3.6. Let $\mathcal{C}$ be a class of recursively enumerable sets. $\mathcal{C}$ is called non-trivial iff it contains some but not all r.e. sets. The index-set of $\mathcal{C}$ is the set $I = \{e : W_e \in \mathcal{C}\}$.

Note that the index set is taken with respect to a Gödel numbering. The following result would fail for Friedberg numberings. Rice [85] showed that all non-trivial index sets are more or less as hard as the halting problem; of course they can be much harder.

Theorem 3.7: Theorem of Rice [85]**.** *Let $\mathcal{C}$ be a non-trivial class, I be the index set of $\mathcal{C}$ and $J = \mathbb{N} - I$ be the index set of the complement of $\mathcal{C}$. Then $\mathbb{K} \leq_m I$ or $\mathbb{K} \leq_m J$.*

Proof. In the case that $\emptyset \notin \mathcal{C}$, there is some other set $A \in \mathcal{C}$. Now define a numbering $L_0, L_1, L_2, \ldots$ by

$$L_x = \begin{cases} \emptyset & \text{if } x \notin \mathbb{K}; \\ A & \text{if } x \in \mathbb{K}. \end{cases}$$

As $W_0, W_1, W_2, \ldots$ is an acceptable numbering, there is a recursive function f with $W_{f(x)} = L_x$. It follows that $W_{f(x)} \in \mathcal{C} \Leftrightarrow x \in \mathbb{K}$. As a consequence, f is a many-one reduction from $\mathbb{K}$ to I.

In the case that $\emptyset \in \mathcal{C}$, one chooses a set $A \notin \mathcal{C}$ and defines $L_0, L_1, L_2, \ldots$ and f in the same way. Then f is a many-one reduction from $\mathbb{K}$ to J. ∎

The Theorem of Rice and Shapiro settles the question when an index set is recursively enumerable. Note that r.e. index sets of non-trivial classes are many-one complete for the halting problem. Odifreddi [79, Proposition II.5.19] states that this theorem had also been discovered by Myhill and Shepherdson [76] and by McNaughton. For this result, one uses the following canonical indices of finite sets which are defined here formally for later use.

Definition 3.8. The canonical indexing of the finite sets is given by $D_{2^{x_1}+2^{x_2}+\ldots+2^{x_n}} = \{x_1, x_2, \ldots, x_n\}$ for any pairwise distinct numbers $x_1, x_2, \ldots, x_n$ with $D_0 = \emptyset$ for the case that $n = 0$.

Theorem 3.9: Theorem of Rice and Shapiro [76]**.** *An index set of a class $\mathcal{C}$ is recursively enumerable iff the following three conditions are satisfied:*

- *$\{x : D_x \in \mathcal{C}\}$ is recursively enumerable;*
- *for any r.e. set A, $A \in \mathcal{C}$ iff there is a finite subset D of A with $D \in \mathcal{C}$;*
- *if A, B are r.e. sets with $A \in \mathcal{C}$ and $A \subseteq B$ then $B \in \mathcal{C}$.*

Note that the third condition is implied by the second.

Proof. Assume that I is r.e.; now the first two properties are shown. The third follows from the second as whenever $A \in \mathcal{C}$ then some finite subset of A is in $\mathcal{C}$ which is also a finite subset of B.

First, there is a recursive function f with $W_{f(x)} = D_x$. Hence $\{x : D_x \in \mathcal{C}\} = \{x : f(x) \in I\}$ and the set $\{x : D_x \in \mathcal{C}\}$ is recursively enumerable.

Second, on one hand it has to be shown that every set in $\mathcal{C}$ has a finite subset also in $\mathcal{C}$. So, let $A \in \mathcal{C}$ and let A_s be the finitely many elements of A enumerated within time s. Similarly $\mathbb{K}_s$ is an approximation to $\mathbb{K}$. Now let

$$L_x = \begin{cases} A & \text{if } x \notin \mathbb{K}; \\ A_s & \text{if } s \text{ is the first stage with } x \in \mathbb{K}_s. \end{cases}$$

As there is a recursive function g with $W_{g(x)} = L_x$, the set of all x with $L_x \in \mathcal{C}$ is recursively enumerable. Furthermore, $x \notin \mathbb{K} \Rightarrow L_x = A \Rightarrow L_x \in \mathcal{C}$. As the complement of $\mathbb{K}$ is not recursively enumerable there is a $x \in K$ with $L_x \in \mathcal{C}$. This L_x is finite and so A has a finite subset in $\mathcal{C}$.

On the other hand, if D is a finite set and A a superset of D then let

$$H_x = \begin{cases} D & \text{if } x \notin \mathbb{K}; \\ A & \text{if } x \in \mathbb{K}. \end{cases}$$

Again one sees that the set $\{x : H_x \in \mathcal{C}\}$ is recursively enumerable as I is. As $\mathbb{N} - \mathbb{K}$ is not recursively enumerable, it follows that all H_x must be in $\mathcal{C}$, thus $A \in \mathcal{C}$.

For the converse direction, assume that all three conditions are satisfied. Then consider the following formula:

$$W_e \in \mathcal{C} \Leftrightarrow \exists s \exists x \; [D_x \in \mathcal{C} \wedge D_x \subseteq W_{e,s}].$$

This formula witnesses that $I = \{e : W_e \in \mathcal{C}\}$ is also recursively enumerable: it is just the domain of a partial-recursive function which searches for the first triple $\langle x, s, t \rangle$ such that $D_x \subseteq W_{e,s}$ and x is enumerated within t steps into the set $\{x : D_x \in \mathcal{C}\}$ which by assumption is recursively enumerable. ∎

Example 3.10. *The set $A = \{x : |W_x| > x\}$ is not an index set.*

Proof. Note that A is an r.e. set as one can simulate the enumeration of W_x until at least $x + 1$ elements are enumerated into W_x and then enumerate x into A. Now there are two cases to consider.

(a) A constains an x with W_x being finite. Then, by the padding lemma, there is an $y > |W_x|$ with $W_y = W_x$. But $y \notin A$ as $|W_y| < y$ and hence A is not an index set.

(b) A contains only indices x with W_x being infinite. Then $A = \{x : W_x \text{ is infinite}\}$ as $|W_x| > x$ for all infinite sets W_x. As A is recursively enumerable, there is a recursive function f with range A. Now $W_{f(0)}, W_{f(1)}, W_{f(2)}, \ldots$ is a numbering of all infinite r.e. sets which, as shown in Exercise 2.14, does not exist. Hence case (b) does not occur and A is not an index-set by case (a). ∎

Exercise 3.11. Define index-sets for partial-recursive functions and transfer the Theorems of Rice and their proofs to this case.

Definition 3.12. A formula is in *normal form* if quantified variables are all before the body of the formula and the body of the formula is a recursive predicate, that is, a recursive $\{0, 1\}$-valued function with 0 interperted as "false" and 1 as "true". A formula without quantified variables is at the same time a Σ_1^0-formula and Π_1^0-formula. Furthermore, for every formula $\phi(x_1, \ldots, x_n)$ with free variables $x_1, \ldots, x_n$ the following holds:

- if $\phi(x_1, \ldots, x_n)$ is Σ^0_n and $k \in \{1, \ldots, n\}$ then $\exists x_k\ \phi(x_1, \ldots, x_n)$ is a Σ^0_n-formula and $\forall x_k\ \phi(x_1, \ldots, x_n)$ is a Π^0_{n+1}-formula;
- if $\phi(x_1, \ldots, x_n)$ is Π^0_n and $k \in \{1, \ldots, n\}$ then $\exists x_k\ \phi(x_1, \ldots, x_n)$ is a Σ^0_{n+1}-formula and $\forall x_k\ \phi(x_1, \ldots, x_n)$ is a Π^0_n-formula.

Furthermore, usually one always takes the best possible level to describe a formula, so if a formula is at the same time a Σ^0_3-formula and Π^0_4-formula then one just calls it a Σ^0_3-formula. A set is Σ^0_n iff it is defined with a Σ^0_n-formula; so if P is a recursive predicate then

$$A = \{x : \exists y_1 \forall y_2 \exists y_3 \forall y_4\ [P(x, y_1, y_2, y_3, y_4)]\}$$

would be a Σ^0_4-set. A set is Σ^0_n-complete iff it is Σ^0_n and every Σ^0_n-set is m-reducible to it. Similarly one defines Π^0_n-sets and Π^0_n-complete sets. A set is Δ^0_n iff it is Σ^0_n and Π^0_n.

Exercise 3.13. Show that a set is Σ^0_1 iff it is recursively enumerable.

Examples 3.14. Here some examples of Σ^0_n-complete and Π^0_n-complete index-sets:

- $\{e : W_e \text{ is not empty}\}$ is Σ^0_1-complete;
- $\{e : |W_e| \leq 32\}$ is Π^0_1-complete;
- $\{e : W_e \text{ is finite}\}$ is Σ^0_2-complete;
- $\{e : W_e = \mathbb{N}\}$ is Π^0_2-complete;
- $\{e : W_e \text{ is recursive}\}$ is Σ^0_3-complete;
- $\{e : W_e \text{ is cofinite}\}$ is Σ^0_3-complete;
- $\{e : W_e \text{ is coinfinite}\}$ is Π^0_3-complete;
- $\{e : W_e \text{ is infinite and coinfinite}\}$ is Π^0_3-complete;
- $\{e : |W_e| \in \{2, 5, 8\}\}$ is Δ^0_2 but neither Σ^0_1 nor Π^0_1;
- $\{e : W_e$ has finitely many even and infinitely many odd numbers$\}$ is Δ^0_3 but neither Σ^0_2 nor Π^0_2.

A sample proof will be given for the class of cofinite sets:

$$\{e : W_e \text{ is cofinite}\} = \{e : \exists x \forall y \exists z\ [(x + y) \in W_{e,z}]\}.$$

Hence the index-set of the cofinite sets is Σ^0_3. Now let any Σ^0_3-set A be given and assume that it is of the following form:

$$A = \{e : \exists x \forall y \exists z \ [P(e,x,y,z) = 1]\}.$$

The following sets G_e and H_e are uniformly recursively enumerable:

$$\begin{aligned} G_{\langle e,x\rangle} &= \{y : \forall v \leq y \exists w \ [P(e,x,v,w) = 1]\}; \\ H_{\langle e,x\rangle} &= \{y : \forall v < y \ [v \in G_{\langle e,x\rangle}]\}. \end{aligned}$$

Note that all sets $H_{\langle e,x\rangle}$ have at least the element 0. There is a recursive uniform one-one enumeration of these sets, this is given as a one-one sequence of triples $\langle e_c, x_c, y_c\rangle$ such that $H_{\langle e,x\rangle} = \{y_c : e_c = e \wedge x_c = x\}$ for all e and x. Now let

$$L_e = \{c : e_c \neq e \vee \exists d \ [e_d = e \wedge x_d \leq x_c \wedge y_d > (x_c - x_d) + y_c]\}$$

It will be shown that L_e is coinfinite iff $e \in A$.

First assume that $e \in A$. Note that for every pair $\langle e, x\rangle$ there is at most one c with $x_c = x$ and $c \notin L_e$. This c can only exist if $G_{\langle e,x\rangle}$ is finite. Then it is the unique c with $e_c = e \wedge x_c = x \wedge |G_{\langle e,x\rangle}| = y_c$. Furthermore, there is an x such that $\forall y \exists z \ [P(e,x,y,z) = 1]$. Fix this x. It is easy to see that $G_{\langle e,x\rangle} = \mathbb{N}$. Now every c with $x_c \geq x$ is in L_e by the way L_e is defined. Hence L_e is cofinite.

Second assume that $e \notin A$. Then all sets $G_{\langle e,x\rangle}$ are finite. There are infinitely many x such that $|G_{\langle e,x\rangle}| + x \geq |G_{\langle e,z\rangle}| + z$ for all $z < x$. For these x, the c with $e_c = e \wedge x_c = x \wedge y_c = |G_{\langle x,e\rangle}|$ satisfies $c \notin L_e$. Hence L_e is coinfinite.

So it holds that $e \in A$ iff L_e is cofinite. Furthermore, there is a recursive function f with $W_{f(e)} = L_e$ for all e. It follows that $e \in A$ iff $W_{f(e)}$ is cofinite. Thus $\{e : W_e$ is cofinite$\}$ is Σ^0_3-complete.

Comprehensive Exercise 3.15. Prove that $\{e : W_e$ is finite$\}$ is Σ^0_2-complete, that $\{e : W_e = \mathbb{N}\}$ is Π^0_2-complete and that $\{e : W_e$ is recursive$\}$ is Σ^0_3-complete.

Remark 3.16. Every Σ^0_n-set is also Σ^0_m and Π^0_m for all $m > n$; similarly every Π^0_n-set is also Σ^0_m and Π^0_m for all $m > n$. To see that Σ^0_n-sets are Π^0_{n+1}, one just has to introduce one unused additional variable and to quantify it universally before the other quantifiers; as this variable occurs nowhere in the formula, it has no influence on the set defined. An example is that $\mathbb{K}$ is not only Σ^0_1 but also Π^0_2 as the following formula shows:

$$\mathbb{K} = \{e : \forall x \exists y \ [e \in W_{e,y}]\}.$$

Furthermore, introducing an unused variable at the last and not the first position gives the inclusion from Σ^0_n-sets to Σ^0_{n+1}-sets as well as from Π^0_n-sets to Π^0_{n+1}-sets.

The arithmetical hierarchy is strongly linked with the concept of relativized recursiveness [115]. One can define computer programs which call other subprograms. Normally the subprogram is written down explicitly, but sometimes one just assumes that a given set A is used as a database in the following sense: the database stores for every $x \in \mathbb{N}$ the information whether $x \in A$ or $x \notin A$. No one cares how A goes into the database, but assuming that it is there, one can - using the programming paradigm - define A-recursive sets, functions and even A-r.e. sets, that is, sets which are recursively enumerable relative to A. In this context, the used database is called an "oracle" – in the time where the concept was introduced, there were no computers or databases and so scientists referred to oracles who, according to Greek mythology, answered questions presented to them using some knowledge not available to the other people. Characterizations which depend heavily on non-relativizable concepts are lost, what, for example, would be an A-Diophantine set or an A-recursive polynomial? But concepts which can naturally be defined relative to the given oracle A carry over: for example, a set B is A-recursive iff both sets B and $\mathbb{N} - B$ are A-r.e. sets. Here an example with programs.

```
// Assume that A = {a0,a1,a2,...} is infinite;
//  B = {n: an+1 notin A} is A-recursive;
//  C = {n: exists m,k (am+n=ak)} is A-r.e.;
//  D = {n: n*n+1 in A} is A-recursive.

function aoracle(x) // oracle for set A
  { if ("x is in A")
      { return(1); }
    else
      { return(0); } }
  // The test whether x is in A does not have a program,
  // it is just looked up in a database or figured out
  // by some other way. In order to make this usable by
  // other functions, this oracle-call is implemented
  // as a function "aoracle". Calling the function "aoracle"
  // is the only way other functions can retrieve information
  // about A.

function b(x) // function b computes characteristic function of B
  { var y = x+1;
    var z = 0;
    while (y>0)
      { if (aoracle(z)==1) { y = y-1; }
```

```
        z = z+1; }
    return(1-aoracle(z)); }

function c(x) // domain of function c is set C
  { var y=0;
    while ((aoracle(y)==0) || (aoracle(y+x)==0)) { y = y+1; }
    return(y); }

function d(x) // function d computes characteristic function of D
  { return(aoracle(x*x+1)); }
```

The relation "B is A-recursive" is also called "B is Turing reducible to A" as the fact that one can define B with a computer program using the oracle A is like saying that B is at most as complicated as A. This reduction is named after Alan Turing, a mathematician who was a pioneer of both, theoretical and practical computer science. Note that $A \leq_m B$ implies $A \leq_T B$ but not vice versa, for example, $\mathbb{K} \leq_T \mathbb{N} - \mathbb{K}$ but $\mathbb{K} \not\leq_m \mathbb{N} - \mathbb{K}$. In the following, φ_e^A denotes the e-th partial-A-recursive function; furthermore, W_e^A is the domain of this function.

Remark 3.17. The set $A' = \{e : e \in W_e^A\} = \{e : \varphi_e^A(e) \text{ is defined}\}$ is called *the jump of A* or the *diagonal halting problem relative to A*. The following facts are known:

- $A' \not\leq_T A$ but $A \leq_m A'$;
- if A is a Δ_n^0-set then A' is a Σ_n^0-set;
- if $A \equiv_T B$ for a Σ_n^0-complete set B then A' is Σ_{n+1}^0-complete;
- for every n, the Σ_n^0-complete sets form one many-one degree.

Exercise 3.18. Let $A = \{e : W_e \text{ is finite}\}$ and $B = \{e : W_e \text{ is cofinite}\}$. Give a direct proof (without using the characterizations through Σ_n^0-completeness) that B is an A-r.e. set.

Exercise 3.19. Assume that $A = \{a_0, a_1, a_2, \ldots\}$, $a_0 = 1$ and $a_{k+1} \in \{2a_k, 2a_k + 1\}$ for all k. Let n be any number. Show that A is a Σ_n^0-set iff A is a Π_n^0-set.

Exercise 3.20. Show that the set $\{x : |W_x| > x\}$ is many-one complete. First show that this set is r.e. and then make a many-one reduction from $\mathbb{K}$ onto this set.

Show $\{x : |W_x| = 0 \vee |W_x| = 2 \vee |W_x| = 4 \vee |W_x| = 6 \vee |W_x| = 8\} \leq_m \{x : |W_x| = 1 \vee |W_x| = 3 \vee |W_x| = 5 \vee |W_x| = 7 \vee |W_x| = 9\}$; that is, construct a many-one reduction from the first set to the second set. Is there also a reverse many-one reduction? Give some ideas to justify your answer, no full proof required.

4 Post's Problem and Reducibilities

Complete r.e. sets are the most natural examples for sets which are r.e. but not recursive. They are all creative and many-one equivalent; hence they are also Turing equivalent. Because of this fact, Post [84] was interested in the question whether all nonrecursive r.e. sets are Turing equivalent to the halting problem, that is, Turing complete. Post studied this question with respect to two aspects: are there incomplete and nonrecursive r.e. sets for other reducibilities and are there structural properties which enforce incompleteness? One necessary condition for many-one completeness is the following.

Proposition 4.1 [84]**.** *If A is many-one complete then there is an infinite recursively enumerable set B which is disjoint to A.*

Proof. As a many-one complete set is creative, there is a recursive function g with $g(x) \notin A \cup W_x$ whenever W_x is disjoint to A. By Theorem 2.3 there is also a function h such that for all x, $W_{h(x)} = W_x \cup \{g(x)\}$. Now let e_0 be an index for the empty set, $e_{n+1} = h(e_n)$ and $B = \bigcup_n W_{e_n} = \{g(e_n) : n \in \mathbb{N}\}$. By induction one can show that W_{e_n} has n elements and is disjoint to A: assuming that W_{e_n} consists of n numbers not in A, $g(e_n)$ is outside $A \cup W_{e_n}$ and $W_{e_{n+1}}$ has the n elements from W_n plus the element $g(e_n)$. The union B of all sets W_{e_n} is disjoint from A as well. Furthermore, B is recursively enumerable and infinite. ∎

This motivated Post to construct a nonrecursive m-incomplete r.e. set by exploiting this method. This set is coinfinite and intersects every infinite r.e. set. Post [84] called such sets *simple*.

Theorem 4.2: Post's Simple Sets [84]**.** Call a set *simple* if it is recursively enumerable, coinfinite and intersects every infinite r.e. set. Then there is a simple set; in particular there is a nonrecursive and m-incomplete r.e. set.

Proof. Let $I_0, I_1, I_2, \ldots$ be a recursive partition of the natural numbers into intervals such that I_n contains more than n elements, for example, $I_n = \{2^n - 1, 2^n, \ldots, 2^{n+1} - 2\}$. Now define $\psi(\langle e, n\rangle)$ by the following algorithm:

If $e < n$ then search the first s such that $W_{e,s} \cap I_n \neq \emptyset$ and let $\psi(\langle e, n\rangle) = \min(W_{e,s} \cap I_n)$. If $e \geq n$ or the previous search for s does not terminate then let $\psi(\langle e, n\rangle)$ be undefined.

Note that, for each n, $\psi(\langle e, n\rangle)$ is defined only for at most n numbers e which are below n and $\psi(\langle e, n\rangle) \in I_n$ whenever $\psi(\langle e, n\rangle)$ is defined. Now let A be the range of ψ. A has the following properties:

- A is recursively enumerable as A is the range of a partial-recursive function;
- A is coinfinite as for every n, $|A \cap I_n| \leq n < |I_n|$;
- A has with every infinite r.e. set W_e an infinite intersection as every I_n with $n > e$ satisfies either $W_e \cap I_n = \emptyset$ or $\psi(\langle e, n \rangle)\downarrow \in A \cap W_e \cap I_n$.

Thus A is a simple set, it follows from Theorem 4.1 that A is not m-complete. ∎

Exercise 4.3. Uspenskii [116] called an r.e. set A *pseudocreative* iff for every r.e. set B disjoint to A there is an infinite r.e. set C disjoint to $A \cup B$. Furthermore, Uspenskii called an r.e. set A *pseudosimple* iff A is not recursive and there is an infinite r.e. set B such that B is disjoint to A and $A \cup B$ is simple. Bulitko [12] investigated the completeness of pseudosimple sets.

Given a simple set D, show that the set $\{2x : x \in D\}$ is pseudosimple. Given an r.e. set E which is neither recursive nor creative, show that the set $\{\langle x, y \rangle : x \in E \wedge y \in \mathbb{N}\}$ is pseudocreative but not creative. Show that every r.e. set is either recursive or simple or pseudosimple or pseudocreative but that no r.e. set satisfies two of these properties. Show that every r.e. many-one degree contains either a recursive set or a pseudocreative set.

Note that the terminology is inconsistent: simple sets are never pseudosimple but creative sets are pseudocreative. Dekker [27] called a set *mezoic* if it is either pseudosimple or pseudocreative but not creative. So the above two sets are two different type of examples of mezoic sets.

Exercise 4.4. Show that if A is simple and W_e infinite then $A \cap W_e$ is also infinite. Furthermore, define $\theta(e)$ to be the first element enumerated into $W_e - \bigcup_{n \leq e} I_n$ whenever such an element exists; $\theta(e)$ is undefined otherwise. Show that the range of θ is a simple set.

For Post this was the starting point of further investigations. Post [84] introduced the truth-table reducibility and showed that certain r.e. sets are neither recursive nor truth-table complete. Motivated by this result, researchers started to study also further reducibilities: Bulitko [11] introduced linear reducibility and Jockusch [44] conjunctive, disjunctive and positive reducibility.

A truth-table reduction reduces a set A to a set B as follows: Given an x, it produces a truth-condition involving B at finitely many places such that $x \in A$ iff that truth-condition is true. Such a condition is called a "truth-table". An example of such a truth-table is

$$16 \in A \Leftrightarrow 2 \in B \vee (5 \in B \wedge 6 \notin B) \vee (256 \notin B).$$

This condition is build up by using and ($\wedge$), or ($\vee$) and not (implicitly by $\notin$) on atomic formulas of the form "$c \in B$" for various constants c which depend on x. While the truth-table reduction permits any formula which can be built this way, the others are more restrictive: positive reducibility does not permit negation, conjunctive uses only "and", disjunctive uses only "or" and linear uses only "exclusive or". One can also define these reductions using canonical indices as defined in Definition 3.8 in order to get a more compact definition.

Definition 4.5. A set A is *truth-table reducible* to B iff there are two recursive functions f, g such that $A(x) = g(\langle x, a\rangle)$ where a is the canonical index of the set $D_{f(x)} \cap B$. Here f decides which elements are queried and g evaluates the resulting truth-table condition.

A set A is *positive reducible* to B iff there are two recursive functions f, g such that $A(x) = g(\langle x, a\rangle)$ where a is the canonical index of the set $D_{f(x)} \cap B$ with the additional constraint that for all x, b, c, whenever $D_b \subseteq D_c$ then $g(\langle x, b\rangle) \leq g(\langle x, c\rangle)$. This monotonicity condition reflects the absence of negation in the positive truth-table condition.

A set A is *conjunctive reducible* to B iff there is a recursive function f such that $x \in A$ iff $D_{f(x)} \subseteq B$. This condition is equivalent to taking the "and" of all conditions $y \in B$ with $y \in D_{f(x)}$.

A set A is *disjunctive reducible* to B iff there is a recursive function f such that $x \in A$ iff $D_{f(x)} \cap B \neq \emptyset$. This condition is equivalent to taking the "or" of all conditions $y \in B$ with $y \in D_{f(x)}$.

A set A is *linear reducible* to B iff there is a recursive function f such that $x \in A$ iff the cardinality of $D_{f(x)} \cap B$ is odd. This condition is equivalent to taking the "exclusive or" of all conditions $y \in B$ with $y \in D_{f(x)}$.

One writes $A \leq_{tt} B$, $A \leq_p B$, $A \leq_c B$, $A \leq_d B$ and $A \leq_l B$ for A being truth-table, positive, conjunctive, disjunctive and linear reducible to B. Similarly one defines tt-complete, p-complete, c-complete, d-complete and l-complete sets.

Exercise 4.6. Let A, B be sets of natural numbers. Show the following:

- if $A \leq_m B$, $A \leq_c B$, $A \leq_d B$ or $A \leq_p B$ and B is recursively enumerable, so is A;
- the set $\mathbb{N} - \mathbb{K}$ is not recursively enumerable but $\mathbb{N} - \mathbb{K} \leq_l \mathbb{K}$ and $\mathbb{N} - \mathbb{K} \leq_{tt} \mathbb{K}$;
- if $A \leq_{tt} B$, $A \leq_p B$, $A \leq_c B$, $A \leq_d B$, $A \leq_l B$ or $A \leq_m B$ and B is recursive, so is A.

Which of these reducibilities preserve that a set is co-r.e., that is, given a co-r.e. set B then every set reducible to B is also a co-r.e. set?

Exercise 4.7. Show the following facts:

- $A \leq_c B$ iff $\mathbb{N} - A \leq_d \mathbb{N} - B$;
- $A \leq_p B$ iff $\mathbb{N} - A \leq_p \mathbb{N} - B$;
- $A \leq_{tt} B$ iff $A \leq_p \{2x : x \in B\} \cup \{2x+1 : x \notin B\}$.

Can a similar connection be shown for many-one reducibility?

Note that whenever A is reducible by one of these reductions to B then A is truth-table reducible to B. Friedberg and Rogers [36] observed that the notion of reducibilities can also be weakened. For a weak reducibility, convergence relative to all oracles is no longer required.

Definition 4.8 [36]**.** A set A is *weakly truth-table reducible* to B iff there is a total recursive function f and a partial-recursive function g such that $g(\langle x, a\rangle)$ is defined and equal to $A(x)$ whenever a is the canonical index of the set $D_{f(x)} \cap B$. Here f decides which elements are queried and g evaluates the outcoming weak truth-table condition; $g(\langle x, a\rangle)$ can be undefined if $D_a \neq D_{f(x)} \cap B$. One writes $A \leq_{wtt} B$ for A being weak truth-table reducible to B and an r.e. set B is wtt-complete iff $\mathbb{K} \leq_{wtt} B$.

Post defined hypersimple sets which are neither complete for any strong reducibility [84] nor for weak truth-table reducibility [36]. Post hoped that hyperhypersimple sets (which form a subclass of the hypersimple sets) would always be incomplete for Turing reducibility, but it turned later out that this is not the case.

Definition 4.9. A set A is *hyperimmune* iff A is infinite and there is for every recursive function f a number x such that no y with $x \leq y \leq f(x)$ is in A. A set A is *hypersimple* iff A is recursively enumerable and its complement is hyperimmune. A set A is *bihyperimmune* iff both A and its complement are hyperimmune.

A set A is *hyperhyperimmune* iff A is infinite and there is no recursive function f such that

- $W_{f(x)}$ is finite for all x;
- $W_{f(x)} \cap W_{f(y)} = \emptyset$ for all different x, y;
- $W_{f(x)} \cap A \neq \emptyset$ for all x.

A set A is *hyperhypersimple* iff A is recursively enumerable and its complement is hyperhyperimmune.

A set A is *strongly hyperhyperimmune* iff A is infinite and there is no recursive function f such that

- $W_{f(x)} \cap W_{f(y)} = \emptyset$ for all different x, y;
- $W_{f(x)} \cap A \neq \emptyset$ for all x.

The strongly hyperhypersimple sets coincide with the hyperhypersimple ones, hence there is no need to define "strongly hyperhypersimple".

The main reason for Post to introduce these sets was that he could prove that hypersimple sets are not tt-complete; Friedberg and Rogers [36] extended the result and showed that hypersimple sets are also not wtt-complete.

Theorem 4.10: Incompleteness of Hypersimple Sets [36, 84]**.** *There is a hypersimple set E. Hypersimple sets are not weak truth-table complete and hence also not complete for any strong reducibility.*

Proof. Let $\Phi_e(z)$ be a Blum complexity measure such that $\Phi_e(z)$ is defined iff $\varphi_e(z)$ is defined and $\varphi_e(z) \leq \Phi_e(z)$ for all pairs (e, z) where $\varphi_e(z)$ is defined. Let

$$E = \{\langle x, y\rangle : y \neq x + \max\{\Phi_e(z) : e, z \leq x \wedge \varphi_e(z) \text{ is defined}\}\}$$

and note that the set E is recursively enumerable. A pair $\langle x, y\rangle$ is enumerated into E if one of the two following conditions are satisfied:

- there are no $e, z \leq x$ with $y = x + \Phi_e(z)$;
- there are $e, z \leq x$ with $y < x + \Phi_e(z)$.

The first condition can be checked by bounded search as $\{(e, z, t) : \Phi_e(z) = t\}$ is decidable and the second condition is a Σ^0_1-condition. Let $h(x)$ be the unique y such that $\langle x, y\rangle \notin E$. Now $x < x'$ implies $h(x) < h(x')$. From the monotonicity of $\langle \cdot, \cdot \rangle$ can also be concluded that $x < x'$ implies $\langle x, h(x)\rangle < \langle x', h(x')\rangle$.

Now assume by way of contradiction that E is not hypersimple. Then there is a recursive function f such that

$$f(\langle x, h(x)\rangle + 1) \geq \langle x + 1, h(x + 1)\rangle$$

for all x; this property must hold as $\langle x + 1, h(x + 1)\rangle$ is the smallest nonelement of E which is strictly larger than $\langle x, h(x)\rangle$. Without loss of generality f is strictly monotonic increasing. Now let $g(0) = h(0)$ and $g(x + 1) = f(\langle x, g(x)\rangle)$. Then a straightforward induction shows that $h(x) \leq g(x)$ for all x. As f is recursive, so is g. Hence there is an index e with $g = \varphi_e$. For $x > e$,

$$h(x) \geq x + \Phi_e(x) \geq x + \varphi_e(x) > g(x)$$

in contradiction to $h(x) \leq g(x)$, hence f and g cannot exist. Thus E is against the assumption hypersimple.

For the second part, assume by way of contradiction that F is a wtt-complete hypersimple set. Let A be Post's simple set as given in Theorem 4.2, not as given in Exercise 4.4. Let $I_0, I_1, I_2, \ldots$ be the intervals from the construction in Theorem 4.2. Let $B = \{\min(I_n - A) : n \in \mathbb{N}\}$; note that for every n the interval I_n is not a subset of A and hence $\min(I_n - A)$ exists. The set B is truth-table reducible to A and hence also weak truth-table reducible to $\mathbb{K}$. The assumption $\mathbb{K} \leq_{wtt} F$ implies that $B \leq_{wtt} F$. Let f, g be the functions of the wtt-reduction from B to F as in Definition 4.8. Let

$$h(x) = \max\{\max(D_{f(z)}) : \exists b\ [D_b \subseteq \{0, 1, 2, \ldots, x\} \wedge z \in I_b]\}.$$

There is an r.e. set $W_e = \{x : \psi(x)\downarrow = 1\}$ for the partial-recursive function ψ which is defined on input u as follows:

- Let b be the index of the interval with $u \in I_b$;
- Let a be the canonical index of $D_{f(u)} - D_b$;
- If $g(\langle u, a\rangle) = 0$ then $\psi(u)\downarrow = 0$;
- If $g(\langle u, a\rangle) = 1$ and $\psi(u')\downarrow = 0$ for all $u' \in I_b \cap \{0, 1, \ldots, u-1\}$ then $\psi(u)\downarrow = 1$;
- Otherwise $\psi(u)\uparrow$.

Note that $|W_e \cap I_b| \leq 1$ for all b. The basic idea is to choose W_e such that using the assumption that f, g describe a wtt-reduction from B to F it holds that for some $b > e$ that D_b contains all elements queried which are outside F and hence $W_e \cap I_b = B \cap I_b$.

As F is hypersimple, there are infinitely many x such that $y \in F$ for all $y \in \{x, x+1, \ldots, h(x)\}$; as F is co-infinite one can choose such an x to be so large such that in addition the canonical index b of $D_b = \{0, 1, 2, \ldots, x\} - F$ satisfies $b > e$. Let u be the unique element of $B \cap I_b$. Note that for any $u' \in I_b$ and any a' satisfying $D_{a'} = D_{f(u')} - D_b$ it holds that $B(u') = g(\langle u', a'\rangle)$ as f, g are a wtt-reduction from B to F and $D_{a'} = F \cap D_{f(u')}$. It follows that $W_e \cap I_b = \{u\}$. Hence $u \in A$ by the construction of A in Theorem 4.2 and $u \notin B$ by the choice of B. This contradicts that $g(\langle u, a\rangle) = 1$ for the a with $D_a = D_{f(u)} - D_b = D_{f(u)} \cap F$. Hence the wtt-reduction f, g from B to F cannot exist and F is not wtt-complete. ∎

Remark 4.11. The hypersimple set E constructed in Theorem 4.10 is Turing complete: To check whether $e \in \mathbb{K}$ one has to check whether $\varphi_e(e)$ is defined. This can be done by computing $h(e)$ relative to E which is the unique e' with $\langle e, e'\rangle \notin E$. Then $\varphi_e(e)$ is defined iff $\Phi_e(e) \leq h(e)$; the latter can be checked explicitly as Φ is a Blum complexity measure and $\{\langle c, d, x\rangle : \Phi_d(x) \leq c\}$ is recursive. So $\mathbb{K} \leq_T E$.

Exercise 4.12. Assume that A is r.e. and assume that there is a recursive function f such that

- $W_{f(x)} \not\subseteq A$ for all x;
- $W_{f(x)} \cap W_{f(y)} = \emptyset$ for all different x, y.

Show that A is not hyperhypersimple. That is, there is a recursive function g such that

- $W_{g(x)}$ is finite for all x;
- $W_{g(x)} \not\subseteq A$ for all x;
- $W_{g(x)} \cap W_{g(y)} = \emptyset$ for all different x, y.

Hence the notions of hyperhypersimple and strongly hyperhypersimple coincide.

Exercise 4.13. The set $A \oplus B = \{2x : x \in A\} \cup \{2y+1 : y \in B\}$ is called *the join* of A and B. Then $A \oplus B$ is the least upper bound of A and B with respect to m-reducibility: First A is m-reducible to $A \oplus B$ via the mapping $x \mapsto 2x$ and B is m-reducible to $A \oplus B$ via the mapping $y \mapsto 2y+1$. Second assume that $A \leq_m D$ via f and $B \leq_m D$ via g. Now let $h(2x) = f(x)$ and $h(2x+1) = g(x)$. It follows that $(A \oplus B)(2x) = A(x) = D(f(x)) = D(h(2x))$ and $(A \oplus B)(2x+1) = B(x) = D(g(x)) = D(h(2x+1))$, hence $(A \oplus B)(y) = D(h(y))$ for all y and $A \oplus B \leq_m D$ via h. Hence $A \oplus B$ is the least upper bound for m-reducibility.

Furthermore, if $A \leq_m B$ via f and $B \leq_m C$ via g then $A \leq_m C$ via $x \mapsto g(f(x))$: $x \in A \Leftrightarrow f(x) \in B \Leftrightarrow g(f(x)) \in C$. Hence m-reducibility is transitive.

Select one of the other reducibilities (conjunctive, disjunctive, linear, truth-table, weak truth-table and Turing) and transfer the proofs given above to that reducibility.

Theorem 4.14. *$A \leq_{tt} B$ iff there is a Turing machine M such that M^B computes A and M^C is total for every oracle C.*

Remark 4.15. One says that a reducibility from A to B is bounded iff there is a constant c such that for every $x \in A$ there are c numbers $y_1, \ldots, y_c$ such that $A(x)$ depends only on the values $B(y_1), \ldots, B(y_c)$ and not on any value $B(z)$ with $z \notin \{y_1, y_2, \ldots, y_c\}$.

For the reducibilities in Definition 4.5 the additional requirement would be that $D_{f(x)}$ contains at most c elements for all x. Similarly one can define bounded weak truth-table reducibility which, depending on the authors, is either abbreviated as bwtt-reducibility or just bw-reducibility. Furthermore, one can define that A is

bounded Turing reducible to B ($A \leq_{bT} B$) iff there is a program which defines how to compute A relative to B such that there is a constant c so that for every input x the program makes at most c oracle queries about the value of B at some place y [8, 40]. Note that these oracle queries can depend on the outcome of previous queries.

Comprehensive Exercise 4.16. Construct sets A and B such that $A \leq_{bT} B$ but $A \not\leq_{wtt} B$. Let $a_0, a_1, a_2, \ldots$ be a recursive one-one enumeration of $\mathbb{K}$. The proof would be based on the following steps.

- Let A depend on B as follows: $\langle x, y\rangle \in A$ iff $y \in \mathbb{K}$ and $\langle x, s+1\rangle \in B$ for the unique number s with $y = a_s$; show that this guarantees that $A \leq_{bT} B$ with two queries.
- Let $\langle x, 0\rangle \in B$ iff $x \in \mathbb{K}$; this is to code $\mathbb{K}$ into B.
- Consider a recursive enumeration (f_x, g_x) of wtt-reductions; let $\langle x, s+1\rangle \in B$ iff $f_x(\langle x, a_s\rangle)$ is defined, $\max(D_{f_x(\langle x,a_s\rangle)}) < \langle x, s+1\rangle$ and $g_x(\langle x, b\rangle)\downarrow = 0$ for the b satisfying $D_b = B \cap D_{f_x(\langle x,a_s\rangle)}$; show that this provides an inductive definition of B.
- Analyze the previous conditions to conclude that $A \not\leq_{wtt} B$.

Exercise 4.17. Let A, B be recursively enumerable sets. Show that $A \leq_{bT} B$ iff there is a constant c and a recursive function f such that for every x,

- $W_{f(x)}$ has at most c elements;
- D_u has at most c elements for all $u \in W_{f(x)}$;
- $x \notin A$ iff there is an $u \in W_{f(x)}$ with $D_u \cap B = \emptyset$.

This characterization of bounded Turing reducibility restricted to r.e. sets was found by Odifreddi [79, page 340], although the second condition was missing due to a typing-error [81].

Exercise 4.18. Show that one can formalize tt-reducibility and wtt-reducibility also as follows: A is tt-reducible (wtt-reducible) to B via f, g iff there is a recursive function f and a further recursive function (partial recursive function) g such that $g(x, B(0)B(1)\ldots B(f(x)))$ is defined and equal to $A(x)$ for all x. Here the binary string $B(0)B(1)\ldots B(y)$ contains the first $y+1$ values of the characteristic function of B.

Remark 4.19. One can generalize conjunctive and disjunctive reducibility by considering r.e. indices of sets instead of canonical indices of finite sets: A set A is Q-reducible to B iff there is a recursive function f such that $x \in A \Leftrightarrow W_{f(x)} \subseteq B$ for all x; A is s-reducible to B iff there is a recursive function f such that $x \in A \Leftrightarrow W_{f(x)} \cap B \neq \emptyset$ for all x. Note that $A \leq_Q B \Leftrightarrow \mathbb{N} - A \leq_s \mathbb{N} - B$. These reducibilities do in general not imply Turing reducibility; for example, $\mathbb{K}$ is s-reducible to every non-empty r.e. set and similarly $\mathbb{N} - \mathbb{K}$ is Q-reducible to every non-empty co-r.e. set. Nevertheless, Q-complete r.e. sets are also Turing complete.

Remark 4.20. Odifreddi [79, Section III.9] gives an overview of the complete sets. He showed that for many-one, bounded truth-table, disjunctive, conjunctive, positive, truth-table, weak truth-table, Q and Turing reducibilities one can deduce all inclusions from the following rules:

- m-complete sets are c-complete and btt-complete;
- btt-complete sets are d-complete;
- c-complete sets are Q-complete and p-complete;
- p-complete sets are tt-complete;
- tt-complete sets are wtt-complete;
- Q-complete sets are T-complete;
- wtt-complete sets are T-complete.

For example, by applying the second, fourth and fifth rule, one gets that every btt-complete set is wtt-complete. On the other hand, some c-complete set is not d-complete as this cannot be deduced from the above rules. Indeed, Post's simple set has a superset which is c-complete but not but not for m-complete. Given the set A from Theorem 4.2, the c-complete set B is given as the union of A and all I_n with $n \in \mathbb{K}$. Then $n \in \mathbb{K} \Leftrightarrow I_n \subseteq B$.

Exercise 4.21. Show that no simple set A is d-complete. Consider a creative set B and a recursive function f such that $f(x) \notin A \cup W_x$ whenever $A \cap W_x = \emptyset$. Assume that $B \leq_d A$ as witnessed by a recursive function g: $x \in B \Leftrightarrow D_{g(x)} \cap A \neq \emptyset$. Now prove that there is a recursive function h such that $W_{h(0)} = \{x : D_{g(x)} = \emptyset\}$ and $W_{h(n+1)} = \{x : D_{g(x)} \subseteq \cup_{y \in W_{h(n)} \cup \{f(h(n))\}} D_{g(y)}\}$. Show that the union E of all $W_{h(n)}$ is r.e. and that $\cup_{y \in E} D_{g(y)}$ is an infinite set disjoint to A; hence A is not simple.

5 The Theorem of Post and Kleene

Ten years after Post published his problem, Kleene and Post [52] published a partial solution: there is a set A such that $\emptyset <_T A <_T \mathbb{K}$; unfortunately this set A was not recursively enumerable. Today there is a wide variety to build such sets. Two methods will be presented: a direct construction of a pair of Turing incomparable sets below $\mathbb{K}$ and a construction of a 1-generic set G with jump $\mathbb{K}$.

Remark 5.1. It is convenient to fix a canonical enumeration $\eta_0, \eta_1, \eta_2, \ldots$ of all binary strings. Now let $\varphi_e^{\eta_a}(x)$ be defined whenever during the computation of $\varphi_e^{\eta_a}(x)$ the oracle is only asked at positions inside the domain of η_a and answered according to the values of the string η_a. One can show the following two connections:

- if $\varphi_e^{\eta_a}(x)$ is defined then $\varphi_e^B(x)$ is defined for all sets B which extend η_a;
- if $\varphi_e^B(x)$ is defined then the oracle B is queried only at finitely many places during this computation and thus there is a string η_a extended by B such that $\varphi_e^{\eta_a}(x)$ is defined.

Furthermore, there is a recursive function jump such that

$$W_{\text{jump}(e)} = \{a : \varphi_e^{\eta_a}(e)\downarrow\}$$

and $\varphi_e^B(e)$ is defined iff there are a, n with $\eta_a = B(0)B(1)\ldots B(n)$ and $a \in W_{\text{jump}(e)}$.

Theorem 5.2: Kleene's and Post's Turing-Incomplete Sets [52]**.** *There are two sets $A, B \leq_T \mathbb{K}$ such that $A \not\leq_T B$ and $B \not\leq_T A$.*

Proof. The sets A, B are build by finite extension. Let α_0, β_0 be both the string 0. Now the following is done in stages $2s+1$ and $2s+2$:

- Stage $2s+1$: Let x be the least number outside the domain of α_{2s}. If there is an η_a extending β_{2s} such that $\varphi_s^{\eta_a}(x)$ is defined then choose the least such a and the least $y \in \{0,1\} - \{\varphi_s^{\eta_a}(x)\}$ and let $\alpha_{2s+1} = \alpha_{2s}y$, $\beta_{2s+1} = \eta_a 0$ else let $\alpha_{2s+1} = \alpha_{2s}0$, $\beta_{2s+1} = \beta_{2s}0$.
- Stage $2s+2$: Let x be the least number outside the domain of β_{2s+1}. If there is an η_a extending α_{2s+1} such that $\varphi_s^{\eta_a}(x)$ is defined then choose the least such a and the least $y \in \{0,1\} - \{\varphi_s^{\eta_a}(x)\}$ and let $\alpha_{2s+2} = \eta_a 0$, $\beta_{2s+2} = \beta_{2s+1}0$ else let $\alpha_{2s+2} = \alpha_{2s+1}0$, $\beta_{2s+2} = \beta_{2s+1}0$.

Now the set A is defined as the limit of all α_s; the construction guarantees that $\alpha_s(x)$ is defined for all $s \geq x$ and so one can take $A(x) = \alpha_x(x)$ for all x. Furthermore, each stage of the construction can be carried out with oracle $\mathbb{K}$; this oracle is also needed for checking whether an η_a with the given property exists or not. Hence $A \leq_T \mathbb{K}$. Similarly one can define B as the limit of all β_s and verify that $B \leq_T \mathbb{K}$.

To see that $A \not\leq_T B$, consider any e such that φ_e^B is total and consider stage $2e+1$: Then there is an extension η_a of β_{2e} such that $\varphi_e^{\eta_a}(x)$ is defined for the least x outside the domain of α_{2e}. Furthermore, B extends η_a for the least such a. The value $A(x)$ is the least value y different from $\varphi_e^{\eta_a}(x)$ and hence $A \neq \varphi_e^B$. So there is no e such that $A = \varphi_e^B$ and therefore $A \not\leq_T B$. Similarly one shows that $B \not\leq_T A$. ∎

This is the direct construction. The next construction is a bit more indirect; it shows that there is a nonrecursive set G whose jump is $\mathbb{K}$; hence $\emptyset <_T G <_T \mathbb{K}$ and G has intermediate Turing degree. The construction does even something more general: it shows that there is for every $A \geq_T \mathbb{K}$ a set G such that $G' \equiv_T A$. Friedberg [34] found this jump inversion theorem in 1957. The set G constructed has a special property which is known as "1-generic".

Definition 5.3. A set G is 1-*generic* iff for every r.e. set W of strings there is a number n such that either $G(0)G(1)G(2)\ldots G(n) \in W$ or no string in W extends $G(0)G(1)G(2)\ldots G(n)$.

Theorem 5.4: Friedberg's Jump Inversion [34]**.** *For every set $A \geq_T K$ there is a 1-generic set G such that $G' \equiv_T A$. Such a set G is always nonrecursive. In particular there is a set G such that $\emptyset <_T G <_T \mathbb{K}$.*

Proof. Recall that the jump G' of G is the halting problem relative to G. The proof that the halting problem is unsolvable relativizes, that is, $G <_T G'$ for all sets G. The construction of G is done by finite extension, that is, in each stage a string γ_{e+1} is constructed which extends the previous string γ_e.

The construction starts with $\gamma_0 = A(0)$. In stage $e+1$ it is checked whether there is $a \in W_e$ such that η_a extends γ_e. This can be done with oracle A as $A \geq_T \mathbb{K}$. If so, $\gamma_{e+1} = \eta_a A(e+1)$ for the least such a; if not, $\gamma_{e+1} = \gamma_e A(e+1)$.

Let G be the unique set whose characteristic function extends all strings γ_e. As every x is in the domain of γ_x, one can also define G by letting $G(x) = \gamma_x(x)$ for all x. Now the desired properties of G are verified.

The set G is 1-generic: Consider any given r.e. set $\{\eta_a : a \in W_e\}$ of strings. If there is an extension η_a of γ_e with $a \in W_e$, then there is a least such a. Then $\gamma_{e+1} = \eta_a A(e+1)$ and $\eta_a = G(0)G(1)\ldots G(n)$ for some n; so $G(0)G(1)\ldots G(n)$ is in the given r.e. set of strings. If there is no extension η_a of γ_e with $a \in W_e$ then let

n be the number with $\gamma_e = G(0)G(1)\ldots G(n)$ and note that $G(0)G(1)\ldots G(n)$ has no extension in the given r.e. set of strings. This case distinction proves that G is 1-generic.

The set G is nonrecursive: For every recursive set R there is a set W_e such that $a \in W_e$ iff $\eta_a(k) \neq R(k)$ for some k in the domain of η_a. The set $\{\eta_a : a \in W_e\}$ consists of all strings which are incompatible with R and each string is extended by some string in $\{\eta_a : a \in W_e\}$. Thus γ_{e+1} satisfies that there is some k in the domain of γ_{e+1} such that $R(k) \neq \gamma_{e+1}(k)$. Hence $G \neq R$.

$A \leq_T G \oplus \mathbb{K}$: The main idea is that one can use G and $\mathbb{K}$ to trace the construction of the strings $\gamma_0, \gamma_1, \gamma_2, \ldots$ successively. The set $\mathbb{K}$ is used to decide whether one takes γ_{e+1} to be of the form $\eta_a A(e+1)$ or $\gamma_e A(e+1)$ and also to determine which string η_a is used in the first case. The set G is needed to determine the last bit of the string γ_{e+1} as that cannot be retrieved using $\mathbb{K}$ and the previously constructed bits. Furthermore, A can then be recovered from the sequence $\gamma_0, \gamma_1, \gamma_2, \ldots$ as $A(x)$ is the last bit of γ_x for all x. So one can reconstruct A from G and $\mathbb{K}$.

$A \leq_T G'$: This follows from $G \leq_T G'$, $\mathbb{K} \leq_T G'$ and the previous paragraph.

$G' \leq_T A$: For given e one can compute $\gamma_{\text{jump}(e)+1}$ using A and determine the a with $\eta_a A(\text{jump}(e)+1) = \gamma_{\text{jump}(e)+1}$. By construction, either $a \in W_{\text{jump}(e)}$ and $e \in G'$ or $b \notin W_{\text{jump}(e)}$ for all extensions η_b of η_a and $e \notin G'$. The test whether $a \in W_{\text{jump}(e)}$ can be carried out with oracle $\mathbb{K}$ and hence also with oracle A. So $G' \leq_T A$. Together with the previous paragraph one even has that $G' \equiv_T A$. ∎

Exercise 5.5. Assume that G is 1-generic. Let $G_e = \{x : \langle e, x\rangle \in G\}$ and show that $G_d \not\leq_T G_e$ whenever $d \neq e$. Conclude that there are infinitely many Turing degrees below $\mathbb{K}$.

Exercise 5.6. Define that a set A is limit-recursive iff there a sequence $A_0, A_1, A_2, \ldots$ of sets such that

- $\{\langle t, x\rangle : x \in A_t\}$ is recursive;
- $\forall x \exists t \forall s > t\ [A_s(x) = A(x)]$.

Show that a set is limit-recursive iff it is Turing reducible to $\mathbb{K}$. Note that the second condition is also often written as $\forall x \forall^\infty s\ [A_s(x) = A(x)]$ and that A_s is called the s-th approximation to A.

Exercise 5.7. Assume that A is limit-recursive and recursively approximated by $A_0, A_1, A_2, \ldots$ and let

$$c_A(x) = \min\{s \geq x : A_s(0)A_s(1)A_s(2)\ldots A_s(x) = A(0)A(1)A(2)\ldots A(x)\}.$$

The function c_A is called the convergence module of A. Let B be any set. Show that $A \leq_T B$ iff there is a B-recursive function f which majorizes c_A, that is, if there is a B-recursive f such that $c_A(x) \leq f(x)$ for all x.

Exercise 5.8. Let A be an r.e. and G be a 1-generic set such that $A \leq_T G$. Show that A is recursive.

As A is r.e., there is a recursive enumeration $A_0, A_1, A_2, \ldots$ of A in the sense that $A = \bigcup_s A_s$, $A_0 \subseteq A_1 \subseteq A_2 \subseteq \ldots$ and $\{\langle x, s\rangle : x \in A_s\}$ is recursive. Based on this enumeration, one can define c_A as in Exercise 5.7. Now the task is to show the following:

- If $A \leq_T G$ then $c_A = \varphi_e^G$ for some index e;
- If $c_A = \varphi_e^G$ then actually c_A is recursive.

The conclusion that with c_A also A is recursive is then straightforward as $x \in A$ iff $x \in A_{c_A(x)}$; the difficulty in Exercise 5.7 comes from the case where A is not r.e. but only limit-recursive.

Remark 5.9. A set A has hyperimmune Turing degree iff there is a hyperimmune set $B \leq_T A$. Otherwise A has a hyperimmune-free Turing degree. One can show that there are uncountably many sets of hyperimmune-free Turing degree. A set A of hyperimmune-free degree has several interesting properties:

- The Turing degree and tt-degree of A coincide; that is, for all sets B, $A \equiv_T B$ iff $A \equiv_{tt} B$.
- Every function $f \leq_T A$ is majorized by a recursive function g.
- $A \leq_T \mathbb{K}$ iff A is recursive; this can be proven by using Exercise 5.7.
- There is no 1-generic set $G \leq_T A$.

The first two conditions are a characterization, that is, if a set A satisfies one of them then the Turing degree of A is hyperimmune-free.

Sets of hyperimmune-free degree were first constructed by Martin and Miller [73]. If A has hyperimmune degree then there is a bi-immune set $B \equiv_T A$; that is, a set B such that both, B and $\mathbb{N} - B$, are immune. The same is true for some but not all sets A of hyperimmune-free degree. Indeed, there are still uncountably many sets of hyperimmune-free degree such that their Turing degree does not contain a bi-immune set.

Exercise 5.10. A binary recursive tree T is a recursive subset of $\{0,1\}^*$ such that for all binary strings $\sigma, \tau \in \{0,1\}^*$, if the concatenation $\sigma\tau \in T$ then also $\sigma \in T$. An

infinite branch of T is a set $A \subseteq \mathbb{N}$ such that $A(0)A(1)A(2)\ldots A(n) \in T$ for all n.

Show König's Lemma: If a binary tree T is infinite then it also has an infinite branch.

Show that there is an infinite binary recursive tree T without an infinite recursive branch. The idea is to let T contain all strings σ such that there is no x in the domain of σ with $\varphi_{x,|\sigma|}(x)\downarrow = \sigma(x)$; here $\varphi_{x,|\sigma|}(x)\downarrow$ iff the computation of $\varphi_x(x)$ halts within $|\sigma|$ steps.

Theorem 5.11 [73]**.** There is a set $A \leq_T \mathbb{K}'$ of hyperimmune-free Turing degree.

Proof. The central idea is to construct a descending sequence $T_0, T_1, T_2, \ldots$ of infinite recursive binary trees. For each tree T_e and each x the following set $Q_{e,x}$ can be defined which is a recursive subtree of T_e:

$$Q_{e,x} = \{\sigma \in T_e : \varphi^{\sigma}_{e,|\sigma|}(x)\uparrow\}.$$

The tree would only be co-r.e. and not recursive if the bound $|\sigma|$ on the computation time of $\varphi^{\sigma}_{e,|\sigma|}$ would not be there; hence this bound is necessary. One can check with oracle $\mathbb{K}$ with $Q_{e,x}$ is infinite for some e, x and one check with oracle $\mathbb{K}'$ whether $\exists x\,[Q_{e,x}$ is infinite$]$. The exact test would be whether

$$\exists x\,\forall t\,[Q_{e,x} \cap \{0,1\}^t \neq \emptyset]$$

holds; this Σ^0_2 formula can be checked with oracle $\mathbb{K}'$. Let T_0 be the recursive infinite tree without recursive branches from Exercise 5.10. Now the following construction is run for $e = 0, 1, 2, \ldots$ using oracle $\mathbb{K}'$:

- Check whether there is an x such that $Q_{e,x}$ is infinite.
- If so, let $S_e = Q_{e,x}$ else let $S_e = T_e$.
- Determine a value $A(e)$ such that $\{\sigma \in S_e : \sigma(e)\uparrow \text{ or } \sigma(e)\downarrow = A(e)\}$ is infinite.
- Let $T_{e+1} = \{\sigma : \sigma(e)\uparrow \text{ or } \sigma(e)\downarrow = A(e)\}$.

It is easy to see by induction that every tree T_e is infinite. Furthermore, one can show by induction that $A(0)A(1)A(2)\ldots A(e)$ is a string in tree T_{e+1}. From this it can be concluded that for all $n \leq e$ and $d \leq e$ also the string $A(0)A(1)A(2)\ldots A(n)$ is on the tree T_{e+1} and its supertree T_d. Hence A is an infinite branch of every tree T_e. As T_0 has no recursive infinite branch, A is nonrecursive.

Now consider any index e. If $S_e = Q_{e,x}$ for some x then $\varphi^A_e(x)$ is undefined as A is an infinite branch of T_{e+1} and $Q_{e,x}$. If $S_e = T_e$ then there are for every x only finitely

many strings $\sigma \in T_e$ such that $\varphi^{\sigma}_{e,|\sigma|}(x)$ is undefined; hence one can find for every x a level $\ell(x)$ such that $\varphi^{\sigma}_{x,|\sigma|}(x)$ is defined for all $\sigma \in \{0,1\}^{\ell(x)+1} \cap T_e$. The function f given as

$$f(x) = \max\{\varphi^{\sigma}_{x,|\sigma|}(x) : \sigma \in \{0,1\}^{\ell(x)+1} \cap T_e\}$$

is recursive and majorizes φ^A_e as $A(0)A(1)A(2)\ldots A(\ell(x))$ is among the strings σ considered in the maximum taken to compute $f(x)$. Hence φ^A_e is either partial or majorized by a recursive function. Therefore the Turing degree of A is hyperimmune-free. ∎

Remark 5.12. A set A has a *minimal Turing degree* if for every set $B \leq_T A$, either B is recursive or $B \equiv_T A$. So there is no set B with $\emptyset <_T B <_T A$. As seen before, the Turing degree of $\mathbb{K}$ is not minimal. Minimal Turing degrees exist and the first was constructed by Spector [107]. One might ask whether there are maximal degrees: the answer is "no" as above every degree there is the jump of this degree. So one might ask whether there is a set A of maximal degree in the sense that $A <_T \mathbb{K}$ and no set B satisfies $A <_T B <_T \mathbb{K}$; the answer is "no" again. Hence only minimal Turing degrees are interesting.

Exercise 5.13. Trakhtenbrot [113] defined that a set is *autoreducible* iff there is an index e such that

$$\forall x\ [A(x) = \varphi^{A \cup \{x\}}_e(x)].$$

In other words, a set is autoreducible iff $A(x)$ can be computed relative to A without querying A at x. In above definition, queries to A at x were made superfluous by using the oracle $A \cup \{x\}$. Given any A, the set $B = \{2x, 2x+1 : x \in A\}$ is autoreducible as one has $B(2x) = B(2x+1)$ for all x. Thus any Turing degree contains an autoreducible set. Show that no 1-generic set is autoreducible; hence there are many Turing degrees containing sets which are not autoreducible.

Remark 5.14. One can generalize the notion of 1-generic set to n-generic sets by defining that G is *n-generic* iff for every Σ^0_n set W there is an m such that either $G(0)G(1)G(2)\ldots G(m) = \eta_a$ for some $a \in W$ or $G(0)G(1)G(2)\ldots G(m)$ is not extended by any η_a with $a \in W$. There are n-generic sets for every n, but not every n-generic set is also $n+1$-generic. For example, all the 1-generic sets below $\mathbb{K}$ are not 2-generic. Furthermore, there are 2-generic but not 3-generic sets below $\mathbb{K}'$.

Downey and Yu [30] constructed a set of hyperimmune-free degree below a 1-generic set (with some additional other properties). But this cannot be done below a 2-generic set: If G is 2-generic and $A \leq_T G$ is nonrecursive then A has hyperimmune Turing degree.

6 The Fixed-Point Theorem and DNR Degrees

A recursive operator is an algorithm to translate functions into functions. An example of such an operator is relative computation: φ_e^A could be viewed as an operator translating a characteristic function of an oracle into some other function. In general, one can define an operator as follows.

Definition 6.1. An operator Ψ maps partial functions ϑ to partial functions $\Psi(\vartheta)$. An operator Ψ is a *recursive operator* iff there is an r.e. set A such that, for all partial functions ϑ and all x, y, $\Psi(\vartheta)(x)\downarrow = y$ iff there is a z such that $\langle x, y, z\rangle \in A$ and $\vartheta(v)\downarrow = w$ for all $\langle v, w\rangle \in D_z$. Defining $\text{graph}(\vartheta) = \{\langle v, w\rangle : \vartheta(v)\downarrow = w\}$, the latter condition can be simplified to $\Psi(\vartheta)(x)\downarrow = y \Leftrightarrow \exists z\,[\langle x, y, z\rangle \in A \wedge D_z \subseteq \text{graph}(\vartheta)]$.

Exercise 6.2. Which of the following mappings are recursive operators?

- $\Psi_1(\vartheta)(x) = \vartheta(x) + 1$ if $\vartheta(x)$ is defined and $\Psi_1(\vartheta)(x) = 0$ otherwise.
- $\Psi_2(\vartheta)(x) = x^2$.
- $\Psi_3(\vartheta)(x) = \vartheta(x) + 1$ if $\vartheta(x)$ is defined and $\Psi_3(x)$ is undefined otherwise.
- $\Psi_4(\vartheta)(x) = \vartheta(2x)+\vartheta(2x+1)$ if both values are defined and $\Psi_4(\vartheta)(x)$ is undefined otherwise.

If Ψ_k is a recursive operator then give also the corresponding r.e. set A_k of strings defining Ψ_k else say why Ψ_k is not a recursive operator.

Remark 6.3. A recursive operator translated partial functions into partial functions; an operator is called a *general recursive operator* if the image of a total function is always a total function. One can show that whenever Ψ is a recursive operator then there is a recursive function f with $\Psi(\varphi_e) = \varphi_{f(e)}$ for all e.

Kleene [51] proved the following result which is known as the first recursion theorem or fixed-point theorem.

Theorem 6.4: Kleene's Fixed-Point Theorem [51]**.** *Given a recursive operator Ψ there is a partial-recursive function ϑ such that*

- $\Psi(\vartheta) = \vartheta$*;*
- *if $\Psi(\theta) = \theta$ for some partial function θ then θ extends ϑ.*

Here "θ extends ϑ" means that $\mathrm{graph}(\vartheta) \subseteq \mathrm{graph}(\theta)$.

Proof. Let A be the r.e. set belonging to Ψ as defined in Definition 6.1. One defines partial functions $\gamma_0, \gamma_1, \ldots$ such that γ_0 is everywhere undefined and $\gamma_{n+1} = \Psi(\gamma_n)$. Then

$$\vartheta(x)\downarrow = y \Leftrightarrow \exists n\,[\gamma_n(x) = y]$$

is the desired fixed-point.

For the verification, it is first shown by induction that γ_{n+1} extends γ_n for all n. This is clear for $n = 0$. For the inductive step, one assumes that γ_{n+1} extends γ_n. Then, whenever $\Psi(\gamma_n)(x)$ is defined and takes a value y then there is a triple $\langle x, y, z\rangle \in A$ with $D_z \subseteq \mathrm{graph}(\gamma_n)$. As $\mathrm{graph}(\gamma_n) \subseteq \mathrm{graph}(\gamma_{n+1})$, $D_z \subseteq \mathrm{graph}(\gamma_{n+1})$ and $\Psi(\gamma_{n+1})(x)\downarrow = y$. Hence $\mathrm{graph}(\gamma_{n+1}) \subseteq \mathrm{graph}(\gamma_{n+2})$. Thus ϑ is the ascending limit of partial-recursive functions and well-defined.

Next it is shown that ϑ is partial-recursive. Given any input x, one can in parallel compute the indices of $\gamma_0, \gamma_1, \gamma_2, \ldots$ and simulate the computations of the $\gamma_n(x)$ until some computation converges with an output y; once this output is found, let $\vartheta(x) = y$. If these simulations and searches do not produce any output then $\vartheta(x)$ remains undefined.

Now it is shown that ϑ is a fixed-point. Whenever $\Psi(\vartheta)(x) = y$ then there is a z with $\langle x, y, z\rangle \in A$ and $D_z \subseteq \mathrm{graph}(\vartheta)$. As $\mathrm{graph}(\vartheta)$ is the ascending union of the sets $\mathrm{graph}(\gamma_n)$ and D_z is finite, there is an n such that $D_z \subseteq \mathrm{graph}(\gamma_n)$. Then $\Psi(\gamma_n)(x)\downarrow = y$ and $\gamma_{n+1}(x)\downarrow = y$. So $\vartheta(x)\downarrow = y$ and ϑ extends $\Psi(\vartheta)$. Furthermore, whenever $\vartheta(x)\downarrow = y$ then $\gamma_n(x)\downarrow = y$ and $\Psi(\gamma_n)(x)\downarrow = y$, hence $\Psi(\vartheta)(x)\downarrow = y$. Thus $\Psi(\vartheta)$ extends ϑ as well and $\Psi(\vartheta) = \vartheta$.

The last part is to show that every further fixed-point θ extends ϑ. So assume that $\Psi(\theta) = \theta$. Now one can show by induction that θ extends every γ_n: θ extends 0 and whenever θ extends γ_n, then $\Psi(\theta)$ extends $\Psi(\gamma_n) = \gamma_{n+1}$. As $\Psi(\theta) = \theta$, θ extends γ_{n+1} as well. Thus θ extends the union of all γ_n; that is, θ extends ϑ. ∎

The second recursion theorem (or fixed-point theorem) deals with indices of partial-recursive functions.

Theorem 6.5: Kleene's Second Fixed-Point Theorem [51]**.** *Let f be a recursive function. Then there is an e with $\varphi_{f(e)} = \varphi_e$.*

Proof. The proof is quite short (but hard to memorize): Let f be the given function and let g be a recursive function satisfying $\varphi_{g(c)} = \varphi_{\varphi_c(c)}$; the function $\varphi_{g(c)}$ is everywhere undefined if $\varphi_c(c)$ is undefined. Let d be an index of the composition $c \mapsto f(g(c))$. Then $\varphi_{\varphi_d(d)} = \varphi_{g(d)}$ by the choice of g. Furthermore, $\varphi_{f(g(d))} = \varphi_{\varphi_d(d)}$ as

d is an index of the mapping $c \mapsto f(g(c))$. It follows that $\varphi_{g(d)} = \varphi_{f(g(d))}$. Now taking $e = g(d)$, the equation $\varphi_e = \varphi_{f(e)}$ follows. ∎

An application of this theorem is to prove the missing direction of Theorem 3.4.

Theorem 6.6 [75]. *Every creative set is many-one complete.*

Proof. Given a creative set A, let f be the recursive function witnessing this fact, that is, f satisfies $f(x) \notin W_x \cup A$ whenever $W_x \cap A = \emptyset$. There is a recursive function g such that

$$W_{\varphi_{g(e)}(x)} = \begin{cases} \{f(\varphi_e(x))\} & \text{if } x \in \mathbb{K} \text{ and } \varphi_e(x)\downarrow; \\ \emptyset & \text{otherwise.} \end{cases}$$

Note that g can be chosen such that both g and $\varphi_{g(e)}(x)$ are total functions as one can make $W_{\varphi_{g(e)}(x)}$ to be the domain of the following partial-recursive function $h_{e,x}$: on input y, simulate $\varphi_e(x)$; if this halts, check whether $y = f(\varphi_e(x))$; if this is also true then $h_{e,x}(y) = y$ else $h_{e,x}(y)$ is undefined. Although $h_{e,x}$ itself is partial, $\varphi_{g(e)}(x)$ produces an index for $h_{e,x}$ and therefore $\varphi_{g(e)}$ is total.

By Kleene's recursion theorem, there is an e such that $\varphi_{g(e)} = \varphi_e$; as the function $\varphi_{g(e)}$ is total, so is the function φ_e. The remaining part is to show that the function $x \mapsto f(\varphi_e(x))$ is a many-one reduction from $\mathbb{K}$ to A. To see this, the following two cases are considered.

If $x \notin \mathbb{K}$ then $W_{\varphi_{g(e)}(x)} = \emptyset$ and thus $W_{\varphi_e(x)} = \emptyset$ by the choice of e as a fixed-point for g. As f witnesses that A is creative, $f(\varphi_e(x)) \notin A$.

If $x \in \mathbb{K}$ then $W_{\varphi_{g(e)}(x)} = W_{\varphi_e(x)} = \{f(\varphi_e(x))\}$ and $f(\varphi_e(x)) \in W_{\varphi_e(x)}$. Hence $W_{\varphi_e(x)}$ must intersect A as otherwise f would not witness that A is creative. This implies that $f(\varphi_e(x)) \in A$.

This completes the case-distinction and the verification that $\mathbb{K}(x) = A(f(\varphi_e(x)))$. So $x \mapsto f(\varphi_e(x))$ is a many-one reduction from $\mathbb{K}$ to A. As A is creative, A is an r.e. set. These two properties give that A is a many-one complete set. ∎

Remark 6.7. There are two further results related to Kleene's recursion theorem.

- A function is called *fixed-point free* if $W_{f(e)} \neq W_e$ for all e. The "set variant" of Kleene's fixed-point theorem says that all fixed-point free functions are nonrecursive.

- A function is called *diagonally nonrecursive* or just *dnr* if $f(e) \neq \varphi_e(e)$ for all $e \in \mathbb{K}$. The dnr functions are, as the name suggests, nonrecursive.

The second result that dnr functions are nonrecursive is trivial and does not seem to be that related to the fixed-point theorems; but when looking at A-recursive functions

instead of recursive functions, Jockusch, Lerman, Soare and Solovay [49] found a striking connection.

Theorem 6.8 [49]**.** *For every set A, there is an A-recursive dnr function f iff there is an A-recursive fixed-point free function g.*

Proof. Let an A-recursive dnr function f be given. Now consider a recursive function h such that $\varphi_{h(e)}(x)$ is defined iff $W_e \neq \emptyset$ and $\varphi_{h(e)}(x) \in W_e$ whenever $\varphi_{h(e)}(x)$ is defined. Such a function exists as one can choose $h(e)$ to be a program which searches inside the set W_e for an element; this program outputs the first of these elements which is found. Now define an A-recursive function g such that $W_{g(e)} = \{f(h(e))\}$. If $W_e = \{y\}$ then $\varphi_{h(e)}(h(e)) = y$ and $f(h(e)) \neq y$ and $W_{g(e)} \neq W_e$. If W_e has 0 or at least 2 elements then $W_{g(e)} \neq W_e$ as well. Hence g is fixed-point free.

Let an A-recursive fixed-point free function g be given. Now consider a recursive function h with $W_{h(e)} = W_{\varphi_e(e)}$ and let $f(e) = g(h(e))$. For $e \in \mathbb{K}$, $W_{f(e)} = W_{g(h(e))} \neq W_{h(e)} = W_{\varphi_e(e)}$ and hence $f(e) \neq \varphi_e(e)$. So f is an A-recursive dnr function. ∎

Remark 6.9. One says that a set A has *dnr Turing degree* iff there is an A-recursive dnr function. Jockusch [47] showed that whenever there is a $\{0, 1, 2, \ldots, c\}$-valued dnr function $f \leq_T A$ then there is also a $\{0, 1\}$-valued dnr function $g \leq_T A$. But there are sets of dnr Turing degree such that none of the dnr functions Turing reducible to these sets are bounded by a constant.

Arslanov [4] showed that among the r.e. sets, only the Turing complete ones can compute fixed-point free functions and thus only those sets have dnr Turing degree. Although Martin [65] and Lachlan [62] observed the same fact, the completeness criterion is named after Arslanov as he showed it also for other reducibilities like weak truth-table and many-one. For example, Arslanov showed that an r.e. set A is wtt-complete iff there is a function $g \leq_{wtt} A$ such that $W_e \neq W_{g(e)}$ for all e.

Theorem 6.10: Arslanov's Completeness Criterion [4]**.** *Given an r.e. set A there is an A-recursive fixed-point free function iff $\mathbb{K} \leq_T A$.*

Proof. By Theorem 6.8 it is enough to show that an r.e. set has dnr Turing degree iff it is Turing complete. In the case that $\mathbb{K} \leq_T A$, the following dnr function is A-recursive so that A has dnr Turing degree:

$$x \mapsto \begin{cases} \varphi_x(x) + 1 & \text{if } \varphi_x(x) \text{ is defined, that is, } x \in \mathbb{K}; \\ 0 & \text{otherwise.} \end{cases}$$

So assume for the other direction that A is r.e. and that there is an A-recursive dnr function φ_e^A. Let use(x) be the maximum of the numbers z satisfying one of the following two conditions:

- z is the number of steps to compute $\varphi_e^A(y)$ for some $y \leq x$;
- it was queried whether $z \in A$ during the computation of $\varphi_e^A(y)$ for some $y \leq x$.

Note that use is also A-recursive and $\text{use}(x)$ can be computed while knowing the oracle A only at the places $0, 1, \ldots, \text{use}(x)$. Furthermore, let A_s be the set of all elements enumerated into A within s steps and let c_A be the corresponding convergence module as defined in Exercise 5.7. There is a recursive function f such that

$$\varphi_{f(x)}(y) = \begin{cases} \varphi_e^{A_s}(y) & \text{for the first } s \text{ where } x \in \mathbb{K}_s \text{ and } \varphi_{e,s}^{A_s}(y)\downarrow; \\ \uparrow & \text{if there is no such } s\text{; that is, if } x \notin \mathbb{K}. \end{cases}$$

Let $x \in \mathbb{K}$ and s be the minimal number with $x \in \mathbb{K}_s$. As $\varphi_e^A(f(x)) \neq \varphi_{f(x)}(f(x))$ there must be an $y \leq \text{use}(f(x))$ with $y \in A - A_s$. Hence $s < c_A(\text{use}(f(x)))$. Thus

$$\forall x \, [x \in \mathbb{K} \Leftrightarrow x \in \mathbb{K}_{c_A(\text{use}(f(x)))}].$$

As c_A and use are A-recursive and f is a recursive function, it follows that $\mathbb{K} \leq_T A$. ∎

Exercise 6.11. Show that a 1-generic set G does not have dnr Turing degree. Consider any function φ_e^G which is total. Recall that $\eta_0, \eta_1, \eta_2, \ldots$ is a recursive enumeration of all binary strings. There is a recursive function f such that $\varphi_{f(a)}(x) = \varphi_e^\sigma(x)$ for the first extension σ of η_a found such that $\varphi_e^\sigma(x)$ is defined; if such a σ does not exist then $\varphi_{f(a)}(x)$ is undefined. Use the fact that G is 1-generic to prove that there is an a with $\varphi_{f(a)}(f(a))\downarrow = \varphi_e^G(f(a))$; hence G does not have dnr Turing degree.

Remark 6.12. The infinite branches of the binary tree constructed in Exercise 5.10 and used as starting point T_0 in Theorem 5.11 consists of all $\{0,1\}$-valued dnr functions; hence the characteristic function of the set of hyperimmune-free degree constructed in Theorem 5.11 is dnr and the set has dnr Turing degree. Sets of hyperimmune-free degree are not above the halting problem, so Arslanov's completeness criterion does not generalize to non-r.e. sets.

A further example of incomplete degrees are low degrees. Here a set A has *low Turing degree* if $A' \equiv_T \mathbb{K}$ and *high Turing degree* if $A' \geq_T \mathbb{K}'$. The high Turing degrees have also a further convenient characterization: A has high Turing degree iff there is a function $f \leq_T A$ which dominates every recursive function, that is, which satisfies $\forall^\infty x \, [f(x) > g(x)]$ for every recursive function g [66]. As there are uncountably many sets of high but only countably many sets of low degree, one has tried to generalize the notion "low" to "generalized low" which is defined as follows: A set A is *generalized low* iff $A' \leq_T A \oplus \mathbb{K}$. For example, all 1-generic sets are generalized low. Now a set is low iff it is generalized low and limit-recursive.

Note that although no high set is low, there are some high sets which are generalized low. Such sets can be obtained by Friedberg's Jump Inversion Theory: One takes

a 1-generic set G with $G' \equiv_T \mathbb{K}'$. Then this set G is high as $G' \geq_T \mathbb{K}'$. Furthermore it is generalized low since it is 1-generic.

Jockusch and Soare [50] showed the "Low Basis Theorem" which states that every infinite binary tree has an infinite branch which is low; they furthermore showed the "Hyperimmune-free Basis Theorem" which says that every infinite binary tree has an infinite branch of hyperimmune-free Turing degree. Taking the tree T_0 in the construction in Theorem 5.11 as the given tree, one can directly take over the proof of Theorem 5.11 to get this result. Also the proof of the Low Basis Theorem is a variant of this result.

Theorem 6.13: Jockusch's and Soare's Low Basis Theorem [50]**.** *Let T be an infinite recursive binary tree. Then T has an infinite branch A of low Turing degree.*

Proof. Recall from Remark 5.1 that the trees

$$Q_e = \{\sigma : \varphi^{\sigma}_{e,|\sigma|}(e)\!\uparrow\}$$

are uniformly recursive. Furthermore, one can test with oracle $\mathbb{K}$ whether a recursive tree R is infinite or finite by using the formula

$$R \text{ is finite } \Leftrightarrow \exists t\ [\{0,1\}^t \cap R = \emptyset].$$

The proof starts with the construction of a descending sequence $T_0, T_1, T_2, \ldots$ of infinite recursive binary trees such that $T_0 = T$ and T_{e+1} is the first of the following four trees which is infinite:

- $\{\sigma \in T_e \cap Q_e : \sigma(e)\!\uparrow \text{ or } \sigma(e)\!\downarrow = 0\}$;
- $\{\sigma \in T_e \cap Q_e : \sigma(e)\!\uparrow \text{ or } \sigma(e)\!\downarrow = 1\}$;
- $\{\sigma \in T_e : \sigma(e)\!\uparrow \text{ or } \sigma(e)\!\downarrow = 0\}$;
- $\{\sigma \in T_e : \sigma(e)\!\uparrow \text{ or } \sigma(e)\!\downarrow = 1\}$.

Note that one tree of these is always infinite, thus the algorithm finds for each T_e an infinite subtree T_{e+1}. Furthermore, let $A(e) = 0$ if T_{e+1} is selected by the first or third choice and let $A(e) = 1$ if T_{e+1} is selected by the second or fourth choice.

Recall that $\eta_0, \eta_1, \eta_2, \ldots$ was a list of all binary strings as defined in Exercise 5.10. The above construction also shows that there is a $\mathbb{K}$-recursive function f such that $\varphi_{f(e)}(a) = 1$ if $\eta_a \in T_e$ and $\varphi_{f(e)}(a) = 0$ if $\eta_a \notin T_e$.

It is easy to see by induction that every tree T_e is infinite. Furthermore, one can show by induction that $A(0)A(1)A(2)\ldots A(e)$ is a string in tree T_{e+1}. From this it

can be concluded that for all $n \leq e$ and $d \leq e$ also the string $A(0)A(1)A(2)\ldots A(n)$ is on the tree T_{e+1} and its supertree T_d. Hence A is an infinite branch of every tree T_e.

If $Q_e \cap T_e$ is infinite then all strings in T_{e+1} are also in Q_e and hence $\varphi^{\sigma}_{e,|\sigma|}(e)$ is undefined for all $\sigma \in T_{e+1}$. It follows that $\varphi^{A}_{e}(e)$ is undefined and $e \notin A'$.

If $Q_e \cap T_e$ is finite then almost all strings in T_{e+1} are outside Q_e. Hence there is an n such that $A(0)A(1)A(2)\ldots A(n) \in T_{e+1} - Q_e$ and $\varphi^{A}_{e}(e)$ is defined. Now $e \in A'$ and there is a $\sigma \in T_{e+1}$ with $\varphi^{\sigma}_{e,|\sigma|}(e)$ being defined.

So it holds that $e \in A'$ iff there is an a such that $\varphi_{f(e+1)}(a) = 1$ and $\varphi^{\eta_a}_{e,|\eta_a|}(e)$ is defined. This condition can be checked using $\mathbb{K}$ as an oracle. Hence $A' \leq_T \mathbb{K}$ and A has low Turing degree. Note that A can be recursive in the case that T_0 has recursive infinite branches. ∎

Exercise 6.14. One might ask whether the "Low Basis Theorem" has a counterpart called "High Basis Theorem" such that every recursive infinite tree has an infinite branch of high Turing degree. This is already wrong for the tree $\{0\}^*$ as this tree has exactly one infinite branch which is recursive. Hence the counterpart has to be stated as follows: An infinite recursive binary tree has either a recursive infinite branch or an infinite branch of high Turing degree. To prove this, do the following: Assume that T is an infinite recursive binary tree without infinite recursive branch. Construct a $\mathbb{K}$-recursive function $F : \{0,1\}^* \to T$ such that $F(\sigma)$ extends $F(\tau)$ iff σ extends τ. Show that there is an unique infinite branch A of T extending all nodes of the form $F(\mathbb{K}'(0)\mathbb{K}'(1)\mathbb{K}'(2)\ldots\mathbb{K}'(n))$. Furthermore, show that $\mathbb{K}' \equiv_T A \oplus \mathbb{K}$; hence A has high Turing degree.

Exercise 6.15. It follows from Remark 6.12 that every 1-generic set $G \leq_T \mathbb{K}$ is low. Prove this fact directly.

For this, use the limit-lemma to prove that there is a uniformly recursive sequence $G_0, G_1, G_2, \ldots$ of recursive sets such that for all x there is an t with $G_s(x) = G(x)$ for all $s > t$. Now use the property of 1-genericity to show the following: If $\varphi^{G}_{e}(e)$ is defined then, for almost all s, $\varphi^{G_s}_{e,s}(e)$ is defined. If $\varphi^{G}_{e}(e)$ is undefined then, for almost all s, $\varphi^{G_s}_{e,s}(e)$ is undefined. Conclude that this proves that G' is Turing reducible to the graph of f.

Exercise 6.16. Jockusch and Soare considered binary recursive trees. Show that this condition is necessary by constructing a recursive tree $T \subseteq \mathbb{N}^*$ which has infinite branches but no low infinite branches. The idea would be to build the tree such that whenever a function f is an infinite branch and whenever $x \in \mathbb{K}$ then $x \in \mathbb{K}_{f(x)}$. This would immediately show that $\mathbb{K}$ is Turing reducible to the graph of f and thus T has no infinite branch of low Turing degree.

7 A solution to Post's Problem for r.e. sets

Post problem was after 12 to 13 years solved independently by Friedberg [33] and Muchnik [74]. They showed that there is a pair of Turing incomparable r.e. sets. The method to construct them is called *priority method* and became one of the most successful proof-methods in recursion theory. The idea is that one wants to satisfy in a construction various requirements; each requirement has some priority and when satisfying a requirement R it is permitted to do changes which destroy goals obtained by requirements of lower priority but one is not permitted to do changes which destroy goals of requirements of higher priority. Furthermore, each goal has to be such that it can be satisfied with finitely many actions without destroying properties protected by requirements of higher priority. As there are for each requirement only finitely many requirements of higher priority, one can prove by induction along the priorities that each goal is eventually satisfied: first the goal of highest priority is satisfied within finitely many steps; then once any activity related to it stops, the goal of second-highest priority is satisfied in finitely many steps without making the goal of highest priority to become unsatisfied again; from then on the goal of third-highest priority will be satisfied in finitely many steps without making the two higher goals to be unsatisfied again; similarly for the further goals.

Theorem 7.1: Friedberg's and Muchnik's Solution of Post's Problem [33, 74]**.** *There are two r.e. sets of incomparable Turing degree. In particular there are r.e. sets which are neither recursive nor Turing complete.*

Proof. The sets are constructed by defining subsets A_s, B_s of A, B in each step s such that $A = \bigcup_s A_s$ and $B = \bigcup_s B_s$. The requirements are the following:

- Requirement R_{2e}: There is a number x_{2e} such that either $\varphi_e^B(x_{2e})\uparrow$ or $\varphi_e^B(x_{2e})\downarrow \neq A(x_{2e})$;
- Requirement R_{2e+1}: There is a number x_{2e+1} such that either $\varphi_e^A(x_{2e+1})\uparrow$ or $\varphi_e^A(x_{2e+1})\downarrow \neq B(x_{2e})$.

The priority of R_c is greater than the one of R_d iff $c < d$. So "higher priority" corresponds to "lower index". Without loss of generality it is assumed that for an index d, input x and an oracle E the approximation $\varphi_{d,s}^E(x)$ is defined only if the computation needs at most s steps and does not query the oracle E beyond s.

In stage 0 it is decided to initialize A_0, B_0 as the empty set and to let $x_d = d$ for all d; the values of x_d might change during the construction; $x_{d,s}$ denotes the value of x_d after stage s and before stage $s+1$. In stage $s+1$ one modifies A_s, B_s to A_{s+1}, B_{s+1} in such a way that the requirement of highest priority which is currently

not satisfied (with A_s, B_s used in place of A, B) becomes satisfied after the step. For the construction, the following notions are defined for requirements of the form R_{2e}:

- *Requirement R_{2e} needs attention when entering stage $s+1$* iff $\varphi_{e,s}^{B_s}(x_{2e,s})$ is defined and $x_{2e,s} \leq s$.
- *Requirement R_{2e} is satisfied when entering stage $s+1$* iff there is a $t \leq s$ such that $x_{2e,s} \leq t$, $\varphi_{e,t}^{B_s}(x_{2e,s})\downarrow \neq A_s(x_{2e,s})$ and $x_{d,s} > t$ for all $d > 2e$.
- *Requirement R_{2e} acts at stage $s+1$* iff the following is done: R_{2e} needs attention when entering stage $s+1$ and changes are done such that R_{2e} is satisfied when leaving stage $s+1$ (which equals entering stage $s+2$).

The definitions for requirements of odd index are parallel, only the roles of A and B are interchanged.

- *Requirement R_{2e+1} needs attention when entering stage $s+1$* iff $\varphi_{e,s}^{A_s}(x_{2e+1,s})$ is defined and $x_{2e+1,s} \leq s$.
- *Requirement R_{2e+1} is satisfied when entering stage $s+1$* iff there is a $t \leq s$ such that $x_{2e+1,s} \leq t$, $\varphi_{e,t}^{A_s}(x_{2e+1,s})\downarrow \neq B_s(x_{2e+1,s})$ and $x_{d,s} > t$ for all $d > 2e+1$.
- *Requirement R_{2e+1} acts at stage $s+1$* iff the following is done: R_{2e+1} needs attention when entering stage $s+1$ and changes are done such that R_{2e+1} is satisfied when leaving stage $s+1$ (which equals entering stage $s+2$).

The overall algorithm does the activities for stages $0, 1, 2, \ldots$ which are specified as follows.

- At stage 0, A_0 and B_0 are set to be the empty sets and every value $x_{d,s}$ equals d.
- At stage $s+1$, it is checked whether some requirement R_d needs attention but is not satisfied.
- If so, the least d is identified for which R_d needs attention and R_d is not satisfied. Then R_d acts and the following activities are carried out where the first part depends on whether d is even or odd.
  - Case $d = 2e$:
    If $\varphi_{e,s}^{B_s}(x_{d,s})\downarrow = 0$ then $A_{s+1} = A_s \cup \{x_{d,s}\}$ else $A_{s+1} = A_s$;
    $B_{s+1} = B_s$.

– Case $d = 2e + 1$:
If $\varphi_{e,s}^{A_s}(x_{d,s})\downarrow = 0$ then $B_{s+1} = B_s \cup \{x_{d,s}\}$ else $B_{s+1} = B_s$;
$A_{s+1} = A_s$.

For all c, if $c \leq d$ then $x_{c,s+1} = x_{c,s}$ else $x_{c,s+1} = c + s + 1$.

- If there is no requirement R_d which needs attention and is not satisfied, then no change is done, that is, $A_{s+1} = A_s$, $B_{s+1} = B_s$ and $x_{c,s+1} = x_{c,s}$ for all c.

For the verification, one shows the following properties (in italic font).

If R_{2e} acts at stage $s + 1$ and $A_s(x_{2e,s}) = 0$ then R_{2e} is satisfied at entering stage $s + 2$: Depending on the outcome of the computation of $\varphi_{e,s}^{B_s}(x_{2e,s})$ the element $x_{2e,s}$ is enumerated or not enumerated into A_{s+1} such that $A_{s+1}(x_{2e,s})$ differs from the outcome of the computation. Furthermore, $x_{c,s+1} = x_{c,s}$ for $c \leq 2e$ in order to keep requirements R_c satisfied whenever they are; $x_{c,s+1} = c + s + 1$ for $c > 2e$ so that the second condition for requirement R_{2e} to be satisfied is met.

If R_{2e+1} acts at stage $s + 1$ and $B_s(x_{2e+1,s}) = 0$ then R_{2e+1} is satisfied at entering stage $s + 2$: This is parallel to the previous proof.

If R_d is satisfied at entering stage $s + 1$ and no requirement R_c with $c < d$ acts at stage $s + 1$ then R_d is satisfied at entering stage $s + 2$ as well: Let $E = B$ if d is even and $E = A$ if d is odd. Let t, e be the parameters satisfying that $d \in \{2e, 2e + 1\}$ and $\varphi_{e,t}^{E_s}(x_{d,t})$ is defined and $x_{c,s} > t$ for all $c > d$; this t exists by the definition of being satisfied. Now in stage $s + 1$, R_d does not act as R_d is satisfied at entering the stage $s + 1$. Hence either a requirement R_c with $c > d$ might act at stage $s + 1$ or no requirement acts. Hence $x_{d,s+1} = x_{d,s}$ and $A_{s+1}(u) = A_s(u), B_{s+1}(u) = B_s(u)$ for all $u \leq \max\{t, x_{d,s}\}$. Furthermore, $x_{c,s+1} \geq x_{c,s}$ for all c and so the requirement R_d is also satisfied at entering stage $s + 2$.

Each requirement R_d acts at most 2^d times: Note that whenever a requirement R_d acts at stage $s + 1$ and later at stage $s' + 1$ then there must be a stage $s'' + 1$ with $s < s'' < s'$ such that requirement R_d is satisfied at entering stage $s'' + 1$ but not at entering stage $s'' + 2$. Hence some requirement R_c with $c < d$ must act at stage $s'' + 1$. Thus R_d can act at most once more than the number of times the requirements R_c with $c < d$ act. This permits to show the bound by induction: assume that all requirements R_c with $c < d$ act at most 2^c times; then these requirements act altogether at most $2^0 + 2^1 + 2^2 + \ldots + 2^{d-1} = 2^d - 1$ times and hence requirement R_d acts at most 2^d times.

For every d there is a value x_d such that $x_d = x_{d,s}$ for almost all s: This follows from the fact that $x_{d,s+1} \neq x_{d,s}$ only if some requirement R_c with $c < d$ acts at stage $s + 1$; this happens at most $2^d - 1$ times and thus almost all values $x_{d,s}$ are the same value x_d.

If $\varphi_e^B(x_{2e})$ is defined then it differs from $A(x_{2e})$: Assume that $\varphi_e^B(x_{2e})$ is defined

as otherwise there is nothing to prove. Let s be the first stage with $x_{2e,s} = x_{2e}$. Then $A_s(x_{2e,s}) = 0$ and no requirement R_c with $c < d$ acts at any stage $t > s$. For almost all $t > s$ it holds that $\varphi_{e,t}^{B_t}(x_{2e,t})$ is defined and $x_{2e,t} < t$, hence R_{2e} needs attention at almost all stages. If R_{2e} does not act at any stage $s'+1$ with $s' \geq s$ then R_{2e} becomes satisfied without having to act and hence $\varphi_e^B(x_{2e})$ differs from $A(x_{2e})$. If R_{2e} acts at some stage $s'+1$ with $s' \geq s$ then this is the unique of these stages where R_{2e} acts. Thus $A_{s'}(x_{2e,s'}) = 0$ and R_{2e} is satisfied at stage $s'+2$ by the first property shown. Then R_{2e} remains satisfied for all further stages and again $\varphi_e^B(x_{2e})$ differs from $A(x_{2e})$.

If $\varphi_e^A(x_{2e+1})$ is defined then it differs from $B(x_{2e+1})$: This is parallel to the previous condition.

$A \not\leq_T B$ and $B \not\leq_T A$: As there is no e such that $\varphi_e^B(x_{2e})\!\downarrow = A(x_{2e})$, the set A is not Turing reducible to B. Similarly B is not Turing reducible to A.

A and B are both recursively enumerable: This can directly be seen from the algorithm. For every s, $A_s \subseteq A_{s+1}$ and $B_s \subseteq B_{s+1}$. Hence $x \in A \Leftrightarrow \exists s\,[x \in A_s]$ and $x \in B \Leftrightarrow \exists s\,[x \in B_s]$. So A, B are Σ_1^0 and hence r.e. sets. ∎

Comprehensive Exercise 7.2. Trakhtenbrot [113] constructed an r.e. set which is not autoreducible. So one has to satisfy in the limit the following requirements:

- Requirement R_e: $\exists x\,[\varphi_e^{A\cup\{x\}}(x)$ is either undefined or different from $A(x)]$.

Work out a construction which satisfies these requirements in the limit and shows that some r.e. set is not autoreducible. The idea is to use movable markers $x_{e,s}$ and each time s where $\varphi_e^{A_s\cup\{x_{e,s}\}}(x_{e,s})$ converges one enumerates $x_{e,s}$ into A_{s+1} iff this spoils the outcome of the computation. The markers have to be moved in the case that markers of higher priority move as well.

Comprehensive Exercise 7.3. Soare [103] proved the existence of an intermediate r.e. Turing degree by constructing a nonrecursive and low r.e. set A. The basic idea is the following.

- Requirement N_e to make A nonrecursive: There is a number x_e with $A(x_e) \neq \varphi_e(x_e)$.
- Requirement L_e to make A low: If there are infinitely many stages s with $\varphi_{e,s}^{A_s}(e)$ being defined then $\varphi_e^A(e)$ is defined.

The number x_e is approximated with $x_{e,s}$ being the value of the approximation after stage s.

- *N_e needs attention at entering stage $s+1$* if $\varphi_{e,s}(x_{e,s})$ is defined and $x_{e,s} \leq s$.
- *N_e is satisfied at entering stage $s+1$* if $\varphi_{e,s}(x_{e,s})\!\downarrow \neq A_s(x_{e,s})$.

- *N_e acts at stage $s+1$* if $A_{s+1} = A_s \cup \{x_{e,s}\}$ and $\varphi_{e,s}(x_{e,s})\downarrow = 0$.
- *L_e needs attention at entering stage $s+1$* if $\varphi_{e,s}^{A_s}(e)$ is defined.
- *L_e is satisfied at entering stage $s+1$* if there is a t such that $\varphi_{e,t}^{A_s}(e)$ is defined and $x_{d,s} > t$ for all $d > e$.
- *L_e acts at stage $s+1$* if $x_{d,s+1} = x_{d,s}$ for all $d \leq s$ and $x_{d,s+1} = d + s + 1$ for all $d > s$.

Now at stage 0 the initialization is done by letting $A_0 = \emptyset$ and $x_{e,s} = e$ for all s. At stage $s+1$, one determines all requirements which need attention and are not satisfied. If no such requirement is found then no change occurs and one goes to $s+1$. If such requirements are found then the requirement of highest priority among these acts and one goes afterwards to stage $s+1$.

Formalize the construction and verify that the resulting set A is indeed a low and nonrecursive r.e. set.

Theorem 7.4. $\mathbb{K}$ is the disjoint union of two low r.e. sets.

Proof. Let f be a one-one recursive function with range $\mathbb{K}$. In each stage s, $\psi(f(s))$ will be defined such that afterwards $A_k = \{f(t) : \psi(f(t)) = k\}$ for $k = 0, 1$. The goal is to preserve a converging computation of the form

$$\varphi_{e,s}^{\{f(t):t<s\wedge\psi(f(t))=k\}}(e)$$

whenever possible. Here the set $\{f(t) : t < s \wedge \psi(f(t)) = k\}$ is the s-th approximation of A_k. Now one says that the e-th diagonal computation on A_k is injured iff

$$\varphi_{e,s}^{\{f(t):t<s\wedge\psi(f(t))=k\}}(e)\downarrow \wedge \varphi_{e,f(s)-1}^{\{f(t):t<s\wedge\psi(f(t))=k\}}(e)\uparrow .$$

Let $r(s)$ be the least number $2e+k$ such that the e-th diagonal computation on A_k is injured at stage s; more formally, let

$$\begin{aligned} r(s) \;=\; & \min\{2e+k : e \in \mathbb{N} \wedge k \in \{0,1\} \wedge \\ & \varphi_{e,s}^{\{f(t):t<s\wedge\psi(f(t))=k\}}(e)\downarrow \wedge \varphi_{e,f(s)-1}^{\{f(t):t<s\wedge\psi(f(t))=k\}}(e)\uparrow\}. \end{aligned}$$

Now one defines $\psi(f(s))$ such that one preserves the diagonal computation of highest priority which is injured by one choice; note that no diagonal computation is injured by both choices:

$$\psi(f(s)) = \begin{cases} 0 & \text{if } r(s) \text{ is odd;} \\ 1 & \text{if } r(s) \text{ is even.} \end{cases}$$

The resulting function ψ is a partial-recursive function with domain $\mathbb{K}$ and thus A_0, A_1 are r.e. sets. It remains to show that both sets are low.

One shows by induction that for every number $2e + k$ ($e \in \mathbb{N}, k \in \{0, 1\}$) the equivalences

$$e \in A'_k \Leftrightarrow \forall^\infty s\, [\varphi_{e,s}^{\{f(t):t<s\wedge\psi(f(t))=k\}}(e)\downarrow] \Leftrightarrow \exists^\infty s\, [\varphi_{e,s}^{\{f(t):t<s\wedge\psi(f(t))=k\}}(e)\downarrow]$$

hold. So assume by way of contradiction that $2e+k$ would be the least number where above equivalences fail. If $e \in A'_k$ then the computation accesses only finitely many members of A_k and so there is an upper bound q on the convergence time of the computation and all $f(t)$ such that $\psi(f(t)) = k$ and $f(t)$ had been queried by the computation. Hence, for all $s > q$,

$$\varphi_{e,s}^{\{f(t):t<s\wedge\psi(f(t))=k\}}(e)\downarrow$$

and so the two other conditions hold as well. For the other implication assume that

$$\exists^\infty s\, [\varphi_{e,s}^{\{f(t):t<s\wedge\psi(f(t))=k\}}(e)\downarrow].$$

By induction hypothesis one can take q so large such that there exists a q' satisfying for all $2d' + k' < 2e + k$ the following conditions: first, $\max\{t : f(t) \leq q'\} < q$; second, whenever there is an $s > q$ with

$$\varphi_{e',s}^{\{f(t):t<s\wedge\psi(f(t))=k'\}}(e')\downarrow$$

then $\varphi_{e',q'}^{A_{k'}}(e)$ converges. For that reason, $r(s) \geq 2e + d$ for all $s > q$. By assumption there is a is a stage $s > q$ with

$$\varphi_{e,s}^{\{f(t):t<s\wedge\psi(f(t))=k\}}(e)\downarrow;$$

this computation will never be destroyed by any assignment to ψ and so $\varphi_e^{A_k}(e)\downarrow$; that is, $e \in A'_k$. Hence all three conditions are equivalent. Hence $A'_0 \leq_T \mathbb{K}$ and $A'_1 \leq_T \mathbb{K}$ and A_0, A_1 are both low. ∎

Comprehensive Exercise 7.5. Sacks' splitting theorem [89] states that every non-recursive r.e. set A is the disjoint union of two low r.e. sets A_0, A_1 of incomparable Turing degree. As $\mathbb{K}$ is not low, A_0 and A_1 as constructed above must be Turing incomparable. Expand the proof of Theorem 7.4 to a full proof of Sacks' splitting theorem.

8 Maximal, r-maximal and semirecursive sets

This section deal with r.e. sets of a special form. Certain sets had already been introduced: *simple sets* which are co-infinite r.e. sets having a nonempty intersection with every infinite r.e. set and *hypersimple sets* where A is hypersimple iff A is co-infinite and for every recursive function f there is an n with $\{x : n \leq x \leq f(n)\} \subseteq A$. Alternatively one can also say that an r.e. coinfinite set A is hypersimple iff there is no recursive function f such that $D_{f(i)} \cap D_{f(j)} = \emptyset$ and $D_{f(i)} \not\subseteq A$ for all distinct i, j. Hyperhypersimple sets satisfy the same with "r.e. indices" in place of canonical indices: A is *hyperhypersimple* iff A is r.e., coinfinite and there is no recursive function such that $W_{f(i)} \cap W_{f(j)} = \emptyset$ and $W_{f(i)} \not\subseteq A$ for all distinct i, j. In the following, further interesting properties of sets will be investigated. Recall that $A \subseteq^* B$ means that all but finitely many elements of A are in B.

Definition 8.1. Let A be an r.e. coinfinite set. A is called *maximal* iff for all r.e. sets B, either $B - A$ or $\mathbb{N} - B - A$ is finite; A is called *r-maximal* iff for all recursive sets B, either $B - A$ or $\mathbb{N} - B - A$ is finite. A is called *dense simple* iff for every recursive function f there is a constant c such that for all n the set $\{0, 1, 2, \ldots, f(n)\} - A$ has at most $n + c$ elements.

An infinite set A is called *regressive* iff there is a partial-recursive function f and a (not necessarily recursive) enumeration $a_0, a_1, a_2, \ldots$ such that $f(a_{n+1}) \downarrow = a_n$; A is called *retraceable* iff there is an enumeration and f as before with the additional property that $a_n < a_{n+1}$ for all n.

Jockusch [45] called a set A is called *semirecursive* iff there is a recursive function f with two inputs such that $f(x, y) \in A \cap \{x, y\}$ whenever $x \in A \vee y \in A$. Alternatively, one can say that A is semirecursive iff there is a recursive function g such that, for all k, either $g(k) \in D_{g(k)} \cap A$ or $D_{g(k)}$ is disjoint to A.

A set A is called (m, n)-*recursive* if there is a recursive function $f : \mathbb{N}^n \to \{0, 1\}^n$ such that for all inputs of n different natural numbers $x_1, x_2, \ldots, x_n$ the output $(y_1, y_2, \ldots, y_n) = f(x_1, x_2, \ldots, x_n)$ satisfies for at least m coordinates k that $A(x_k) = y_k$.

Trakhtenbrot [113] called a set A *autoreducible* iff there is an index e such that $\varphi_e^{A \cup \{x\}}(x) = A(x)$ for all x; that is, φ_e can compute $A(x)$ using the oracle A without querying A at x.

Exercise 8.2. Let A be an infinite r.e. set. Show that A is regressive. Show that A is retraceable iff A is recursive. Show that every infinite retraceable set B is autoreducible.

Exercise 8.3. Show that every r.e. set with infinite retraceable complement is semirecursive. Show that for every semirecursive set A there is a recursive linear ordering

$\sqsubset$ such that A is an initial segment of $\sqsubset$, that is, $\forall x \in A \forall y \sqsubset x\, [y \in A]$.

Exercise 8.4. Consider a set $A = \{a_0, a_1, a_2, \ldots\}$ with $a_n < a_{n+1}$ for all n. Show that if A is retraceable via a total function f then A is $(1,2)$-recursive. But show that one can choose A also such that A is only retraceable via a partial-recursive function and is not $(1,n)$-recursive for every n.

Theorem 8.5 [111]. *If $2m > n$ and A is (m,n)-recursive then A is recursive.*

Proof. Assume that f witnesses that A is (m,n)-recursive. Let T be the tree of all σ such that for all distinct $x_1, x_2, \ldots, x_n \in \mathrm{dom}(\sigma)$ it holds that m of the n bits in $f(x_1, x_2, \ldots, x_n)$ and $(\sigma(x_1), \sigma(x_2), \ldots, \sigma(x_n))$ coincide. Then A is an infinite branch of T.

Let B be a further infinite branch of T. Then for every n numbers $x_1, x_2, \ldots, x_n$ the function $f(x_1, x_2, \ldots, x_n)$ differs from A on at most $n-m$ places and from B on at most $n-m$ places; as there are n inputs and $2(n-m) < n$ there is an index k such that $A(x_k)$ and $B(x_k)$ coincide both with the k-th bit of $f(x_1, x_2, \ldots, x_n)$. It follows that there are at most $n-1$ numbers on which A and B differ. So all infinite branches of T are finite variants and T has at most countably many infinite branches.

As T is a recursive tree and as T has at most countably many infinite branches there is an infinite branch B which is an isolated infinite branch of T. Hence there is a node σ such that B is the only infinite branch of T which contains σ. Note that

$$B(x) = a \Leftrightarrow \exists t > x + |\sigma|\, \forall \tau \in \{0,1\}^t\, [\tau \in T \wedge \tau \text{ extends } \sigma \Rightarrow \tau(x) = a]$$

and that this definition provides a method to compute B and that hence B is recursive. As A is a finite variant of B, A is recursive as well. ∎

Exercise 8.6. Let B be a nonrecursive set and let $a \mapsto \eta_a$ be the bijection from natural numbers into strings which was defined in Remark 5.1. Now define

$$A = \{a : \exists x\, [\eta_a = B(0)B(1)B(2)\ldots B(x)]\}$$

and show that A is (m,n)-recursive iff $2m \leq n$. This shows that Theorem 8.5 cannot be improved.

Dëgtev [25] showed that there is a proper hierarchy of $(1,n)$-recursive sets and that the differences in the hierarchy can be witnessed by r.e. sets.

Theorem 8.7 [25]. *For every n there is an r.e. set A such that A is $(1,n+1)$-recursive but not $(1,n)$-recursive.*

Proof. This is another finite injury priority construction which diagonalizes against the e-th function to $(1,n)$-compute A on the interval I_e which might be moved during the construction. The goal of the construction is to satisfy the following requirement:

R_e: The function φ_e fails to compute on I_e a vector of n bits such that at least one component coincides with A on these n inputs.

P: The resulting set A is $(1,n+1)$-recursive.

The construction is the following one:

- One uses intervals I_e of length n which can be moved, the algorithm to move them is the initialization $I_{e,0} = \{ne, ne+1, ne+2, \ldots, ne+n-1\}$ and which might be moved upwards. The value of A_0 is $\emptyset$.
- One searches at stage $s+1$ for the least e for which R_e needs attention and is not satisfied. This is the least e for which there is are $a, b_0, b_1, b_2, \ldots, b_{n-1}$ such that $I_{e,s} = \{na+k : k < n\} \subseteq \{0,1,2,\ldots,sn-1\}$, $\varphi_{e,s}(na, na+1, na+2, \ldots, na+n-1) = (b_0, b_1, b_2, \ldots, b_{n-1})$ and either $(b_0, b_1, b_2, \ldots, b_{n-1}) \neq (1,1,1,\ldots,1)$ or $I_{e,s} \cap A_s = \emptyset$.
- If e is found then let $I_{d,s+1} = \{(s+d)n+k : k < n\}$ for all $d > e$, let $I_{d,s+1} = I_{d,s}$ for all $d \leq e$ and let

$$A_{s+1} = A_s \cup \{na+k : k < n \wedge b_k = 0\} \cup \{x < sn : \forall d\, [x \notin I_{d,s+1}]\}$$

  else let $I_{d,s+1} = I_{d,s}$ for all d and let

$$A_{s+1} = A_s \cup \{x < sn : \forall d\, [x \notin I_{d,s+1}]\}.$$

The set A is recursively enumerable: This follows from the fact that the algorithm to compute A_{s+1} from A_s is recursive, that $A_0 = \emptyset$ and that $A_s \subseteq A_{s+1}$ for all s.

The set A is not $(1,n)$-recursive: To see this, note that whenever $I_{d,s+1} \neq I_{d,s}$ then $\min(I_{d,s+1}) \geq sn$. Hence it does not happen that elements are enumerated into $I_{e,s+1} \cap A$ and later into $I_{d,t+1}$ such that $I_{d,t+1} = I_{e,s+1}$ and $s < t$. Thus, for the final value I_e of the interval $I_{e,s}$ it holds that either φ_e does not output an n-bit vector on the inputs of this interval or φ_e is ultimately diagonalized by enumerating the corresponding elements into A. Thus each R_e is satisfied and A is not $(1,n)$-recursive.

The set A is $(1,n+1)$-recursive: To see this, consider inputs $x_0, x_1, x_2, \ldots, x_n$ with maximum x_k and $s = x_k + 2$. Now let

$$f(x_0, x_1, x_2, \ldots, x_n) = \begin{cases} 1^{n+1} & \text{if } A_s \cap \{x_0, x_1, x_2, \ldots, x_n\} \neq \emptyset; \\ 0^k 10^{n-k} & \text{otherwise.} \end{cases}$$

In the case that A_s contains already some of the inputs the answer 1^{n+1} is certainly correct on one component. Otherwise either one of the zeroes is correct or there is a stage $t+1$ where the $\min\{x_0, x_1, x_2, \ldots, x_n\}$ is enumerated into A_{t+1}. This happens either because the interval containing this minimum at stage t is moved away or the interval containing this minimum at stage t gets some elements enumerated at stage $t+1$. As $x_k \geq \min\{x_0, x_1, x_2, \ldots, x_n\} + n$, it happens in both cases that x_k does not belong to any interval in stage $t+1$ and that $x_k \leq tn$; hence $x_k \in A_{t+1}$ as well and the value $0^k 10^{n-k}$ coincides with $(A(x_0), A(x_1), A(x_2), \ldots, A(x_n))$ in at least one component. ▮

Exercise 8.8. Show by induction over n that every $(1,n)$-recursive set is autoreducible. This is obviously true for $n = 1$. Assuming that it is true for n, consider any set A which is $(1, n+1)$-recursive but not $(1,n)$-recursive. Taking a function f witnessing that A is $(1, n+1)$-recursive, show that for every x there are $y_1, y_2, \ldots, y_n$ such that $f(x, y_1, y_2, \ldots, y_n) = (a, b_1, b_2, \ldots, b_n)$ with $b_k \neq A(y_k)$ for $k = 1, 2, \ldots, n$. Conclude that then $A(x) = a$ and use this property to construct an autoreduction.

Remark 8.9. Every $(1,2)$-recursive r.e. set is semirecursive but there is a $(2,4)$-recursive r.e. set which is not semirecursive. Every $(m+1, n+1)$-recursive set is (m,n)-recursive but the converse does not hold as one can construct for every k an r.e. set A_k such that A_k is (m,n)-recursive iff $2m + k \leq n$. A well-studied problem is the set $\{(m,n,h,k) :$ every (m,n)-recursive set is (h,k)-recursive$\}$ and its variant $\{(m,n,h,k) :$ every (m,n)-recursive r.e. set is (h,k)-recursive$\}$. McNicholl [70] showed that these sets are recursive.

Friedberg [35] constructed a maximal set. Maximal sets are the standard example of hyperhypersimple sets. If one orders the r.e. sets by $A \subseteq^* B$ which says that all but finitely many elements of A are in B then the maximal sets do not have any set properly above them besides the cofinite ones. So they form a maximal point in the lattice of coinfinite r.e. sets ordered by $\subseteq^*$ and this explains their name.

Theorem 8.10 [35]**.** *There is a maximal set.*

Proof. The basic idea is to assign to every position y an evaluation $eval_e(y)$ (called the "e-state of y") defined as

$$\mathrm{eval}_e(y) = \sum_{d \in \{0,1,\ldots,e\}} 2^{-d} W_d(y)$$

with the corresponding approximation

$$\mathrm{eval}_{e,s}(y) = \sum_{d \in \{0,1,\ldots,e\}} 2^{-d} W_{d,s}(y)$$

at stage s. The idea is now to build a set A with complement $\{x_0, x_1, x_2, \ldots\}$ given in ascending order such that

$$\forall e, i, j\,[e \leq i \leq j \Rightarrow \mathrm{eval}_e(x_i) \geq \mathrm{eval}_e(x_j)].$$

This condition is satisfied as follows: At stage 0 let $A_0 = \emptyset$ and $x_{e,0} = e$ for all e. At stage $s+1$ the updates have the goal that on the one hand for every e there are only finitely many s with $x_{e,s+1} \neq x_{e,s}$ and on the other hand the goal that eval_e is nonincreasing on $x_e, x_{e+1}, x_{e+2}, \ldots$ is met in the limit.

- For $e = 0, 1, \ldots, s$ check whether there is d_e such that
  - $e < d_e$;
  - $x_{e,s} < x_{d_e,s} \leq s$;
  - $\mathrm{eval}_{e,s}(x_{e,s}) < \mathrm{eval}_{e,s}(x_{d_e,s})$.
- If d_e exists for some e then take the least e for which d_e exists and update the markers accordingly:
  - $x_{c,s+1} = x_{c,s}$ for all $c < e$;
  - $x_{e,s+1} = x_{d_e,s}$;
  - $x_{c,s+1} = s + c$ for all $c > e$.
- If d_e does not exist for any e then let $x_{c,s+1} = x_{c,s}$ for all s.
- In both cases let $A_{s+1} = \{0, 1, 2, \ldots, s\} - \{x_{0,s+1}, x_{1,s+1}, \ldots, x_{s,s+1}\}$.

Now the various properties of the construction are verified.

The set A is recursively enumerable: One can easily see that whenever $s > y$ and $y = x_{e,s}$ but $y \notin \{x_{c,s+1} : c \in \mathbb{N}\}$ then $y \neq x_{d,t}$ for all d and all $t > s$. Furthermore, $x_{e,s} \geq e$ for all e. It follows that

$$A = \{y : \exists s > y\,[y \notin \{x_{0,s}, x_{1,s}, \ldots, x_{y,s}\}]\}$$

and so A is a Σ^0_1-set; that is, recursively enumerable.

For each e there are only finitely many s with $x_{e,s} \neq x_{e,s+1}$: This can be proven by induction. So given e, let s be so large that $x_{d,u} = x_{d,s}$ for all $u > s$ and $d < e$. So, if $u > s$ and $x_{e,u+1} \neq x_{e,u}$ the change is not imposed by any $d < e$ and so $\mathrm{eval}_{e,s}$ goes up: $\mathrm{eval}_{e,u+1}(x_{e,u+1}) > \mathrm{eval}_{e,u}(x_{e,u})$. The rate of going up is at least 2^{-e} by the way how eval_e is defined; as the minimal value is 0 and the maximal value below $1 + \frac{1}{2} + \frac{1}{4} + \ldots = 2$, this going up can only happen finitely often and there is a stage

$t > s$ with $\text{eval}_{e,t}(x_{e,t}) = \text{eval}_{e,u}(x_{e,u})$ for all $u > t$. Hence $x_{e,t} = x_{e,u}$ for all $u > t$. This completes the inductive step. In particular there is for every e a limit x_e of all $x_{e,s}$.

If $e < d$ then $\text{eval}_e(x_e) \geq \text{eval}_e(x_d)$: If this would be false then there would e, d, s with $e < d$, $x_{e,s} = x_e < x_d = x_{d,s}$ and $\text{eval}_{e,s}(x_{e,s}) < \text{eval}_{e,s}(x_{d,s})$. As a consequence either $x_{e,s+1} = x_{d,s}$ or $x_{c,s+1} \neq x_{c,s}$ for some $c < e$ which imposes that $x_{e,s+1} \neq x_{e,s}$ as well. So the update rules for the $x_{e,s}$ contradict that it can happen in the limit that $e < d \wedge \text{eval}_e(x_e) < \text{eval}_e(x_d)$.

The condition $\forall e, i, j\,[e \leq i \leq j \Rightarrow \text{eval}_e(x_i) \geq \text{eval}_e(x_j)]$ is satisfied: Assume by way of contradiction that $\text{eval}_e(x_i) < \text{eval}_e(x_j)$. Now this implies that $\text{eval}_i(x_i) < \text{eval}_i(x_j)$ as the additional parts of the sum cannot compensate a difference which is at least 2^{-e}. This resulting statement contradicts the property found in the previous paragraph.

Either $W_e - A$ or $\mathbb{N} - W_e - A$ is finite, hence A maximal: Note that $\text{eval}_e(x_e) \geq \text{eval}_e(x_{e+1}) \geq \text{eval}_e(x_{e+2}) \geq \ldots$ and that all these are members of a finite set of rational numbers. Hence there is a d such that $\text{eval}_e(x_c) = \text{eval}_e(x_d)$ for all $c \geq d$. It follows that $W_e(x_c) = W_e(x_d)$ for all $c \geq d$. If $W_e(x_d) = 1$ then $\mathbb{N} - W_e - A$ is finite; if $W_e(x_d) = 0$ then $W_e - A$ is finite. Thus A is a maximal set. ∎

Exercise 8.11. Show that the maximal set constructed in Theorem 8.10 is $(1, 3)$-recursive.

Exercise 8.12. Show that every maximal set A is dense simple. Given any recursive and strictly increasing function f, let

$$B = \{x : \exists n\,[f(n) < x \wedge \{y : f(n) \leq y < x\} \subseteq A]\}$$

and show that $A \cup B$ is cofinite. Then conclude

$$\forall^\infty n\,[|\{x : f(n) \leq x < f(n+1) \wedge x \notin A\}| \leq 1]$$

and use this property to show that A is dense simple.

Obviously every maximal set is r-maximal. So one might ask whether the converse is also true. Lachlan [63] and Robinson [86] constructed r-maximal sets which do not have a maximal superset. The following result follows essentially this construction, but is formulated slightly more general.

Theorem 8.13 [62, 86]**.** *There is an r-maximal set which has no dense simple superset.*

Proof. The construction goes in two stages: First a set B will be constructed such that no r.e. superset of B is dense simple. B will be defined in parallel on intervals I_n of length $2n \cdot 2^n$. Then, given a maximal set A, one considers the set

$$E = \{x : x \in B \vee \exists n\,[x \in I_n \wedge n \in A]\}$$

and shows that this set is r-maximal.

The set B is constructed in stages in parallel on each I_n. At stage 0, $I_n \cap B_0 = \emptyset$. At stage $s+1$, let

$$e_s = \min\{e : e = n \vee [I_n \not\subseteq W_{e,s} \cup B_s \wedge 2 \cdot |I_n - B_s - W_{e,s}| \leq |I_n - B_s|]\}$$

and define

$$B_{s+1} \cap I_n = \begin{cases} (B_s \cap I_n) \cup (I_n - W_{e,s}) & \text{if } e_s < n; \\ B_s \cap I_n & \text{if } e_s = n. \end{cases}$$

Now it is verified that B and E have the desired properties.

B and E are recursively enumerable: In the case of B the algorithm how to enumerate elements into B on an interval I_n is clearly listed; the set E is constructed from A and B using existential quantification and union, hence E is recursively enumerable as well.

B and E are co-infinite r.e. sets: As A is coinfinite, it is sufficient to consider the infinitely many $n \notin A$. For each $n \notin A$, there are m stages s where $B_{s+1} \cap I_n$ is a proper superset of $B_s \cap I_n$; in each of these stages, $|I_n - B_{s+1}| > 0.5 \cdot |I_n - B_s|$ and hence there are at least $2n \cdot 2^{n-m}$ elements in $I_n - B$; as for each $e < n$ there is at most one such stage, one knows that $m \leq n$ and $|I_n - B| \geq 2n$.

B and E have no dense simple superset: Assume that W_e is a dense simple set and let $f(n) = \max(I_n)$ for all n. Then, for almost all n, $|\{0, 1, 2, \ldots, f(n)\} - W_e| \leq 2n$. Hence, for almost all n there is a stage s such that $|I_n - W_{e,s}| \leq 2n$ and so there is by the construction of B a stage s with $B_{s+1} \supseteq I_n - W_{e,s}$. Hence all sufficiently large elements in the complement of W_e are also in B and W_e; as W_e is coinfinite, B contains elements outside W_e and W_e is not a superset of B. So B has no dense simple superset. The same is true for the coinfinite superset E of B.

If $W_i \cup W_j = \mathbb{N}$ and $n > i + j$ then either $I_n - W_i \subseteq B$ or $I_n - W_j \subseteq B$: Let s be so large that $B_s \cap I_n = B \cap I_n$ and $W_i \cap I_n = W_{i,s} \cap I_n$ and $W_j \cap I_n = W_{j,s} \cap I_n$. Note that $I_n \subseteq W_{i,s} \cup W_{j,s}$, hence there is $e \in \{i, j\}$ such that at least half of the elements in $I_n - B_s$ are in $W_{e,s}$; that is, $|I_n - B_s - W_{e,s}| \leq 0.5 \cdot |I_n - B_s|$. As $e < n$ it follows that either $I_n - W_{e,s} \subseteq B_s$ or $B_{s+1} \cap I_n \supset B_s \cap I_n$. But the latter does not occur according to the choice of s, so the former holds and $I_n - W_e \subseteq B$ as well.

E is r-maximal: Let W_i, W_j be a recursive partition of $\mathbb{N}$ and consider the sets $W_{i'} = \{n : I_n \subseteq E \cup W_i\}$, $W_{j'} = \{n : I_n \subseteq E \cup W_j\}$. By the previous property,

every $n > i + j$ is either in $W_{i'}$ or $W_{j'}$. Furthermore, $A \subseteq W_{i'}$ and $A \subseteq W_{j'}$. But if $I_n \not\subseteq E$ then $n \notin W_{i'} \cap W_{j'}$; hence one of these sets has to be coinfinite and a finite variant of A while the other one is cofinite. It follows that almost all elements of the complement of E are either in W_i or in W_j; thus E is r-maximal. This completes the proof. ∎

Exercise 8.14. Show that the following sets are always hypersimple: maximal sets, r-maximal sets, hyperhypersimple sets and dense simple sets.

Exercise 8.15. Let a simple and semirecursive set A be given. Show that A is hypersimple. Show that $B = \{y : \exists x \in A\,[x^2 \leq y < (x+1)^2]\}$ is also simple and semirecursive but not dense simple.

Exercise 8.16. Recall from Exercise 8.3 that every semirecursive set is an initial segment of a recursive partial ordering $\sqsubset$. Use this characterization to show that every semirecursive simple set A has a dense simple superset.

Here an outline of a possible proof for the second fact: Call an equivalence relation $\sim$ is called positive iff the set $\{\langle x, y\rangle : x \sim y\}$ is recursively enumerable. Given a maximal set B, show that the equivalence relation $\sim$ given by

$$x \sim y \Leftrightarrow \forall z\,[\min\{x, y\} \leq z < \max\{x, y\} \Rightarrow z \in B]$$

is a positive equivalence relation where each equivalence class is finite. Consider $E = \{x : \exists y \sim x\,[x \sqsubset y]\} \cup A$ and show that E is a dense simple superset of A.

Proposition 8.17. *The complement of an r-maximal set is not regressive.*

Proof. Assume that A is r.e. and has a regressive complement. Then $\mathbb{N} - A$ can be (nonrecursively) enumerated as $a_0, a_1, a_2, \ldots$ such that there is a partial-recursive function f with $f(a_{n+1}) = a_n$ for all n. Let the partial-recursive function g be the least fixed-point of the operator which maps every partial recursive function θ to the function θ' such that $\theta'(x) = 0$ if $x = a_0$ and $\theta'(x) = \theta(f(x)) + 1$ if $x \neq a_0$ and both, $f(x)$ and $\theta(f(x))$, are defined. It is easy to see that $g(a_n) = n$ for all n; g might also be defined on some members of A. Now one can define a recursive set B by the following case-distinction where, in the case that both cases apply, that case is taken which is proven to hold first:

$$B(x) = \begin{cases} 1 & \text{if } x \in A \text{ or } g(x) \text{ is defined and odd;} \\ 0 & \text{if } g(x) \text{ is defined and even.} \end{cases}$$

Clearly $B(x)$ is defined for all members of A. Furthermore, every $x \notin A$ is equal to some a_n and then $B(x) = 1$ if n is odd and $B(x) = 0$ if n is even. So B is a recursive set such that $\mathbb{N} - A$ has an infinite intersection with both, B and $\mathbb{N} - B$. Hence A is not r-maximal. In other words, no r-maximal set has a regressive complement. ∎

9 Permitting and Infinite Injury Priority Methods

The topic of this section are more advanced techniques to construct recursively enumerable sets with certain properties. The first method to be presented is permitting and the proof given is that for every non-recursive r.e. set bounds a simple but not hypersimple set. Note that permitting gives normally not only a Turing reduction but even a wtt-reduction.

Theorem 9.1. *For every nonrecursive r.e. set A there is a simple but not hypersimple set B with $B \leq_{wtt} A$.*

Proof. Let $I_0, I_1, I_2, \ldots$ be a recursive partition of the natural numbers into intervals such that $|I_n| \geq n+1$ for every number n. Recall the definition of the convergence module c_A of A:

$$c_A(n) = \min\{s \geq n : A_s(0)A_s(1)A_s(2)\ldots A_s(n) = A(0)A(1)A(2)\ldots A(n)\}.$$

Now define a partial-recursive function ψ as follows:

$$\psi(\langle n,e\rangle) = \begin{cases} \min(W_{e,t} \cap I_n) & \text{if } t \text{ is the first number with} \\ & W_{e,t} \cap I_n \neq \emptyset \text{ and } t \leq c_A(n); \\ \uparrow & \text{if there is no such } t. \end{cases}$$

Furthermore, let $B = \{\psi(\langle n,e\rangle) : n \in \mathbb{N} \wedge e < n \wedge \psi(\langle n,e\rangle)\downarrow\}$. Clearly the set B is r.e.; furthermore, B is coinfinite as for every n the intersection $B \cap I_n$ equals $\{\psi(\langle n,e\rangle) : e < n \wedge \psi(\langle n,e\rangle)\downarrow\}$ and has at most n elements although $|I_n| > n$. Furthermore, as $n \leq \min(I_n)$ for all n, B has a nonelement between n and $\max(I_n)$ and is not hypersimple.

Now assume by way of contradiction that there is an infinite set W_e with $W_e \cap B = \emptyset$. There is a recursive function g such that for every n, $\max(W_{e,g(n)}) \geq \min(I_n)$. Now $g(n) > c_A(n)$ for all $n > e$ as otherwise there is an $n > e$ and an $m \geq n$ with $I_m \cap W_{e,g(n)} \neq \emptyset$, $c_A(m) \geq c_A(n) \geq g(n)$ and $\psi(\langle m,e\rangle)\downarrow \in A \cap W_e$. But the condition $c_A(n) < g(n)$ implies that $n \in A \Leftrightarrow n \in A_{g(n)}$ for all $n > e$ giving that A is recursive in contradiction to the assumption. Hence B intersects every infinite r.e. set and B is simple. ∎

The next two results are parallel. If $A \leq_T \mathbb{K}$ then $A' \geq_T \mathbb{K}$ and A' is r.e. relative to $\mathbb{K}$. The next two results show that this result can be inverted: First, if $B \geq_T \mathbb{K}$ and B is r.e. relative to $\mathbb{K}$ then there is an $A \leq_T \mathbb{K}$ with $A' \equiv_T B$. Second, this A can be taken to have r.e. Turing degree. The first result will be proven by a direct construction while the second result needs an infinite injury priority construction. For the first result, one needs a definition.

Definition 9.2. Call a function F *growth-generic* iff for every r.e. set $W \subseteq \mathbb{N}^*$ of strings either there is an n with $F(0)F(1)\ldots F(n) \in W$ or it holds for almost all m that $F(0)F(1)\ldots F(m) \cdot \{F(m), F(m)+1, F(m)+2, \ldots\}^*$ is disjoint from W.

If one calls a number m with $\forall k \geq m\,[F(m) \leq F(k)]$ a "true stage" then one can also reformulate the definition as follows: F is growth-generic iff for every r.e. set $W \subseteq \mathbb{N}^*$ of strings either there is an n with $F(0)F(1)\ldots F(n) \in W$ or there is a true stage m with $F(0)F(1)\ldots F(m) \cdot \{F(m), F(m)+1, F(m)+2, \ldots\}^*$ being disjoint from W. In the following, one uses also the function F as an oracle which return $F(m)$ when queried at m. The definitions of an F-recursive function and the jump F' of F are parallel to the corresponding definitions for sets.

Theorem 9.3: Shoenfield's Jump Inversion [97]. *Let $B \geq_T \mathbb{K}$ and B be r.e. relative to $\mathbb{K}$. Then there is a growth-generic function $F \leq_T \mathbb{K}$ with $F' \equiv_T B$.*

Proof. Let G be a $\mathbb{K}$-recursive one-one enumeration of B. Now the goal is to make a function F such that

- F takes every value only finitely often;
- For all n there is $m \geq n$ with $F(m) = G(n)$;
- $F' \leq_T B$.

Then B can be approximated in the limit relative to F as follows: Let G_s be an approximation to G at stage s and let $x \in B_s$ iff there are n, m with $n \leq m \leq s$, $G_s(n) = x$ and $F(m) = x \vee m = 0$. Clearly the B_s are uniformly recursive relative to F. Furthermore, if s is sufficiently large, then the parameter m in this condition is the maximal m where $F(m) = x$ and $G_s(n) = G(n)$ for all $n \leq m$; hence $B_s(x) = 1$ iff there is an $n \leq m$ with $G_s(n) = x$ iff there is an $n \leq m$ with $G(n) = x$ iff $x \in B$. Thus $B \leq_T F'$. Thus this fourth condition is not postulated explicitly as it follows from the other three conditions on F.

Now, for the construction of F, one chooses a finite-extension method. That is, one constructs a sequence of strings $\sigma_{n,m}$ such that $\sigma_{n,m+1}$ is always an extension of $\sigma_{n,m}$ and $\sigma_{n+1,0}$ is an extension of $\sigma_{n,n}$. This construction uses G and is thus $\mathbb{K}$-recursive. Let $W_0, W_1, W_2, \ldots$ be an enumeration of all r.e. sets of strings. Now define the strings used in the construction inductively according to the first case which applies.

- $\sigma_{0,0} = G(0)$;
- $\sigma_{n+1,0} = \sigma_{n,n}G(n+1)$;

- $\sigma_{n+1,m+1}$ with $m < n$ is τ for the first string τ found in

$$W_m \cap \sigma_{n+1,m} \cdot \{G(n+1), G(n+1)+1, G(n+1)+2, \ldots\}^*$$

  and $\sigma_{n+1,m+1} = \sigma_{n+1,m}$ if such a τ does not exist.

Let F be the limit of all $\sigma_{n,0}$. It is easy to see that the string $\sigma_{n,0}$ has at least length $n+1$ and ends with $G(n)$, hence the second condition holds.

Note that the enumeration G takes every value at most once. Given now any x, there is a k such that $G(n) > x$ for all $n \geq k$. It follows for all $n \geq k$ that the extensions $\sigma_{n+1,0}$ of $\sigma_{n,n}$ and $\sigma_{n+1,m+1}$ of $\sigma_{n+1,m}$ only append values to the given construction which are larger than x, hence $F(y) \neq x$ for all $y > |\sigma_{k,k}|$. So the first condition holds.

The third condition follows from the construction. The main ingredient is that can determine the true stages of F using B: One can find using B all the n such that $G(n) < G(k)$ for all $k > n$; these are just the n such that when enumerating $G(n)$ into B then all the values of B below $G(n)$ are already there. Now, for each set W_m and each true stage $n > m$ of G, one has that the extension $\sigma_{n,m+1}$ of $\sigma_{n,m}$ either is an element of

$$W_m \cap \sigma_{n,m} \cdot \{G(n), G(n)+1, G(n)+2, \ldots\}^*$$

or this intersection is empty. As F does not take any value below $G(n)$ beyong $|\sigma_{n,m}|$, it follows that either some prefix of F is in W_m or

$$W_m \cap F(0)F(1)\ldots F(k) \cdot \{F(k), F(k)+1, F(k)+2, \ldots\}^*$$

is empty for all $k > |\sigma_{n,m}|$. As $F \leq_T B$, one can compute from B whether there is an k such that $F(0)F(1)\ldots F(k) \in W_m$. Furthermore, for each e one can compute an index m with

$$W_m = \{\sigma : \varphi^{\sigma}_{e,|\sigma|}(e)\downarrow\}$$

and then check relative to B whether W_m contains a prefix of F. In other words, one can decide relative to B whether $e \in F'$. Hence $F' \leq_T B$. Note that this verification of the third condition included the proof that F is growth-generic. ∎

The next theorem is due to Sacks. Sacks' Jump Inversion Theorem states that the jumps of sets below $\mathbb{K}$ are already the jumps of r.e. sets. The main modification to the proof above is to make the function F to be approximable from below, that is, to obtain that $\{(x,y) : y \leq F(x)\}$ is an r.e. set. F is of course Turing equivalent to this set. For the result, the following notion of approximations to a $\mathbb{K}$-recursive enumeration is essential.

Exercise 9.4. Let G be a $\mathbb{K}$-recursive one-one numbering of a set. Then one can define an approximation $\gamma_0, \gamma_1, \gamma_2, \ldots$ by strings such that

- γ_0 is the empty string;
- either $\gamma_{s+1} = \gamma_s a$ for some a or γ_{s+1} is a prefix of γ_s;
- for all n there is an s such that $\gamma_t(n)\downarrow = G(n)$ for all $t \geq s$.

A stage s is called a true stage for the approximation $\gamma_0, \gamma_1, \gamma_2, \ldots$ iff $\gamma_t(n)\downarrow = G(n)$ for all $t \geq s$ and all $n \in \mathrm{dom}(\gamma_s)$.

Given a traditional approximation G_s to G, construct the above approximation $\gamma_0, \gamma_1, \gamma_2, \ldots$ from G and show that the three properties are satisfied. Show furthermore that the three conditions enforce that there are infinitely many true stages.

Theorem 9.5: Sacks' Jump Inversion [88]. *Let $B \geq_T \mathbb{K}$ and B be r.e. relative to $\mathbb{K}$. Then there is a function F which is approximable from below such that $F' \equiv_T B$.*

Proof. The idea is to build F meeting the following requirements. Again the construction uses a K-recursive one-one numbering G of B.

- P_e (preserve e-th computation): if $\varphi_e^{F_s}(e)$ converges and satisfies some additional condition specified later then $F(m) = F_s(m)$ for all m queried in the computation.
- C_e (coding of G): if $G(m) = e$ then there is $n \geq m$ with $F(n) \leq e$.
- M_e (mimimal value e): for almost all n, $F(n) > e$.

Among these requirements, M_e is the one which is violated infinitely often and therefore needs to act infinitely often. A further problem in the construction is that G is not available as in the previous theorem. Instead one has to use an approximation to G. The most suitable way is to approximate G by strings γ_s. According to Exercise 9.4 one can choose an approximation which satisfies the constraint that γ_{s+1} is either $\gamma_s a$ for some a or is a prefix of γ_s and γ_0 is the empty string. Then s will be called a true stage for this approximation iff it holds for all $n \in \mathrm{dom}(\gamma_s)$ and all $t \geq s$ that $\gamma_t(n)\downarrow = G(n)$. As shown in the exercise, the approximation $\gamma_0, \gamma_1, \gamma_2, \ldots$ has infinitely many true stages in which longer and longer pieces of G get defined permanently. The requirements P_e and C_e can set restraints which have to be observed by M_e. The restraints are adjusted in every step.

Let $F_s(n)$ be the value of F at n before step s and $F_{s+1}(n)$ be the value of F at n after step s. $F_0(n) = 0$ for all n. Now in step s of the construction, the following activities are carried out.

- Clear all old restraints from previous steps.

- For each $n \in \mathrm{dom}(\gamma_s)$, put on the least $m \geq n$ with $F_s(m) \leq \gamma(n)$ the restraint $C_{\gamma_s(n)}$.
- For $e = 0, 1, 2, \ldots, s$, if $\varphi_{e,s}^{F_s}(e)$ converges and all queries for a value $F_s(y)$ at some y satisfy that either $F_s(y) = y$ or $F_s(y) \geq e$ or y carries the restraint $C_{F_s(y)}$ or y carries a restraint P_d with $d < e$ then put the restraint P_e on all m where $F_s(m)$ had been queried during this computation.
- For each $m \leq s$, let $F_{s+1}(m)$ be the maximum of $\{F_s(m)\} \cup \{n+1 : n < m$ and m does not carry any of the restraints $C_0, C_1, C_2, \ldots, C_n$ and $P_0, P_1, \ldots, P_n\}$. For each $m > s$, $F_{s+1}(m) = 0$.

Note that the update of F_s to F_{s+1} makes sure that $F_s(m) \leq m$ for all m, hence F is on every value defined and takes never the value ∞.

Consider now any true stage s. If $\gamma_s(n)$ is defined, then by the definition of the true stage, $\gamma_t(n) = G(n)$ for all $t \geq s$. Then there is a least $m \geq n$ with $F_s(m) \leq G(n)$. One can now see that for all $t \geq s$ the restraint $P_{\gamma_s(n)}$ will be placed on m and hence $F_t(m) \leq G(n)$ for all $t \geq s$. It follows that $F(m) \leq G(n)$. Hence the coding condition C_n is met eventually. A restraint P_e does only protect $F_s(m)$ if $F_s(m) \geq e$; otherwise the requirements M_d with $d < e$ have higher priority than P_e. But one can show by induction over $e = 0, 1, 2, \ldots, s$ that whenever a restraint P_e is set, then it protects a computation which queries F at values $F_s(m)$ if either $F_s(m) \geq e$ or if $F_s(m)$ has to be kept at the current value due to another restraint of higher priority or due to $F_s(m) = m$. Therefore, one can show by induction for $t = s, s+1, s+2, \ldots$ that the restraints P_e will be placed at each of these stages and the values will be preserved when making F_{t+1} from F_t.

Now it is verified that each requirement M_e is met in the limit. Let s be any true stage which is so large that every restraint P_d with $d \leq e$ which will be placed at some true stage (and then remain forever) is already placed at s and that $m \in \mathrm{dom}(\gamma_s)$ whenever $G(m) \leq e$. Recall that for each $m \in \mathrm{dom}(\gamma_s)$ it holds that $\gamma_s(m) = G(m)$ and that the restraint $C_{G(m)}$ is placed on s or below as $F_s(s) = 0$. If d is in the range of G then the restraint C_d will be set by stage s on some value up to s and never change again; if d is not in the range of G then the restraint C_d will not be set at any true stage, although it might temporarily be set at other stages. Hence, for all true stages $t > s$, no number m with $s < m < t$ will carry any of the restraints $C_0, C_1, C_2, \ldots, C_e$, $P_0, P_1, P_2, \ldots, P_e$ and therefore $F_{t+1}(m) > e$. Hence $F(m) > e$ for all $m > s$ and the requirement M_e will be met in the limit.

Next one has to show that $F' \leq_T B$. This is shown by giving an inductive B-recursive decision procedure for F'. So assume that $F'(d)$ had already been determined for all $d < e$ and that s is a true stage in the construction so large that $G(n) \leq e \Rightarrow \gamma_s(n)\!\downarrow = G(n)$ for the finitely many n involved and that all restraints P_d which are set

to protect the converging computation $\varphi_d^F(d)$ are already set and that $F_s(m) = F(m)$ for all the m queried. The restraints C_d with $d \leq e$ are, if they will ever be set permanently, already set at the true stage s.

Now $e \in F'$ iff there is a stage $t \geq s$ such that the computation $\varphi_{e,t}^{F_t}(e)$ is protected by P_e and the m carrying restraints P_d with $d < e$ or C_d with $d \leq e$ are the same as the m carrying these restraints at stage s.

One uses B to compute s and $\mathbb{K}$ to check for the existence of t. As $\mathbb{K} \leq_T B$, it follows that the condition can be checked using the oracle B.

The reason for the condition on the P_d and C_d to be the same as at stage s is that one is searching for true stages t with the property that $\varphi_{e,t}^{F_t}(e)$ converges and is protected by P_e. But as one cannot search over all true stages relative to B, this approximation is used instead. It mainly enforces that no stages t with false protected computations qualify which will be erased later. Indeed, in the case that a computation halts and becomes protected, then all those $F_t(m)$ queried which are below e are protected by some other restraint of higher priority which already preexists and is verified by the algorithm; hence the computation will be preserved at all future stages and survive. So whenever the algorithm says $e \in F'$ then this is also true. For the other way round, assume that $\varphi_e^F(e)$ converges and let t be a true stage above s where the computation has already converged and all m queried by the computation satisfy $m \in \mathrm{dom}(\gamma_t)$ and $F_t(m) = F(m)$. Those m with $F_t(m) < \min\{m, e\}$ have to be protected by a restraint of higher priority as otherwise $F_{t+1}(m) > F_t(m)$; as t is a true stage these restraints must be some of the restraints C_d with $d \leq e$ or P_d with $d < e$ already placed at stage s. Hence the stage t would qualify in the search and the B-recursive algorithm also says that $e \in F'$. Hence the algorithm is correct and $F' \leq_T B$.

The property that $B \leq_T F'$ is very similar to the one proven in Theorem 9.3. An F-recursive approximation to B is defined as follows: $x \in B_s$ iff there are n, m with $n \leq m \leq s$, $\gamma_s(n)\!\downarrow = x$ and $F(m) \leq x \vee m = 0$. The main difference is the use of γ_s in place of G_s and of $F(m) \leq x$ instead of $F(m) = x$. One can verify that both changes still lead to a valid F-recursive approximation of B which proves that $B \leq_T F'$. ∎

The previous result shows that the r.e. sets can realize all jumps which can be obtained by a set below $\mathbb{K}$. But the result does not yet give that a set of high Turing degree (where the jump is $\mathbb{K}'$) is also Turing incomplete. This is obtained by the next result of Sacks [90] who constructed a maximal set with this property.

Theorem 9.6 [90]. *For every r.e. and nonrecursive set A there is a maximal set B such that $A \not\leq_T B$.*

Proof. Let A be given and let c_A be the convergence module of A. The goal is to build a set B with movable markers which on one hand want to optimize their e-state

and on the other hand are restrained by attempts to preserve computations giving the wrong value for the convergence module c_A. On the one hand, for each function φ_e^B restraints try to protect more and more of computations relative to B_s until the computation turns out to be different from c_A below the current value which should be restrained; on the other hand, the goal to make the e-state as large as possible tries to move the markers more and more out. The compromise between these two requirements will depend on the quantity the marker is moved. Recall the definition of the e-state and its approximation as

$$\mathrm{eval}_e(y) = \sum_{d=0,1,\ldots,e} W_d(y) \text{ and } \mathrm{eval}_{e,s}(y) = \sum_{d=0,1,\ldots,e} W_{d,s}(y).$$

At stage 0, $x_{e,0} = e$ for all e. Requirements X_e try to make the e-state $\mathrm{eval}(x_{e,s})$ as large as possible by moving x_e from $x_{e,s}$ to a larger value $x_{e,s+1}$, whenever needed and possible.

Instead of one requirement to deal with a potential equation $A = \varphi_a^B$ one has infinitely many requirements $R_{\langle a,b,c\rangle}$ attacking this reduction. Each of these requirements has an assumption on the *outcome of other ingredients* of the construction, namely $R_{\langle a,b,c\rangle}$ acts only infinitely often if for all $d \geq a+b$, $\mathrm{eval}_a(x_d) = c \cdot 2^{-a}$. The name "outcome" for this assumption stems for the fact that in many similar constructions the assumption deals with the behaviour of the strategies to satisfy higher priority requirements and an outcome is the behaviour of these strategies in the limit. The requirement $R_{\langle a,b,c\rangle}$ requires attention if a sufficiently large computation should be secured and the $x_{d,s+1}$ where $B_{s+1}(x_{d,s+1})$ is queried in this computation satisfy $\mathrm{eval}_{a,s+1}(x_{d,s+1}) = c \cdot 2^{-a}$ and the old part of the computation is consistent with $c_{A,s+1}$. The action of $R_{\langle a,b,c\rangle}$ is to increase its restraint to the number s of the current stage.

The initialization of the restraints is $r_{e,0} = 0$. Requirement R_e tries to prevent markers x_e from moving in order to preserve certain computations whenever it is needed. At stage $s+1$ the following updates are done.

- Requirement X_e tries to make e-states as large as possible and it needs attention if

$$\exists i \leq e\, \exists j > e\, \forall \langle a,b,c\rangle < e \quad [x_{j,s} < s \wedge \mathrm{eval}_{i,s}(x_{e,s}) < \mathrm{eval}_{i,s}(x_{j,s}) \wedge (r_{\langle a,b,c\rangle,s} < x_{e,s} \vee a \geq i)].$$

- Find the least e such that X_e needs attention with parameters i, j as above and update

$$x_{d,s+1} = \begin{cases} x_{d,s} & \text{if } d < e; \\ x_{j,s} & \text{if } d = e; \\ d+s+1 & \text{if } d > e. \end{cases}$$

- In the following let $B_{s+1} = \{0, 1, 2, \ldots, s\} - \{x_{e,s+1} : e \leq s\}$ and

$$\begin{aligned} c_{A,s+1}(x) \;\; = \;\; & \min\{t : t = s+1 \;\vee\; [t \geq x \;\wedge \\ & A_t(0)A_t(1)\ldots A_t(x) = A_{s+1}(0)A_{s+1}(1)\ldots A_{s+1}(x)]\}; \end{aligned}$$

  note that $c_{A,s+1}(x) = s+1$ for $s < x$.

- Requirement R_e needs attention iff $e < s$ and there are numbers a, b, c, x, u such that
  - $\langle a, b, c\rangle = e$, $r_{e,s} < x < u < s$ and $x_{a+b+u,s} < s$,
  - $\varphi_{a,u}^{B_{s+1}}(y)$ is defined for all $y \leq x$,
  - $\mathrm{eval}_{a,s}(x_{d,s+1}) = c \cdot 2^{-a}$ for all $d \in \{a+b, a+b+1, \ldots, a+b+u\}$,
  - there is no $y < x$ such that $\varphi_{a,r_{e,s}}^{B_{s+1}}(y)\!\downarrow < c_{A,s+1}(y)$.
- For each e, if R_e needs attention then update $r_{e,s+1} = s$ else keep $r_{e,s+1} = r_{e,s}$.

Now it is verified by induction that the construction does what it is required. The following facts are verified:

1. For every e the limit x_e of the $x_{e,s}$ exists and $B = \mathbb{N} - \{x_0, x_1, x_2, \ldots\}$ is recursively enumerable.
2. For all i and almost all e there is no $j > e$ such that $\mathrm{eval}_i(x_e) < \mathrm{eval}_i(x_j)$. Hence $\mathbb{N} - B \subseteq^* W_i$ or $\mathbb{N} - B \subseteq^* \mathbb{N} - W_i$.
3. For all e the limit $r_e = r_{e,s}$ exists and $r_e < \infty$.
4. For all a where φ_a^B is total there is an y with $\varphi_a^B(y) < c_A(y)$.

It is easy to see that these four conditions together give that $A \not\leq_T B$ and that B is maximal. Now these four properties are verified.

First, the $x_e = \lim_s x_{e,s}$ exist by the same inductive argument as in the construction of a maximal set: for every e there is an s such that $x_{d,s} = x_{d,t}$ for all $d < e$ and $t > s$. Hence any $t \geq s$ with $x_{e,s+1} \neq x_{e,s}$ implies that $\mathrm{eval}_{e,s}(x_{e,s}) + 2^{-e} \leq \mathrm{eval}_{e,s}(x_{e,s+1})$. One combines this with the facts that $\mathrm{eval}_{e,t}(y) \leq \mathrm{eval}_{e,t+1}(y)$ and that $0 \leq \mathrm{eval}_{e,t}(y) \leq 2$ for all y, t and obtains then that there are only finitely many $t > s$ with $x_{e,t+1} \neq x_{e,t}$. Thus $x_e = \lim_t x_{e,t}$ exists.

The further property that B is recursively enumerable follows from the fact that the sets $B_s = \{y < s : \forall e \leq s\, [y \neq x_{e,s}]\}$ are uniformly recursive and that whenever there is for some y a stage $s > y$ with $y \in B_s$ then at no future stage t there will be any marker $x_{e,t}$ with $x_{e,t} \neq y$ and $x_{e,t} = y$: the marker would either move to the

position of another marker and none of them sits on y at stage t or it would move to a place beyond t and $t \geq s$. Hence

$$B = \{y : \exists s > y \, \forall e \leq s \, [y \neq x_{e,s}]\}$$

and this set is given by an existential quantifier followed by a bounded quantifier, so it is recursively enumerable.

Second, assume by way of contradiction that i is the least number such that the second condition fails with parameter i and the third condition does not fail with any parameter $d < i$.

By assumption $r_d = \lim_s r_{d,s}$ exists and is in $\mathbb{N}$ for all $d < i$. Now consider any e such that $e \geq i$ and $e > r_d$ for all $d < i$; almost all e satisfy this property. Furthermore let $j > e$. If $\text{eval}_i(x_e) < \text{eval}_i(x_j)$ and s is sufficiently large then $x_{i,s} = x_i$, $x_{j,s} = x_j < s$ and $\text{eval}_{i,s}(x_{e,s}) < \text{eval}_{i,s}(x_{j,s})$. It would follow that requirement X_e needs attention from stage s onward and there would be $t > s$ with $x_{i,t} = x_{j,s}$ which is a contradiction to the assumptions. Hence, for almost all e, the values $\text{eval}_i(x_e)$ are the same and either almost all x_e are in W_i or almost all x_e are outside W_i. So it follows that $\mathbb{N} - B \subseteq^* W_i$ or $\mathbb{N} - B \subseteq^* \mathbb{N} - W_i$.

Third, assume by way of contradiction that e is the least number such that the third condition fails with parameter e and that the second condition does not fail with parameter $i \leq e$.

Let a, b, c be the components of e, that is, $e = \langle a, b, c \rangle$. Note that $a \leq e$. If there is $d \geq a + b$ with $\text{eval}_a(x_d) \neq c \cdot 2^{-a}$ then all sufficiently large stages s satisfy that either $r_{e,s} \leq x_d$ or $r_{e,s} > x_d \wedge x_{d,s} = x_d \wedge \text{eval}_{a,s}(x_{d,s}) = \text{eval}_a(x_d) \neq c \cdot 2^{-a}$; so either $r_{e,s} \leq x_d$ for almost all s or the second condition holds for all sufficiently large s and implies that R_e does not request attention anymore.

Thus assume the case that the outcome required by R_e on $\text{eval}_a(x_d)$ for $d \geq a + b$ is correctly guessed. Let s_0 be so large that $s_0 > x_{a+b+e}$, $x_{d,s_0} = x_d$ and $\text{eval}_{a,s_0}(x_d) = \text{eval}_a(x_d)$ for all $d \leq a + b + e$. Note that no requirement X_d with $d \leq e$ acts after stage s_0 by the choice of that stage. Now define inductively s_{x+1} to be the first stage such that there is an u_x for which the following three conditions hold:

- $s_{x+1} > u_x$, $s_{x+1} > s_x$ and $r_{e,s_{x+1}} > u_x$;
- $\text{eval}_{a,s_{x+1}}(x_{d,s_{x+1}}) = c \cdot 2^{-a}$ for all $d \geq a + b$ with $x_{d,s_{x+1}} \leq u_x$;
- $\varphi_{a,u_x}^{B_{s_{x+1}}}(y)\!\downarrow$ for all $y \leq x$.

If this inductive definition would go through for all x then the mapping $x \mapsto s_x$ is recursive and furthermore no requirement X_d with $d > a + b + e$ will move any $x_{d,s}$ with $x_{d,s} \leq u_x$ to a new value $x_{d,s+1} > x_{d,s}$ at a stage $s \geq s_x$ as this would

require $\text{eval}_{a,s}(x_{d,s}) > \text{eval}_{a,s}(x_{d,s}) = \text{eval}_a(x_{d,s})$ and result in some $x_{d'}$ with $d' \in \{a+b+e+1, a+b+e+2, \ldots, d\}$ having $\text{eval}_a(x_{d'}) > c{\cdot}a$ in contradiction the assumption. Hence $\varphi_{a,s_{x+1}}^{B_{s_{x+1}}}(x) = \varphi_a^B(x)$ for all x and φ_a^B would be recursive. Furthermore, it would follow from the fact that requirement R_e acts infinitely often that $\forall y\,[c_A(y) \leq \varphi_e^B(y)]$, a contradiction to A being not recursive.

This contradiction enforces that there is a maximal x where s_x is defined. As the second condition comes eventually true by the assumption on the outcome, it is the lack of the first or third condition that s_{x+1} cannot be defined. If there is an s with $r_{e,s+1} > r_{e,s} = r_{e,s_x}$ then the way the update is done implies that there is an u_x such that the third condition is satisfied and s_{x+1} could be taken as that $s+1$, hence $r_e = r_{e,s_x}$ in that case and the induction hypothesis is transferred from all $d < e$ to e.

Note that "Second" and "Third" above cover all possible cases. To see this let i be the minimum of all k such that either the second condition fails for k or $k = \infty$; let e be the minimum of all k such that either the third conditions fails for k or $k = \infty$. If $i < \infty \wedge i \leq e$ then the argumentation in "Second" shows that this case does not occur; if $e < \infty \wedge e < i$ then the argumentation in "Third" shows that this case does not occur. Hence $i = \infty \wedge i = \infty$ and both conditions are satisfied.

Fourth, consider now any a where φ_a^B is total. Furthermore let c be the unique number such that $\text{eval}_a(x_d) = c{\cdot}2^{-a}$ for almost all d. Let b be so large that $\text{eval}_a(x_d) = c \cdot 2^{-a}$ for all $d \geq a+b$. Then one can conclude that there is a y with $\varphi_a^B(y) < c_A(y)$ as otherwise there would be infinitely many stages s with $r_{\langle a,b,c\rangle,s+1} > r_{\langle a,b,c\rangle,s}$. ∎

Recall the definition of low and of high degrees in Remark 6.12. By Exercise 8.12, the maximal set B constructed in the previous theorem is dense simple; so for every recursive function f and almost all e, $x_{2e} > f(e)$. The function $x \mapsto x_{2e}$ is B-recursive and dominates all recursive functions. Hence B has high Turing degree by one of the characterizations given in Remark 6.12. So the maximal set B from the previous theorem has high Turing degree and is not Turing above the given r.e. set A. Hence $B <_T \mathbb{K}$ and it is shown that several high r.e. Turing degrees exist. The following corollary is the counterpart of Exercise 7.3.

Corollary 9.7. *There is a set B of high Turing degree such that $B <_T \mathbb{K}$.*

Comprehensive Exercise 9.8. The construction of the maximal set B not Turing above a given r.e. set A can be combined with permitting. So assume that E is a dense simple set and modify the permitting such that a marker x_e is permitted by E to move from $x_{e,s}$ to a new position $x_{e,s+1}$ iff there is an element $y \in E_{s+1} - E_s$ such that $|\{0,1,2,\ldots,y\} - E_{s+1}| \leq x_{e,s}$. Using this modified permitting, build a maximal set B such that $B \leq_T E$ and $A \not\leq_T B$.

Remark 9.9. The structure of the Turing degrees of r.e. sets is well-studied. Here some basic facts are given: Sacks [91] showed that whenever A, B are r.e. Turing degrees with $A <_T B$ then there are further incomparable r.e. sets E_1, E_2 with $A <_T E_k <_T B$ for $k = 1, 2$. So there are no minimal r.e. Turing degrees. But Lachlan [61] and Yates [118] constructed minimal pairs of r.e. Turing degrees, that is, they constructed r.e. and nonrecursive sets A, B such that every set E with $E \leq_T A \wedge E \leq_T B$ is already recursive. The sets A, B can even be chosen to have high Turing degree. A famous result is Lachlan's Non-Diamond Theorem [61] which shows that for every minimal pair A, B of r.e. sets, $\mathbb{K} \not\leq_T A \oplus B$. So either the join of two r.e. sets is Turing incomplete or they bound a nonrecursive set or both.

The Theorem of Friedberg and Muchnik can be proven by showing that there is a low r.e. set L which is not recurisive. The preceding result showed the existence of a high r.e. set which is not Turing complete (as all maximal sets have high Turing degree). Sacks [88] showed that there are much more jump-classes: every set which is r.e. relative $\mathbb{K}$ and above $\mathbb{K}$ is the jump of a (non-relativized) r.e. set. On the other hand, the jump of every set below $\mathbb{K}$ is r.e. relative to $\mathbb{K}$.

One can show that the structure of sets which are above $\mathbb{K}$ and r.e. relative to $\mathbb{K}$ is similarly rich as the structure of the (non-relativized) r.e. sets. Therefore one looked at r.e. sets A such that A' is low relative to $\mathbb{K}$, that is, $A'' \equiv_T \mathbb{K}'$. Such sets are called low_2. High and low_2 r.e. Turing degrees have many characterizations. For any given r.e. set A the following conditions hold:

- A has high Turing degree iff A is Turing equivalent to a maximal set;
- A has high Turing degree iff A is Turing equivalent to an r-maximal set;
- A has high Turing degree iff some A-recursive function dominates all recursive functions;
- A has low_2 Turing degree iff every coinfinite r.e. set $B \leq_T A$ has a maximal superset;
- A is low_2 iff some $\mathbb{K}$-recursive function dominates all A-recursive functions.

For this reason, the low_2 degrees are considered as a natural counterpart to the high degrees. One can generalize and has that a set A has low_n Turing degree iff the n-fold jumps of A and $\emptyset$ satisfy $A^{(n)} \equiv_T \emptyset^{(n)}$ and A has high_n Turing degree iff $A^{(n)} \geq_T \mathbb{K}^{(n)}$. No set has low_n and high_n Turing degree at the same time. There are r.e. sets which have neither low_n nor high_n Turing degree for every n; furthermore, for every n there are r.e. sets A, B such that A has high_{n+1} but not high_n Turing degree and B has low_{n+1} but not low_n Turing degree.

10 The Analytical Hierarchy

Up to now sets had been defined by quantifying over numbers only. In this section, one asks what happens if one quantifies over functions from $\mathbb{N}$ to $\mathbb{N}$. Here one can define a Σ^1_n formula inductively: Formulas with only existential quantifiers over functions are called Σ^1_1-formulas and with only universal quantifiers over functions are called Π^1_1-formulas. For example, giving a recursive enumeration A_s of A, let $\tilde{c}_A(x)$ be the first s with $A_s(x) = A(x)$. The following formula $\phi_{\tilde{c}_A}(f)$ defines when a function f equals $\tilde{c}_A$:

$$\phi_{\tilde{c}_A}(f) \Leftrightarrow \forall y \forall s \, [f(y) > 0 \Rightarrow y \in A_{f(y)} - A_{f(y)-1} \wedge f(y) = 0 \Rightarrow y \notin A_s].$$

Hence one can also define A via both, a Σ^1_1-formula and a Π^1_1-formula:

$$x \in A \Leftrightarrow \exists f \, [\phi_{\tilde{c}_A}(f) \wedge f(x) > 0] \Leftrightarrow \forall f \, [\phi_{\tilde{c}_A}(f) \Rightarrow f(x) > 0].$$

Recursively enumerable sets share the property of being in $\Sigma^1_1 \cap \Pi^1_1$ with many other sets. A typical situation for these sets is that they are computed relative to a unique infinite branch of a suitable r.e. tree. This has to be formalized in the next exercise.

Exercise 10.1. Given an r.e. set A, define a recursive tree $T \subseteq \mathbb{N}^*$ such that $\tilde{c}_A$ is the only infinite branch of T; that is, $\tilde{c}_A$ is the only function f such that $\forall x \, [f(0)f(1)f(2)\ldots f(x) \in T]$.

Remark 10.2. There is an enumeration $T_0, T_1, T_2, \ldots$ of all r.e. unbounded trees. Here "unbounded" means that every node can σ can have arbitrary many successors and not only $\sigma 0$ and $\sigma 1$. Hence $T_e \subseteq \mathbb{N}^*$ and not $T_e \subseteq \{0,1\}^*$. Recall that a subset of $\mathbb{N}^*$ is a tree iff for all $\sigma, \tau \in \mathbb{N}^*$ the implication $\sigma\tau \in T_e \Rightarrow \sigma \in T_e$ holds. A function f is an infinite branch of T_e iff $f(0)f(1)f(2)\ldots f(n) \in T_e$ for all n. A tree is called *well-founded* iff it has no infinite branches. If σ, τ nodes on a tree then the *Kleene-Brouwer ordering* $<_{KB}$ is defined by $\sigma <_{KB} \tau$ iff either σ extends τ as a string or the first x where $\sigma(x), \tau(x)$ differ satisfies $\sigma(x)\downarrow < \tau(x)\downarrow$. The importance of this ordering stems from the fact that the nodes of a tree T_x are well-ordered by $<_{KB}$ iff T_x is well-founded. Here $<_{KB}$ is an well-ordering means that there is no infinite descending chain of nodes, that is, no $\sigma_0, \sigma_1, \ldots \in T_x$ with $\sigma_{k+1} <_{KB} \sigma_k$ for all k.

Obervation 10.3: Suslin-Kleene-Classes and Δ^1_1-sets [53, 110]**.** For every r.e. tree T_e define

$$SK_e = \begin{cases} W_{f(0)}^{\{\langle 1, f(1)\rangle, \langle 2, f(2)\rangle, \ldots\}} & \text{if } f \text{ is the only infinite branch of } T_e; \\ \mathbb{N} & \text{if } T_e \text{ has 0 or at least 2 infinite branches.} \end{cases}$$

Now observe that the class of all SK_e has the following properties:

First: There is a recursive function g_1 such that $SK_{g_1(x)} = \{x\}$ for all x; that is, indices for the singleton sets can be effectively generated.

Second: There is a recursive function g_2 such that for all e, $SK_{g_2(e)} = \bigcup_{a \in W_e} SK_a$; that is, the class $\{SK_0, SK_1, SK_2, \ldots\}$ is closed under effective union. The idea is to build a tree $T_{g_2(e)}$ and to compute a decoding-programme d from e such that $T_{g_2(e)}$ has exactly one infinite branch f iff for every $a \in W_e$ the tree T_a has exactly one infinite branch; furthermore, one can compute from e a recursive one-one enumeration $\langle a_0, b_0\rangle, \langle a_1, b_1\rangle, \langle a_2, b_2\rangle, \ldots$ of $(W_e \cup \{u\}) \times \mathbb{N}$ and then define $T_{g_2(e)}$ such that the following conditions hold:

- $SK_u = \emptyset$ and T_u has exactly one infinite branch f_u;
- $T_{g_2(e)}$ has a unique infinite branch f iff for every $a \in W_e$ the tree T_a has a unique infinite branch f_a then $T_{g_2(e)}$ has exactly one infinite branch f and the branches f and f_a with $a \in W_e$ satisfy the equation $f(k+1) = f_{a_k}(b_k)$ for all k and $f(0)$ is a canonical index which enumerates the union of all sets which have the form
$$W_{f(k+1)}^{\{\langle b_\ell, f(\ell+1)\rangle : a_\ell = a_k \wedge b_\ell > 0\}}$$
for some k with $b_k = 0$. Here "canonical index" means an index produced by some fixed procedure to translate the above algorithm into an index.
- If some T_a with $a \in W_e$ has no infinite branch then $T_{g_2(e)}$ has also no infinite branch.
- If every T_a with $a \in W_e$ has an infinite branch and some T_a with $a \in W_e$ has several infinite branches then $SK_{g_2(e)}$ has also several infinite branches.
- The resulting set is defined as
$$SK_{g_2(e)} = \begin{cases} W_{f(0)}^{\{\langle 1, f(1)\rangle, \langle 2, f(2)\rangle, \ldots\}} & \text{if } f \text{ is the unique infinite branch of } T_{g_2(e)}; \\ \mathbb{N} & \text{otherwise.} \end{cases}$$

This is an outline how to construct $T_{g_2(e)}$, the missing part is to give the details when a node is in $T_{g_2(e)}$. This is roughly done by constructing for every k from σ the projected string σ_{a_k} from which the entry $\sigma(0)$ and every entry $\sigma(\ell+1)$ with $a_\ell \neq a_k$ is deleted and the remaining nodes are put into the order induced from the components b_ℓ; then $\sigma \in T_{g_2(e)}$ iff $\sigma_{a_k} \in T_{a_k}$ for all k where σ_{a_k} is not the empty string.

Third: If $SK_e = \mathbb{N}$ then SK_u is its complement. If $SK_e \neq \mathbb{N}$ then one can compute via a recursive function g_3 an index such that $SK_{g_3(e)} = \mathbb{N} - SK_e$. Given the unique infinite branch f of T_e the tree $T_{g_3(e)}$ is constructed such that it has a unique infinite branch f' satisfying that

- $f'(2m+1) = f(m)$;
- $f'(2m+2) = 0$ if $m \notin W_{f(0)}^{\{\langle k, f(k)\rangle : k>0\}}$ and $f'(2m+2) = s+1$ for s being the first stage with $m \in W_{f(0),s}^{\{\langle k, f(k)\rangle : k>0\}}$ otherwise;
- $f'(0)$ is a fixed index such that for all functions f'' $W_{f'(0)}^{\{\langle k, f''(k)\rangle : k>0\}}$ is the set of all m not in $W_{f''(1), f'(2m)}^{\{\langle k, f''(2k+1)\rangle : k>0\}}$.

Then $W_{f'(0)}^{\{\langle k, f'(k)\rangle : k>0\}} = ZK_{g_3(e)}$ is the complement of ZK_e.

This can be summarized as follows: SK is a class of sets together with some indexing such that SK contains an effective indexing of the singleton sets, is closed under complements and is closed under effective union. Based on work of Suslin [110], Kleene [53] showed that SK is indeed the smallest such class which exists. Furthermore, this class coincides with the intersection Δ_1^1 of Σ_1^1 and Π_1^1. This class SK or Δ_1^1 is also called the *class of hyperarithmetic sets*.

In the following, Kleene's $\mathcal{O}$ will be introduced which is the standard set of notations of recursive ordinals. Such notations represent the ordinals and it is unfortunately impossible to get unique representations for all sufficiently large ordinals. Therefore one has to work with notations in place of the ordinals themselves.

Definition 10.4. Kleene's set $\mathcal{O}$ of *Notations of Ordinals* is a subset of $\mathbb{N}$ given together with a partial ordering $<_o$ such that there is a recursive set limit, a recursive function succ and a recursive function embed satisfying the following for all x: if $x \in \mathcal{O}$ then $\{y : y <_o x\}$ is well-ordered and thus isomorphic to the order-type of an at most countable ordinal; if $x \in \mathcal{O}$ then $\{y : y <_o x\}$ is isomorphic to a limit ordinal iff $x \in$ limit; if $x \in \mathcal{O}$ then $x <_o \text{succ}(x)$ and every $y <_o \text{succ}(x)$ satisfies $y \leq_o x$, hence succ(x) represents the successor ordinal of x; if φ_e computes a well-ordering on $\mathbb{N}$ then the function $\varphi_{\text{embed}(e)}$ is an order-preserving mappping from $(\mathbb{N}, \varphi_e)$ into some well-ordered set of the form $\{y : y <_o x\}$ for some $x \in \mathcal{O}$.

Exercise 10.5. Let $\sqsubset$ be a universal partial ordering on $\mathbb{N}$ with the following properties: (a) There is an acceptable enumeration of partial orderings $<_e$ on $\mathbb{N}$ such that every r.e. partial ordering is captured by one ordering $<_e$ and $x <_e y$ iff $\langle e, x\rangle \sqsubset \langle e, y\rangle$. (b) $\langle d, x\rangle \sqsubset \langle e, y\rangle$ implies $d = e \vee d = 0$. (c) $\langle 0, 0\rangle \sqsubset \langle 0, 1\rangle \sqsubset \ldots$ and $\langle 0, x\rangle \sqsubset \langle e, y\rangle$ for all x, y and $e > 0$.

Furthermore let $|x|$ denote the ordinal isomorphic to $\{y : y \sqsubset x\}$ whenever this set is well-ordered. Let $E_0, E_1, E_2, \ldots$ be a canonical listing of all multisets of natural numbers. Whenever for each x_k occurring in E_e the set $\{z : z \sqsubset x_k\}$ is well-ordered

then let $|x_k|$ be the corresponding ordinal and whenever all x_i, x_j occurring in E_e are comparable with respect to $\sqsubset$ then let $e \in \mathcal{O}$ and let it represent the ordinal

$$\omega^{|x_1|} \cdot y_1 + \omega^{|x_2|} \cdot y_2 + \ldots + \omega^{|x_n|} \cdot y_n$$

in Cantor Normal Form where y_k is the multiplicity with which x_k occurs in E_e and $x_1, x_2, \ldots, x_n$ are ordered such that $x_n \sqsubset \ldots \sqsubset x_2 \sqsubset x_1$. Otherwise let $e \notin \mathcal{O}$.

Show that the functions succ and embed and the set limit exist as postulated in Definition 10.4. Show that furthermore all polynomials in ω (like $\omega^3 \cdot 2 + \omega^2$) have a unique representation in this variant of $\mathcal{O}$. Show that there is also a recursive set P such that for all $x \in \mathcal{O}$ [$x \in P$ iff x is an ω-power]. Furthermore let f be the recursive function which satisfies $E_{f(x,y)} = E_x + E_y$; that is, f makes a union of multisets which adds the multiplicities of the elements of the multisets E_x and E_y. So if 3 is 2 times in E_x and 5 times in E_y then 3 is 7 times in $E_{f(x,y)}$. Show that

$$\forall x, y \in \mathcal{O}\,[x \leq_o y \vee y \leq_o x \Rightarrow f(x,y) \in \mathcal{O} \wedge x \leq_o f(x,y) \wedge y \leq_o f(x,y)].$$

That is, f realizes the ordinal addition, provided that the notations for the ordinals are compatible.

Note that this is not Kleene's original coding as Kleene's original $\mathcal{O}$ does not permit to recover ω-powers in all cases. Kleene coded successors and limits of other ordinals explicitly using as indices powers of 2, 3 and 5.

Remark 10.6. For every notation x of a recursive ordinal α, that is, for every x such that $\{y : y <_o x\}$ is a set of the same order type as the well-ordered set $\{\beta : \beta < \alpha\}$, there is a tree T such that T has exactly one infinite branch with the following property:

$$f(\langle e, y\rangle) = \begin{cases} s+1 & \text{if } y <_o x \text{ and } s = \min\{t : e \in W_{e,t}^{\{\langle d,z,f(\langle d,z\rangle)\rangle : z <_o y\}}\}; \\ 0 & \text{otherwise.} \end{cases}$$

This set is called the α-jump (of the empty set). Although the notations are not unique, one can show that the α-jump is unique in the sense that if one uses different notations of ordinals $x, \tilde{x}$ then the resulting jumps based on x and $\tilde{x}$ are Turing equivalent. It can furthermore be shown that a set is hyperarithmetic iff it can be computed relative some α-jump of the empty set. One can generalize the construction in order to define the α-jump $A^{(\alpha)}$ of a set A.

A set A is *hyperarithmetic relative to* B iff there is a B-recursive ordinal α such that $A \leq_T B^\alpha$. The degrees induced by this reduction are called *hyperdegrees* and are a bit different from Turing degrees.

Exercise 10.7. Show that $\emptyset^{(\omega)} \not\leq \emptyset^{(n)}$ for any $n < \omega$ and conclude that there is a hyperarithmetic set which is not arithmetic, that is, which is not defined by a Σ^0_n-formula for some n.

Remark 10.8. The sets $\mathcal{O}$, $\{e : <_e \text{ is a well-ordering on } \mathbb{N}\}$ and $\{x : T_x \text{ is well-founded}\}$ are many-one complete for Π^1_1.

It is easy to see that these sets are all Π^1_1-sets. For example, T_x is well-founded iff $\forall f\, \exists x\, \forall s\, [f(0)f(1)f(2)\ldots f(x) \notin T_{x,s}]$ where $T_{x,s}$ is the number of nodes enumerated into T_x within s steps.

One can also check whether $<_e$ is a well-ordering: $<_e$ must be irreflexive, that is, $\forall x\, [\neg(x <_e x)]$; $<_e$ must be transitive, that is, $\forall x, y, z\, [x <_e y \wedge y <_e z \Rightarrow x <_e z]$; $<_e$ must not have infinite descending chains, that is, $\forall f\, \exists n\, [\neg(f(n+1) <_e f(n))]$.

A sample reduction is that from the index-set of well-founded trees to that of well-orders: Given T_x one can easily define an enumeration $\sigma_0, \sigma_1, \sigma_2, \ldots$ of the union of T_x with all nodes of height 0 or 1; these nodes are added in order to avoid problems when T_x is finite. Now one can define an ordering $<_{g(e)}$ as

$$i <_{g(e)} j \Leftrightarrow \sigma_i <_{KB} \sigma_j$$

and the resulting ordering is a well-ordering iff T_x is well-founded, that is, has no infinite branch.

Remark 10.9. A Σ^1_n-formula is a formula with quantification over number-variables and function-variables and with a recursive predicate such that either there is a first existentially quantified function-variable with $n-1$ alternations of quantifiers following or there are less than $n-1$ alternations of quantifiers over function-variables or there are no function-quantifiers at all. Similarly one defines a Π^1_n-formula with requiring "universal quantification" in place of "existential quantification". Here are some examples of Σ^1_3-formulas:

$$\begin{aligned}
\phi_1(f,g,x) &\Leftrightarrow \exists h_1\, \forall h_2\, \forall\, h_3\, \exists h_4\, [f(h_1(h_2(x))) = g(h_3(h_4(x)))];\\
\phi_2(f,g,x) &\Leftrightarrow \exists y_1 \forall y_2\, \exists y_3\, \forall y_4\, \exists y_5\, \forall h\, [f(y_1+y_2+y_3+y_4+y_5) = g(h(x+y_1))];\\
\phi_3(f,g,x) &\Leftrightarrow \forall y\, [f(y) = g(x)].
\end{aligned}$$

Note that Σ^1_3-formulas do not say anything about the number-variables occurring and also do not require that there are functional quantifiers; 3 is just an upper bound but not a lower bound on the number of blocks of the same type of function-quantifiers.

Furthermore one can also define classes of functions. A class S is a Σ^1_n-class of functions iff there is a Σ^1_n-formula ϕ with one free variable f such that $S = \{f : \phi(f)\}$. Analogously one defines Π^1_n-classes. These definitions can be relativized to an oracle

A so that one obtains $\Sigma^{1,A}_n$-classes and $\Pi^{1,A}_n$-classes.

There is a close connection to the classical theory of Borel classes: A class S is Borel iff there is an oracle A and a Σ^1_1-formula $\exists g\,[\phi^A(f,g)]$ and a Π^1_1-formula $\forall g\,[\psi^A(f,g)]$ such that

$$S = \{f : \exists g\,[\phi^A(f,g)]\} = \{f : \forall g\,[\psi^A(f,g)]\}.$$

To understand this characterization, recall the Borel classes are the smallest collection of classes such that all classes of the form $S_{m,n} = \{f : f(n) = m\}$ are Borel classes, with every class S also the complement of S is a Borel class and with every countable list $S_0, S_1, \ldots$ of Borel classes also their union $S_0 \cup S_1 \cup \ldots$ is a Borel class.

On every level of the analytical hierarchy they are many-one complete sets. The Σ^1_n-complete and Π^1_n-complete sets have the same Turing degree but different many-one degrees; but they are all many-one reducible to the Σ^1_{n+1}-complete and Π^1_{n+1}-complete sets.

Remark 10.10. Shoenfield [99] considered the reducibility given as $A \leq_{\Delta^1_n} B$ iff A is a $\Sigma^{1,B}_n$-set and $\Pi^{1,B}_n$-set. He showed that this reducibility is transitive for every fixed parameter n. It should be noted that the statement $A \leq_{\Delta^1_1} B$ is equivalent to A being hyperarithmetic relative to B.

Exercise 10.11. Let $<_0, <_1, <_2, \ldots$ be a numbering of all r.e. partial orderings where $x <_e y$ which are given by r.e. sets $V_0, V_1, V_2, \ldots$ such that $x <_e y \Leftrightarrow \langle x, y\rangle \in V_e$. Here the sets V_e are obtained from W_e as follows: V_e is the transitive closure of the union of all sets $W_{e,s}$ where the transitive closure of the pairs enumerated into $W_{e,s}$ does not cause that some element x is below itself; thus $x \not<_e x$ remains guaranteed.

It had been mentioned above that $\{e : <_e \text{ is a well-ordering}\}$ is Π^1_1-complete. What about $\{e : <_e \text{ does not have an infinite descending chain}\}$, $\{e : <_e \text{ has no infinite ascending chain}\}$, $\{e : <_e \text{ has no infinite antichain}\}$, $\{e : <_e \text{ is a linear ordering}\}$? Recall that an ordering is linear if each two elements are either equal or one is strictly below the other. An antichain is a set A such that for all $x, y \in A$ neither $x <_e y$ nor $y <_e x$.

Exercise 10.12. There is a first ordinal ω_{CK} which cannot be represented by any recursive well-ordering or notation of ordinals. Assume now that A is an oracle such that ω_{CK} is A-recursive; that is, that there is some A-recursive well-ordering which is order-isomorphic to the well-ordered set $\{\alpha : \alpha < \omega_{CK}\}$. Now show that the Π^1_1-complete set

$$\{e : <_e \text{ is a well-ordering}\}$$

is Δ^1_1-reducible to A.

11 Algorithmic randomness

The central question of Algorithmic Randomness is "What is random?" The answer to this will depend first on the object under investigation: is it a number (= finite string) or is it an infinite binary sequence? There are two approaches: the classical one and the algorithmic one. The classical approach considers the way distributions are generated and how likely a number is chosen in a distribution; the algorithmic way more looks at the probability that one can extract nontrivial information from a sequence or that one describe it in a nontrivial way. The first could be most easily be explained using the example of gambling: if the sequence of numbers drawn by a gambling automaton can be predicted so often that some gambler will make more and more profits than the sequence is not random; so the casino relies on the fact that its gambling machines use sequences of random numbers which guarantee that the casino and not the gambler will win on the long term. The second could be most easily explained as follows: the number `18446744073709551616` would be nonrandom as one could describe this number as `2^64` while for a random number like `56913576862312945` there would be no accepted short description. It will turn out later that both concepts are connected with each other and investigating these connections is part of the field of Algorithmic Randomness or Kolmogorov complexity. Although this field has several founders including Gregory Chaitin [20], Per Martin-Löf [67], Claus-Peter Schnorr [94] and Ray Solomonoff [105], it is named after Andrei N. Kolmogorov [54] who is known for his outstanding contributions to the theories of probability and description complexity.

Convention. Recall that η_a is the a-th binary string which is obtained by taking the digits after the leading 1 of the binary representation of $a+1$. So η_0 is the empty string and η_{37} is 00110 as 38 has the binary representation 100110. Within this section there occur many sets of pairs $\langle e, \eta_a \rangle$ which formally should be written as sets of pairs $\langle e, a \rangle$ with a representing η_a. In other words, often an r.e. set of numbers is identified with the set

$$\{\langle e, \eta_a \rangle : \langle e, a \rangle \in E\}$$

which is more convenient to handle.

Definition 11.1. A betting strategy or martingale is a function mg from binary strings into the nonnegative real numbers such that

$$\forall \sigma \in \{0,1\}^* \, [\mathrm{mg}(\sigma) = \tfrac{1}{2}(\mathrm{mg}(\sigma 0) + \mathrm{mg}(\sigma 1))].$$

The martingale mg is recursive iff the set

$$\{\langle e, \sigma \rangle : \mathrm{mg}(\sigma) > q_e\}$$

is recursive and the martingale mg is r.e. if this set is r.e.; here $q_{\langle i,j\rangle} = \frac{i}{j+1}$.

Note that q_b is the b-th positive rational number in some straight-forward numbering of the non-negative rational numbers. Other conventions to number the non-negative rational numbers are possible.

The intuitive idea behind a martingale is that $\mathrm{mg}(\sigma)$ denotes the amount of money owned by a gambler after having bet on the first $|\sigma|$ bits such that the outcome is σ. The underlying betting system is fair and does not take any provisions. So the gambler can be any real number r with $0 \leq r \leq \mathrm{mg}(\sigma)$ on the next bit being 1 and the martingale is then updated as follows:

- $\mathrm{mg}(\sigma 1) = \mathrm{mg}(\sigma) + r$;
- $\mathrm{mg}(\sigma 0) = \mathrm{mg}(\sigma) - r$.

Here a bet on 0 instead of 1 could just be realized by using a value r with $-\mathrm{mg}(\sigma) \leq r \leq 0$ and then applying the same update rules. The update rules then guarantee the main martingale equality:

$$\forall \sigma \in \{0,1\}^* \, [\mathrm{mg}(\sigma) = \tfrac{1}{2} \cdot (\mathrm{mg}(\sigma 0) + \mathrm{mg}(\sigma 1))].$$

Being successful means that during his (infinite) life the gambler becomes richer and richer (although he might be poor inbetween); a set A is random iff no "algorithmic gambler" can succeed with betting on A's characteristic sequence. This is formalized in the next definition.

Definition 11.2. A martingale mg succeeds on a set A iff

$$\forall e \, \exists n \, [\mathrm{mg}(A(0)A(1)\ldots A(n)) > q_e];$$

a set A is recursively random iff no recursive martingale succeeds on A and Martin-Löf random iff no r.e. martingale succeeds on A.

Note that the money can be broken down to arbitrary small units and that there are gambling strategies which exploit this. For example there is a gambling strategy succeeding on all finite sets by letting $\mathrm{mg}(\sigma) = 2^{-|\sigma|} \cdot 3^{|\{x:\sigma(x)\downarrow=1\}|}$. The next exercise shows that for recursive martingales, one can enforce that the martingale takes rational values although one cannot enforce that the martingale takes natural numbers as values; hence it is essential to break money down to arbitrary small units.

Exercise 11.3. Let mg be a given recursive martingale. Show that there is a recursive function f and a martingale mg' such that

- $\forall a\,[q_{f(a)} = \mathrm{mg}'(\eta_a)]$, that is, f codes mg' and mg' takes rational values;
- $\forall A\,[\text{if mg succeeds on } A \text{ then mg}' \text{ succeeds on } A]$.

Show furthermore that no $\mathbb{N}$-valued recursive martingale succeeding on any 2-generic set. Note that Proposition 11.10 below shows that there is a recursive martingale succeeding on all 1-generic and thus 2-generic sets. Hence one cannot enforce that a martingale is $\mathbb{N}$-valued.

Remark 11.4. Martin-Löf [67] introduced randomness not with martingales but with tests. Here a randomness test is an r.e. set RT such that for all e,

$$\mu\{A : \exists n\,[\langle e, A(0)A(1)A(2)\ldots A(n)\rangle \in RT]\} \leq 2^{-e}$$

where μ is the standard measure on the Cantor space $\{0,1\}^\infty$; this measure satisfies for all partial $\{0,1\}$-valued functions θ that

$$\mu\{A : \forall x \in \mathrm{dom}(\theta)\,[\theta(x) = A(x)]\} = 2^{-|\mathrm{dom}(\theta)|}.$$

One can show that for every randomness test RT the class of sets which are covered by the randomness test has measure 0: For every e the measure of the class of sets A such that there are $d > e$ and n with $\langle d, A(0)A(1)A(2)\ldots A(n)\rangle \in RT$ is at most $2^{-e-1} + 2^{-e-2} + \ldots = 2^{-e}$; hence for every e the measure of the class of all sets covered by RT is at most 2^{-e}. In contrast to this, the whole Cantor space has measure 1. So every randomness test covers only a tiny fraction of the Cantor space.

Exercise 11.5. For a given randomness test, one can construct a further test RT such that the following four conditions hold:

- RT and the given test cover exactly the same sets;
- For every set A and every d, e with $d < e$, if $\langle e, A(0)A(1)A(2)\ldots A(n)\rangle \in RT$ for some n then $\langle d, A(0)A(1)A(2)\ldots A(m)\rangle \in RT$ for some m as well;
- For every A and every e there is at most one n with $\langle e, A(0)A(1)A(2)\ldots A(n)\rangle \in RT$;
- For every e, $\sum_{\sigma:\langle e,\sigma\rangle\in RT} 2^{-|\sigma|} \leq 2^{-e}$.

Note that the fourth condition is a consequence of the third, so one has to make sure in the construction that the first three conditions are satisfied and then to show that the fourth condition follows from the third. The second condition enforces that the component number $e+1$ of the test contains only sets which are also in the component

number e of the test. Hence RT covers a set A iff for all e there exists an n such that $\langle e, A(0)A(1)A(2)\ldots A(n)\rangle \in RT$.

Theorem 11.6. *Let $\mathcal{C}$ be a class of sets. Then there is a randomness test RT covering every $A \in \mathcal{C}$ iff there is an r.e. martingale succeeding on every $A \in \mathcal{C}$.*

Proof. Assume that mg is an r.e. martingale which succeeds on every set in $\mathcal{C}$. Now one can define a randomness test RT based on mg by letting

$$RT = \{\langle e, \sigma\rangle : \mathrm{mg}(\sigma) > 2^e \cdot r\}$$

where r is a fixed rational number with $\mathrm{mg}(\eta_0) \leq r$. One can easily see the following: RT is an r.e. set and a set A is covered by RT iff the martingale mg succeeds on A. Hence RT covers every set in $\mathcal{C}$.

For the converse direction assume that some randomness which covers every set in $\mathcal{C}$. There is by Exercise 11.5 a further randomness test RT such that

- RT covers every set in $\mathcal{C}$;
- For every A and every e there is at most one n with $\langle e, A(0)A(1)A(2)\ldots A(n)\rangle \in RT$;
- For every e, $\sum_{\sigma:\langle e,\sigma\rangle\in RT} 2^{-|\sigma|} \leq 2^{-e}$.

Now define for every a the martingale mg_a by taking on each input the first case which applies:

$$\mathrm{mg}_a(\sigma) = \begin{cases} 2^{|\sigma|-|\eta_a|} & \text{if } \eta_a \text{ extends } \sigma; \\ 1 & \text{if } \sigma \text{ extends } \eta_a; \\ 0 & \text{otherwise.} \end{cases}$$

The martingale mg is the sum of those mg_a selected by RT as follows:

$$\mathrm{mg}(\sigma) = \sum_{\langle e,\eta_a\rangle\in RT\wedge e>0} \mathrm{mg}_a(\sigma).$$

Now it is shown that for every σ this sum converges and satisfies the bound

$$\mathrm{mg}(\sigma) \leq 2^{|\sigma|}.$$

To see this, note that for each $e > 0$,

$$\sum_{a:\langle e,\tau\rangle\in RT} 2^{|\sigma|-|\tau|} \leq 2^{|\sigma|} \cdot \sum_{a:\langle e,\tau\rangle\in RT} 2^{-|\tau|} \leq 2^{|\sigma|-e}$$

by the choice of RT as above. Hence $\mathrm{mg}(\sigma)$ is bounded by the sum of $2^{|\sigma|-e}$ over all positive e which is $2^{|\sigma|}$. The main equation of martingales holds for all mg_a, that is,

$$\forall a \in \mathbb{N} \forall \sigma \in \{0,1\}^* [\mathrm{mg}_a(\sigma) = \tfrac{1}{2}(\mathrm{mg}_a(\sigma 0) + \mathrm{mg}_a(\sigma 1))].$$

Hence it also holds for the sum of these mg_a, that is,

$$\forall \sigma \in \{0,1\}^* [\mathrm{mg}(\sigma) = \tfrac{1}{2}(\mathrm{mg}(\sigma 0) + \mathrm{mg}(\sigma 1))].$$

To see that $\{\langle e, \sigma \rangle : \mathrm{mg}(\sigma) > q_e\}$ is r.e., note that $\mathrm{mg}(\sigma) > q_e$ iff there is a finite subset D of RT with $\sum_{\langle e,\eta_a \rangle \in D} \mathrm{mg}_a(\sigma) > q_e$. For each finite set D, this condition can be checked effectively.

It remains to be shown that mg succeeds on every set covered by RT. So let A be a set covered by RT. For each $c \in \mathbb{N}$ there is an $n \in \mathbb{N}$ such that there are at least c different pairs $\langle e_1, \eta_{a_1} \rangle, \langle e_2, \eta_{a_2} \rangle, \ldots, \langle e_c, \eta_{a_c} \rangle \in RT$ with $A(0)A(1)A(2)\ldots A(n)$ extending $\eta_{a_1}, \eta_{a_2}, \ldots, \eta_{a_c}$ and $e_1, e_2, \ldots, e_c > 0$. It follows that

$$\mathrm{mg}(A(0)A(1)A(2)\ldots A(n)) \geq \sum_{d=1,2,\ldots,c} \mathrm{mg}_{a_d}(A(0)A(1)\ldots A(n)) = c.$$

Hence mg converges on A to ∞ and so mg succeeds on A. ∎

Exercise 11.7. Martin-Löf [67] showed that there is a universal randomness test. Prove his result by doing the following steps:

- Show that there is a recursive function f such that $W_{f(e)}$ is a randomness test for each e and that $W_{f(e)} = W_e$ whenever already W_e is a randomness test;
- Let $URT = \{\langle d, \sigma \rangle : \exists e [\langle d + e + 1, \sigma \rangle \in W_{f(e)}]\}$;
- Show that URT is a randomness test which covers every set covered by any randomness test. This property is called *universal*.

Conclude that the measure of all sets covered by some randomness test is 0. Conclude furthermore that there is a universal r.e. martingale which succeeds on every set on which some r.e. martingale succeeds.

Remark 11.8. Given the universal randomness test URT, one can define a recursive binary tree T such that A is an infinite branch of T iff there is no e, σ such that A extends σ, $e > 1$ and $\langle e, \sigma \rangle \in URT$. So T could be defined as

$$T = \{\sigma \in \{0,1\}^* : \forall e \in \{2,3,4,\ldots,|\sigma|\} \forall k < |\sigma| [\langle e, \sigma(0)\sigma(1)\ldots\sigma(k) \rangle \notin URT_{|\sigma|}]\}$$

where URT_s is the set of all pairs enumerated into the r.e. set URT within s steps.

Remark 11.9. An important question is which set is random and which not. Some easy facts are the following: For every recursive martingale mg there is a recursive set R such that mg does not succeed on R; the idea is to choose R inductively in an adversary manner such that

$$R(n+1) = \begin{cases} 0 & \text{if } \mathrm{mg}(R(0)R(1)R(2)\ldots R(n)0) < \mathrm{mg}(R(0)R(1)R(2)\ldots R(n)1); \\ 1 & \text{if } \mathrm{mg}(R(0)R(1)R(2)\ldots R(n)0) \geq \mathrm{mg}(R(0)R(1)R(2)\ldots R(n)1). \end{cases}$$

Then mg does not increase on R and thus mg does not succeed on R. On the other hand, for every recursive set A there following martingale mg_A succeeds on A and no other set:

$$\mathrm{mg}_A(\sigma) = \begin{cases} 2^{|\sigma|} & \text{if } A \text{ extends } \sigma; \\ 0 & \text{otherwise.} \end{cases}$$

Let mg' be the universal r.e. martingale mg' from Exercise 11.7. Then mg' succeeds on every recursive set. But there is a $\mathbb{K}$-recursive set on which mg' does not succeed, this set is defined from mg' in the same way as the recursive counterexample R above was defined from the recursive martingale mg.

Proposition 11.10. *No 1-generic set is recursively random.*

Proof. Let $\mathrm{mg}(\sigma) = 2^{-|\sigma|} \cdot 3^{|\{x:\sigma(x)\downarrow=1\}|}$ and define the recursive set W of all strings σ such that at least two thirds of the bits in σ are 1. Note that W is dense, that is, every string τ has an extension in W, namely $\tau 1^{2|\tau|}$. For every $\sigma \in W$, $\mathrm{mg}(\sigma) \geq 1.125^{|\sigma|/3}$, hence mg succeeds on every set G where there are infinitely many n with $G(0)G(1)G(2)\ldots G(n) \in W$. A 1-generic set G satisfies this property as for every m the set $W - \{\sigma : G(0)G(1)G(2)\ldots G(m) \text{ extends } \sigma\}$ is a recursive and dense set of strings and thus there is an $n > m$ with $G(0)G(1)G(2)\ldots G(n) \in W$. Hence mg succeeds on all 1-generic sets. ∎

Exercise 11.11. Extend this result to the following: If G is 1-generic and $A \leq_T G$ then A is not Martin-Löf random. So assume that $A = \varphi_e^G$ and consider the following set RT: Let W be the r.e. set of all pairs $\langle d, \tau \rangle$ such that there is a $\sigma \in \{0,1\}^d$ for which τ is the first extension found such that $0, 1, 2, \ldots, 2d \in \mathrm{dom}(\varphi_e^\tau)$. Now let $RT = \langle d, \varphi_e^\tau(0)\varphi_e^\tau(1)\varphi_e^\tau(2)\ldots\varphi_e^\tau(2d)\rangle : \langle d, \tau\rangle \in W$. Show that RT is a randomness test, that W is r.e. and that G does not strongly avoid W. Conclude that G meets W infinitely often and that hence A is covered by RT.

One can relativize the definition of randomness: A is Martin-Löf random relative to B iff there is no B-r.e. martingale mg^B which succeeds on A. Van Lambalgen proved the following famous theorem, which is stated without proof.

Theorem 11.12. *Given two sets A, B, the join $A \oplus B$ is Martin-Löf random iff A is Martin-Löf random relative to B and B is Martin-Löf random iff A is Martin-Löf random and B is Martin-Löf random relative to A.*

Besides randomness of infinite objects (sets of natural numbers) one considers also the randomness of finite objects or strings. The problem is that one cannot really say whether a particular string or natural number is random as this depends on the system used to measure randomness. But one can introduce a function C mapping each string σ to its complexity such that any further measure differs from C only by a constant. The measure C is based on a universal function U_C by

$$C(\sigma) = \min\{|\tau| : U_C(\tau) = \sigma\}$$

where U_C is a partial-recursive function from $\{0,1\}^*$ to $\{0,1\}^*$. Here U_C is universal if for every further partial-recursive function $V : \{0,1\}^* \to \{0,1\}^*$ there is a constant c such that

$$\forall \tau \in \mathrm{dom}(V)\, [C(V(\tau)) \leq |\tau| + c],$$

in other words, using V instead of U_C would improve the complexity of each string at most by the constant c. The value $C(\sigma)$ is called the Kolmogorov complexity of σ.

Besides the plain Kolmogorov complexity C one also considers the prefix-free complexity H which was in particular studied by Chaitin and Levin. This is based on a machine U_H which is a universal prefix-free function. Here "prefix-free" means that there are no $\tau, \tau' \in \mathrm{dom}(U_H)$ such that τ' properly extends τ. Again

$$H(\sigma) = \min\{|\tau| : U_H(\tau) = \sigma\}$$

and U_H is universal means that for every prefix-free partial-recursive function $V : \{0,1\}^* \to \{0,1\}^*$ there is a constant c such that

$$\forall \tau \in \mathrm{dom}(V)\, [H(V(\tau)) \leq |\tau| + c],$$

Note that Chaitin uses the letter K for the plain complexity C in several papers; Downey, Hirschfeldt and Nies use K for the prefix-free complexity H in several recent papers. The preferences are not the same in all papers of the same author as different co-authors impose different constraints.

The complexities can also be defined for numbers by simply defining $C(a) = C(\eta_a)$ and $H(a) = H(\eta_a)$. This permits to iterate them and to interpret expressions like $C(C(H(x)))$. Furthermore, one defines $|a| = |\eta_a|$ in order to extend the length-function to numbers as well. Note that $|a| = \max\{n : 2^n - 1 \leq a\}$ and therefore $|a|$ and $\log(a)$ differ for positive natural numbers at most by 1.

Exercise 11.13. To see that a universal function U_C exists, consider a recursive enumeration $\tilde{\varphi}_e$ of all partial-recursive functions from $\{0,1\}^* \to \{0,1\}^*$; if $\varphi_e(a)$ is defined then $\tilde{\varphi}_e(\eta_a) = \eta_{\varphi_e(a)}$ else $\tilde{\varphi}_e(\eta_a)$ is undefined. Now one can define a universal function U by letting $U(0^e 1\sigma) = \tilde{\varphi}_e(\sigma)$. Given any V, it equals to some function $\tilde{\varphi}_e$; so $V(\sigma) = U(0^e 1\sigma)$ and $C(V(\sigma)) \leq e+1+|\sigma|$ for all $\sigma \in \mathrm{dom}(V)$. Transfer this proof from C to H to show that U_H exists. When transferring the proof, take care of that the underlying enumeration replacing $\tilde{\varphi}_0, \tilde{\varphi}_1, \tilde{\varphi}_2, \ldots$ consists of prefix-free functions only.

Exercise 11.14. Show that there is a constant c with $C(\sigma) \leq |\sigma| + c$ for all $\sigma \in \{0,1\}^*$ and that for every n there is a $\sigma \in \{0,1\}^n$ with $C(\sigma) \geq n$. Furthermore find a recursive function f such that for infinitely many n the value $f(n)$ differs from $\max\{H(\sigma) : \sigma \in \{0,1\}^n\}$ by at most $\log\log(n)$. Note that one cannot find a recursive f such that this difference is bounded by a constant.

Theorem 11.15. *Let f be a partial-recursive function with one and g a partial-recursive function with n inputs. Then there is a constant c such that for all inputs $x, y_1, y_2, \ldots, y_n$,*

$$\begin{aligned} C(f(x)) &\leq C(x) + c; \\ H(g(y_1, y_2, \ldots, y_n)) &\leq H(y_1) + H(y_2) + \ldots + H(y_n) + c. \end{aligned}$$

It is not possible to extend the formula for C to n-ary functions as there will be some logarithmic term which is impossible to avoid.

Proof. The more involved result for H is proven. Let

$$V(\tau) = \begin{cases} g(U_H(\tau_1), U_H(\tau_2), \ldots, U_H(\tau_n)) & \text{if } \tau = \tau_1\tau_2\ldots\tau_n \text{ and} \\ & \tau_1, \tau_2, \ldots, \tau_n \in \mathrm{dom}(U_H); \\ \uparrow & \text{if there are no such } \tau_1, \tau_2, \ldots, \tau_n. \end{cases}$$

Note that for any τ there is at most one n-tuple $(\tau_1, \tau_2, \ldots, \tau_n)$ of n strings in the domain of U_H such that $\tau = \tau_1\tau_2\ldots\tau_n$. The reason is that τ_1 is the unique prefix of $\tau_1\tau_2\ldots\tau_n = \tau$ such that $U_H(\tau_1)$ is defined; then $\tau_2\ldots\tau_n$ is the remaining part and τ_2 is the unique prefix of this remaining string such that $U_H(\tau_2)$ is defined; so with this method one can determine all n parts of the full input $\tau_1\tau_2\ldots\tau_n$. This method makes V also to be prefix-free. Hence there is a constant c such that

$$H(V(\tau_1\tau_2\ldots\tau_n)) \leq |\tau_1\tau_2\ldots\tau_n| + c = |\tau_1| + |\tau_2| + \ldots + |\tau_n| + c.$$

for all strings $\tau_1, \tau_2, \ldots, \tau_n \in \mathrm{dom}(U_H)$. Considering now any inputs $y_1, y_2, \ldots, y_n$ for g, let τ_k be the shortest string with $U_H(\tau_k) = y_k$. Then it follows that

$$H(g(y_1, y_2, \ldots, y_n)) \leq H(y_1) + H(y_2) + \ldots + H(y_n) + c$$

which gives the desired inequality. The result for C is proven correspondingly. ∎

Exercise 11.16. Consider the following two-place function: $f(\sigma,\tau) = \eta_{|\sigma|}\sigma\tau$. Show that there is a constant c such that there are infinitely many pairs (σ,τ) with $C(\sigma,\tau) \geq C(\sigma)+C(\tau)+\log(|\sigma|+|\tau|)-c$. Note that $C(\sigma)$ and $C(\eta_{|\sigma|}\sigma)$ differ only by a constant; is it possible to formulate a similar result for the string-concatenation function?

Proposition 11.17. *There is no recursive function g with $C(g(x)) \geq x$ for all x. The same holds with "H" in place of "C".*

Proof. If g would exist then there would be a constant c with $C(g(2^x)) \leq x+c$ while $C(g(2^x)) \geq 2^x$ as well. This could work out only for finitely many x, a contradiction. ∎

Theorem 11.18. *The overgraphs $\{\langle e,\sigma\rangle : C(\sigma) \leq e\}$ of C and $\{\langle e,\sigma\rangle : H(\sigma) \leq e\}$ of H are recursively enumerable; that is, C and H can be approximated from above.*

Proof. The proof is given for C; the one for H is similar. As U_C is partial-recursive, there is a recursive enumeration $p_0, p_1, p_2, \ldots$ of the domain of U_C. Furthermore, it follows from the universality of U_C that for every x there is an n with $U_C(p_n) = x$. Thus one can define an approximation C_s to C from above using an auxiliary function f_C and derive that the overgraph is recursively enumerable:

$$\begin{aligned} f_C(\sigma) &= \min\{n : U_C(p_n) = \sigma\}; \\ C_s(\sigma) &= \min\{|p_m| : m \leq f_C(\sigma) + s \wedge U_C(p_m) = \sigma\}; \\ \{\langle e,\sigma\rangle : C(\sigma) \leq e\} &= \{\langle e,\sigma\rangle : \exists s\,[C_s(\sigma) \leq e]\}. \end{aligned}$$

Taking a recursive enumeration of the domain of U_H, these definitions can be done in parallel to the case for C:

$$\begin{aligned} f_H(\sigma) &= \min\{n : U_H(q_n) = \sigma\}; \\ H_s(\sigma) &= \min\{|q_m| : m \leq f_H(\sigma) + s \wedge U_H(q_m) = \sigma\}; \\ \{\langle e,\sigma\rangle : H(\sigma) \leq e\} &= \{\langle e,\sigma\rangle : \exists s\,[H_s(\sigma) \leq e]\}. \end{aligned}$$

This shows that both overgraphs are recursively enumerable. ∎

Future references to approximations C_s and H_s of C and H, respectively, from above will be defined as in the proof of Theorem 11.18.

Exercise 11.19. Prove the Kraft-Chaitin Theorem which says that whenever there is an r.e. set E of pairs such that $\sum_{\langle e,\sigma\rangle \in E} 2^{-e} < \infty$ then there is a constant c such that $H(\sigma) \leq e + c$ for all pairs $\langle e,\sigma\rangle \in E$. These pairs $\langle \sigma, e\rangle$ in E are often called

axioms. For a given set E, let $\sum_{\langle e,\sigma\rangle\in E} 2^{-e}$ be the *Kraft-Chaitin sum*. The set E is called a *Kraft-Chaitin set* iff E is r.e. and the Kraft-Chaitin sum is finite, that is, $\sum_{\langle e,\sigma\rangle\in E} 2^{-e} < \infty$. Kraft-Chaitin sets are a useful tool to get uniform upper bounds on $H(\sigma)$ for an infinite number of strings σ.

Exercise 11.20. The following sets are called the *sets of compressible strings* or *sets of nonrandom numbers*. The definitions are parallel for C and H:

$$\begin{aligned} NRC &= \{x : C(x) < |x|\}; \\ NRH &= \{x : H(x) < |x|\}. \end{aligned}$$

Show that NRC (nonrandom numbers for C) and NRH (nonrandom numbers for H) are both simple, not hypersimple and do not have a maximal superset. For further information (which is not part of the exercise) note that there is one difference between NRC and NRH: NRH has r-maximal supersets but NRC has not.

There is a close connection between Martin-Löf randomness of a set A on one side and the function mapping n to the Kolmogorov complexity of $A(0)A(1)A(2)\ldots A(n)$ on the other side. The second and third characterization were obtained independently by Levin [64] and Schnorr [94, 95], the fourth and fifth characterization are due to Miller and Yu [72].

Theorem 11.21: Characterizing Randomness [64, 72, 94, 95]**.** *The following conditions are equivalent for a set A:*

1. *A is Martin-Löf random;*
2. *$H(A(0)A(1)A(2)\ldots A(n)) \geq n$ for almost all n;*
3. *There is a constant c such that $H(A(0)A(1)A(2)\ldots A(n)) \geq n - c$ for all n;*
4. *$\sum_{n\in\mathbb{N}} 2^{n-H(A(0)A(1)A(2)\ldots A(n))} < \infty$;*
5. *For every recursive function g satisfying the condition $\sum_n 2^{-g(n)} < \infty$ it holds that $\forall^\infty n\,[C(A(0)A(1)A(2)\ldots A(n)) \geq n - g(n)]$.*

Proof. Assume that A is random, that is, that Condition 1 holds. Now modify the definition of mg_a to the following definition of mg_p for $p \in \mathrm{dom}(U_H)$:

$$\mathrm{mg}_p(\sigma) = \begin{cases} 2^{|\sigma|-|p|} & \text{if } U_H(p) \text{ extends } \sigma; \\ 2^{|U_H(p)|-|p|} & \text{if } \sigma \text{ extends } U_H(p); \\ 0 & \text{otherwise.} \end{cases}$$

Now let

$$\mathrm{mg}(\sigma) = \sum_{p \in \mathrm{dom}(U_H)} \mathrm{mg}_p(\sigma).$$

Recall that $\sum_{p \in \mathrm{dom}(U_H)} 2^{-|p|} \leq 1$. This gives directly that $\mathrm{mg}(\sigma) \leq 2^{|\sigma|}$ for all σ. Hence $\mathrm{mg}(\sigma) < \infty$ for all $\sigma \in \{0,1\}^*$ and mg is a martingale as seen in similar proofs before.

Now let $P = \{p \in \mathrm{dom}(U_H) : A \text{ extends } U_H(p)\}$. There is a constant c such that mg does on A never go above this constant c. It is easy to see that for every finite subset Q of P and all sufficiently large n,

$$\begin{aligned} c &\geq \mathrm{mg}(A(0)A(1)A(2)\ldots A(n)) \\ &\geq \sum_{p \in Q} \mathrm{mg}_p(A(0)A(1)A(2)\ldots A(n)) = \sum_{p \in Q} 2^{|U_H(p)|-|p|}. \end{aligned}$$

Thus it follows that

$$\sum_{p \in Q} 2^{|U_H(p)|-|p|} \leq c.$$

As this holds for all finite subsets Q of P,

$$\sum_{p \in P} 2^{|U_H(p)|-|p|} \leq c.$$

For every n there is a shortest program $p_n \in \mathrm{dom}(U_H)$ such that

- $|p_n| = H(A(0)A(1)A(2)\ldots A(n))$ and
- $U_H(p_n) = A(0)A(1)A(2)\ldots A(n)$.

Note that $p_n \neq p_m$ if $n \neq m$. Each program p_n is in P. Hence

$$\sum_{n \in \mathbb{N}} 2^{|U_H(p_n)|-|p_n|} = \sum_{n \in \mathbb{N}} 2^{n+1-H(A(0)A(1)A(2)\ldots A(n))} \leq c$$

which gives that $c/2$ is a finite upper bound for the sum in Condition 4. So Condition 4 is satisfied and Conditions 2 and 3 follow from Condition 4. To see Condition 5, consider for given g the martingale

$$\mathrm{mg}_g(\sigma) = \sum_{a : U_C(\eta_a) < |\eta_a| - g(|\eta_a|)} \mathrm{mg}_a(\sigma)$$

and note that $m_g(\sigma) \leq 2^{|\sigma|} \cdot \sum_{n \in \mathbb{N}} 2^{-g(n)}$ as for each length n there are at most $2^{n-g(n)}$ strings $\eta_a \in \{0,1\}^n$ with $C(\eta_a) < n - g(n)$ and the mg_a of these η_a contribute to

the sum $\mathrm{mg}_g(\sigma)$ not more than $2^{|\sigma|} \cdot 2^{-g(n)}$. Hence mg_g is a martingale. If there are infinitely many a such that A extends η_a and $C(\eta_a) < |\eta_a| - g(|\eta_a|)$ then for each such a the martingale mg_a is part of the sum contributing to mg_g and mg_g goes on A to ∞ as for each such a and almost all n, $\mathrm{mg}_a(A(0)A(1)A(2)\ldots A(n)) = 1$. This contradicts A being Martin-Löf random and Condition 5 holds.

For the other way round, assume that A is not Martin-Löf random and that RT is a randomness test covering A as defined in Exercise 11.5. Now let

$$E = \{\langle |\sigma| - e/2, \sigma\rangle : \langle e, \sigma\rangle \in RT\}$$

and note that

$$\sum_{\langle d,\sigma\rangle \in E} 2^{-d} \leq \sum_{\langle e,\sigma\rangle \in RT} 2^{e/2-|\sigma|} \leq \sum_{e\in\mathbb{N}} 2^{e/2-e} \leq \frac{1+\sqrt{2}}{2}$$

so that by Exercise 11.19 there is a constant d with

$$\forall \langle e, \sigma\rangle \in RT\, [H(\sigma) \leq |\sigma| + d - e/2].$$

As RT covers A one can conclude that for infinitely many numbers e there is a prefix of A of length $n+1$ and complexity below $n + 1 + d - e/2$; in other words,

$$\sup\{n - H(A(0)A(1)A(2)\ldots A(n)) : n \in \mathbb{N}\} = \infty$$

and thus Condition 2, Condition 3 and Condition 4 do not hold. It remains to show that Condition 5 fails as well. For this recall that $H(a)$ is recursively approximable from above with $H_s(a)$ being the s-th approximation as defined in the proof of Theorem 11.18. One chooses g of the following special form:

$$g(\langle a, s\rangle) = \begin{cases} H_s(a) & \text{if } H_s(a) < a \text{ and } \forall t < s\, [H_s(a) < H_t(a)]; \\ \langle a, s\rangle & \text{otherwise.} \end{cases}$$

Note that due to the case-distinction in the definition of g, one knows that either $g(\langle a, s\rangle) = \langle a, s\rangle$ or $g(\langle a, s\rangle) = H(a) + b$ where for each b there is at most one s with $g(\langle a, s\rangle) = H(a) + b$. It follows that

$$\sum_{n\in\mathbb{N}} 2^{-g(n)} \leq \sum_{n\in\mathbb{N}} 2^{-n} + \sum_{a,b\in\mathbb{N}} 2^{-H(a)-b} \leq 2 + \sum_{a\in\mathbb{N}} 2^{1-H(a)} \leq 4.$$

Furthermore there is a constant d' such that

$$C(A(0)A(1)A(2)\ldots A(\langle a, s\rangle)) \leq H(A(0)A(1)A(2)\ldots A(a)) + (\langle a, s\rangle - a) + d'$$

for all a, s. As seen before, there is for every e some a such that

$$H(A(0)A(1)A(2)\ldots A(a)) \leq a - e$$

and taking s to be the first value such that $H_s(a) = H(a)$ it follows that

$$\begin{aligned} C(A(0)A(1)A(2)\ldots A(\langle a,s\rangle)) &\leq a - e + (\langle a,s\rangle - a) + d' \\ &= \langle a,s\rangle + d' - e = \langle a,s\rangle + d' - g(\langle a,s\rangle). \end{aligned}$$

One can replace the function g by the function $g' : n \mapsto g(n) + d' + 1$ in order to get that

$$\exists^\infty n\, [C(A(0)A(1)A(2)\ldots A(n)) < n - g'(n)]$$

and use that $\sum_n 2^{-g'(n)} < \infty$ as well. This shows then that also Condition 5 fails. ∎

Exercise 11.22. Solovay [106] gave an alternative definition of a test. A *Solovay test* is an r.e. set E of strings such that $\sum_{\sigma\in E} 2^{-|\sigma|} < \infty$. E covers a set A iff there are infinitely many n such that $A(0)A(1)A(2)\ldots A(n) \in E$. Show that a set is Martin-Löf random iff it is not covered by any Solovay test.

Remark 11.23. Chaitin [14, 20] introduced the halting probability Ω which is the unique set of natural numbers such that

$$\sum_{n\in\Omega} 2^{-n-1} = \sum_{p\in\mathrm{dom}(U_H)} 2^{-|p|}.$$

The name halting-probability refers to the fact that one could by random draw a sequence B of bits and say that U_H halts on B iff B extends some string p in the domain of U_H. The probability that this happens is exactly the real number $\sum_{n\in\Omega} 2^{-n-1}$. Theorem 11.25 below shows that Ω is Martin-Löf random.

Exercise 11.24. One might ask whether there is an easier characterization of the Martin-Löf random sets using C. The most natural candidate would be to request that there is a constant c with $C(A(0)A(1)A(2)\ldots A(n)) \geq n - c$ for all n. Show that this characterization does not work out and that in fact no set A satisfies this property.

One might therefore look at the sets A for which there is a constant c such that $C(A(0)A(1)A(2)\ldots A(n)) \geq n - c$ for infinitely many n; such sets are called *Kolmogorov random*. Show that every Kolmogorov random set is Martin-Löf random and show that the converse direction does not hold as Ω is not Kolmogorov random. Note, but this is not part of this exercise, that a set is Kolmogorov random iff it is Martin-Löf random relative to the oracle $\mathbb{K}$ [71, 78].

Theorem 11.25. *Chaitin's Ω is Martin-Löf random.*

Proof. Let $A <_{lex} B$ mean that either $A = B$ or the minimum x of the set $\{y : A(y) \neq B(y)\}$ satisfies $A(x) < B(x)$, in other words, $x \in B - A$. Note that the set Ω is left-r.e., that is, that

$$\{\sigma : \sigma 0^\infty <_{lex} \Omega\}$$

is a recursively enumerable set. Fix a recursive one-one enumeration $p_0, p_1, p_2, \ldots$ of the domain of U_H. Now one can define for every s the approximation Ω_s to Ω such that

$$\sum_{m \in \Omega_s} 2^{-m-1} = 2^{-|p_0|} + 2^{-|p_1|} + 2^{-|p_2|} + \ldots + 2^{-|p_s|}.$$

This sequences approximates Ω "from the left", that is, $\Omega_0 <_{lex} \Omega_1 <_{lex} \Omega_2 <_{lex} \ldots <_{lex} \Omega$. Now one defines the following partial-recursive function V:

$V(p)$ simulates $U_H(p)$ until $U_H(p)$ halts with some output σ. Then $V(p)$ searches for the first s such that Ω_s extends the string σ. Then $V(p)$ determines the set

$$D = \{\tau : |\sigma| = |\tau| \wedge \forall m \leq s\, [|p_m| \geq |\sigma| \vee U_H(p_m) \neq \tau]\}.$$

If all these simulations and searches terminate then $V(p)$ takes as value the lexicographic least string in D else $V(p)$ is undefined.

Obviously V is a prefix-free partial-recursive function. Hence there is a constant d such that $H(V(p)) \leq |p| + d$ for all p in the domain of V.

Given n, let p be the shortest program such that $U_H(p) = \Omega(0)\Omega(1)\Omega(2)\ldots\Omega(n)$ and let s be the first stage such that $U_H(p) = \Omega_s(0)\Omega_s(1)\Omega_s(2)\ldots\Omega_s(n)$. This s is used in the construction of $V(p)$ and thus there is no $r \leq s$ with $|p_r| \leq n \wedge U_H(p_r) = V(p)$. Furthermore, there is no $r > s$ with $2^{-|p_r|} \geq 2^{-n}$ as otherwise Ω_s and Ω would differ on the domain $0, 1, 2, \ldots, n$; hence $|p_r| \geq n$ for all $r > s$. Therefore, $n \leq H(V(p)) \leq |p|+d$ and $H(\Omega(0)\Omega(1)\Omega(2)\ldots\Omega(n)) \geq n-d$. This constant d is independent of n and Condition 3 from Theorem 11.21 is satisfied. Hence Ω is Martin-Löf random. ∎

Remark 11.26. The Turing degree of Ω and $\mathbb{K}$ is the same and one can even get stronger reductions than Turing reducibility.

The direction $\Omega \leq_{tt} \mathbb{K}$ follows from the fact that Ω is left-r.e. and one knows $\Omega(n)$ iff one knows which of the strings $\{0,1\}^{n+1}$ are enumerated into the set $\{\sigma \in \{0,1\}^* : \sigma 0^\infty <_{lex} \Omega\}$. The other relation $\mathbb{K} \leq_{wtt} \Omega$ is implicit already in a modification previous proof, one just shows that there is a function V such that $V(0^e 1)$ takes some string σ of length $2e$ with $H_s(\sigma) \geq 2e$ at the stage s where e is enumerated into $\mathbb{K}$ and then one can conclude that for all sufficiently large e, $e \in \mathbb{K}_s$ iff $e \in \mathbb{K}$ for the least s with $\Omega(0)\Omega(1)\Omega(2)\ldots\Omega(3e) = \Omega_s(0)\Omega_s(1)\Omega_s(2)\ldots\Omega_s(3e)$.

So Ω is a natural example for a left-r.e. Martin-Löf random set. Kučera and

Slaman [55] showed that all such sets are the halting-probability of a universal prefix-free machine: so a left-r.e. set is Martin-Löf random iff there is a prefix-free machine V such that the prefix-free complexity based on V and the given complexity H do not differ more than some constant. In other words, every left-r.e. Martin-Löf random set is the halting probability of a machine which generates some legitimite variant of H.

Ω permits to produce natural examples for the Theorem of Kleene and Post: Let $\Omega_{ev} = \{x : 2x \in \Omega\}$ and $\Omega_{od} = \{x : 2x+1 \in \Omega\}$. Then $\Omega = \Omega_{ev} \oplus \Omega_{od}$ and by the Theorem of Van Lambalgen, Ω_{ev} is Martin-Löf random relative to Ω_{od} and vice versa. Hence Ω_{ev} and Ω_{od} are Turing incomparable and Turing reducible to Ω and $\mathbb{K}$.

Furthermore, the set $\{a : \eta_a 0^\infty <_{lex} \Omega\}$ is a natural example of an r.e. set which is wtt-complete but not tt-complete; it is tt-equivalent to Ω and the tt-incompleteness of Ω was found by Calude and Nies [15].

In the following, the notions of C and H are used to compare the amount of randomness or nonrandomness of sets. This is done by defining C-reducibility and H-reducibility as follows.

Definition 11.27. For given sets A and B, say that A is *C-reducible* to B (A is *H-reducible* to B) iff the following corresponding definition is satisfied:

$$A \leq_C B \;\Leftrightarrow\; \exists k\, \forall n\, [C(A(0)A(1)A(2)\ldots A(n)) \leq C(B(0)B(1)B(2)\ldots B(n)) + k];$$
$$A \leq_H B \;\Leftrightarrow\; \exists k\, \forall n\, [H(A(0)A(1)A(2)\ldots A(n)) \leq H(B(0)B(1)B(2)\ldots B(n)) + k].$$

Furthermore a set is *C-trivial* iff it is C-reducible to every set and *H-trivial* iff it is H-reducible to every set.

Remark 11.28. Every C-trivial set is recursive. Furthermore, for every sets A, B, if $\{2^{2^n} : n \in A\} \leq_C \{2^{2^n} : n \in B\}$ then $A \leq_T B$. One can also show that there are minimal C-degrees, pairs of r.e. sets which form minimal pairs in the C-degrees as well as in the Turing degrees and C-degrees which coincides with Turing degrees. On the other hand, the Turing degree of $\mathbb{K}$ contains infinitely many C-degrees. And some C-degrees intersect with uncountably many Turing degrees although they do not contain any of these degrees completely, an example is

$$\{A : \exists c\, \forall n\, [\tfrac{n-c}{2} \leq C(A(0)A(1)A(2)\ldots A(n)) \leq \tfrac{n+c}{2}]\}$$

which is the C-degree of all sets where the C-complexity of A grows approximately with the factor of $\frac{1}{2}$.

The situation of H-reducibility and H-degrees is more complex as the following results show.

Theorem 11.29. *There is an r.e. H-trivial set which is not recursive.*

Proof. The set A is the range of a partial-recursive function ψ. A and ψ are build in stages with ψ_0 being undefined everywhere and $A_s = \{\psi_s(e) : e \leq s \wedge \psi_s(e)\downarrow\}$. For the construction, let $H_s(y)$ be an approximation from above to H. In stage $s+1$, the definition of ψ_s is updated as follows and three Kraft-Chaitin sets E, F, G are enumerated in parallel to ψ, A in order to make sure that H will be H-trivial. E_0, F_0, G_0 are all the empty set.

- For all $x \leq s$ define the weight $w(x,s)$ of x at s using the formula

$$w(x,s) = \sum_{y=x,x+1,\ldots,s} 2^{-H_{s+1}(y)}.$$

- For all $e \leq s$ with $\psi_s(e)\uparrow$ check whether there is an $x \in W_{e,s}$ such that

$$x > 2e \wedge w(x,s) < 2^{-e}.$$

  If so then $\psi_{s+1}(e)$ is the least such x else $\psi_{s+1}(e)$ remains undefined.

- Update the sets A, E, F, G as follows:

$$\begin{aligned}
A_{s+1} &= \{\psi_{s+1}(e) : e \leq s \wedge \psi_{s+1}(e)\downarrow\};\\
E_{s+1} &= E_s \cup \{\langle H_{s+1}(x), A_{s+1}(0)A_{s+1}(1)A_{s+1}(2)\ldots A_{s+1}(x)\rangle :\\
&\qquad x < s \wedge \exists y \leq x\,[y \in A_{s+1} - A_s]\};\\
F_{s+1} &= F_s \cup \{\langle H_{s+1}(x), A_{s+1}(0)A_{s+1}(1)A_{s+1}(2)\ldots A_{s+1}(x)\rangle :\\
&\qquad x < s \wedge H_{s+1}(x) < H_s(x)\};\\
G_{s+1} &= G_s \cup \{\langle H_{s+1}(s), A_{s+1}(0)A_{s+1}(1)A_{s+1}(2)\ldots A_{s+1}(s)\rangle\}.
\end{aligned}$$

Now it is verified that A is a simple and H-trivial set.

First, A is simple. Assume that W_e is infinite. As the sum $\sum_{x\in\mathbb{N}} 2^{-H(x)}$ converges, there is an z such that $\sum_{x\geq z} 2^{-H(x)} < 2^{-e}$ and there is an $x \geq z + 2e$ and a stage s with $x \in W_{e,s}$. Now

$$w(x,s) = \sum_{y=x,x+1,\ldots,s} 2^{-H_{s+1}(y)} < 2^{-e}$$

and $\psi_{s+1}(e)$ is defined. So $\psi(e)\downarrow \in W_e \cap A$ for all indices e of infinite r.e. sets W_e. The set A is the range of the partial-recursive function ψ and recursively enumerable. Furthermore, $A \cap \{0,1,2,\ldots,2d\} \subseteq \{\psi(e) : e \leq d \wedge \psi(e)\downarrow\}$ for all d, hence A has d non-elements below $2d$ and is co-infinite. It follows that A is simple.

Second, E is a Kraft-Chaitin set. Consider any s and any pair $\langle d, a_0a_1\ldots a_n\rangle \in$

$E_{s+1} - E_s$. Then there is an index e such that $\psi_{s+1}(e)$ but not $\psi_s(e)$ is defined, $\psi_{s+1}(e) \leq n \leq s$, $A_{s+1}(m) = a_m$ for $m = 0, 1, 2, \ldots, n$ and $d = H_{s+1}(a_0 a_1 a_2 \ldots a_n)$. It follows that $2^{-H_{s+1}(n)}$ is part of the sum $w(e, s)$ and that the pair $\langle d, a_0 a_1 \ldots a_n \rangle$ is the only pair to which this part of the sum is linked. Hence

$$\sum_{\langle d,\sigma\rangle \in E} 2^{-d} \leq \sum_{\langle e,s\rangle : \psi_{s+1}(e)\downarrow \wedge \psi_s(e)\uparrow} w(\psi_{s+1}(e), s) \leq \sum_{e \in \mathbb{N}} 2^{-e} \leq 2.$$

As the Kraft-Chaitin sum is finite and E r.e., E is a Kraft-Chaitin set.

Third, F is a Kraft-Chaitin set. Every pair going into F is of the form $\langle H_{s+1}(x), \sigma \rangle$ for some σ and each time such a pair goes in, $H_{s+1}(x) < H_s(x)$. So there is for each x and each $d \geq H(x)$ at most one s and σ with $H_{s+1}(x) = d$, $|\sigma| = x + 1$ and $\langle d, \sigma \rangle$ going into F at stage $s + 1$. It follows that

$$\sum_{\langle d,\sigma\rangle \in F} 2^{-d} \leq \sum_{x \in \mathbb{N}} \sum_{d \geq H(x)} 2^{-d} \leq \sum_{x \in \mathbb{N}} 2^{1-H(x)} \leq 2.$$

As the Kraft-Chaitin sum is finite and F r.e., F is a Kraft-Chaitin set.

Fourth, G is a Kraft-Chaitin set. This follows from the fact as in each stage exactly one pair of the form $\langle H_{s+1}(s), \sigma \rangle$ is put into G and $\sum_{s \in \mathbb{N}} 2^{-H_{s+1}(s)} \leq \sum_{s \in \mathbb{N}} 2^{-H(s)} \leq 1$. As the Kraft-Chaitin sum is finite and G is r.e., G is a Kraft-Chaitin set.

Fifth, $\langle H(x), A(0)A(1)A(2)\ldots A(x) \rangle \in E \cup F \cup G$ for all x. Now let a number x be given and let $\sigma = A(0)A(1)A(2)\ldots A(x)$. Let s be the first stage such that A_{s+1} extends σ and let t be the first stage such that $H_{t+1}(x) = H(x)$. If $x < s$ and $t \leq s$ then $\langle H(x), \sigma \rangle \in E_{s+1}$. If $x < t$ and $s \leq t$ then $\langle H(x), \sigma \rangle \in F_{t+1}$. If $x \geq s$ and $x \geq t$ then $\langle H(x), \sigma \rangle \in G_{x+1}$. Hence $\langle x, \sigma \rangle \in E \cup F \cup G$ as desired.

Sixth, A is H-trivial. As E, F, G are Kraft-Chaitin sets, so is their union $E \cup F \cup G$. Hence there is a constant c with $H(\sigma) \leq c + e$ whenever $\langle e, \sigma \rangle \in E \cup F \cup G$. For every x, $\langle H(x), A(0)A(1)A(2)\ldots A(x) \rangle \in E \cup F \cup G$ and $H(A(0)A(1)A(2)\ldots A(x)) \leq H(x) + c$. Hence A is H-trivial. ∎

Comprehensive Exercise 11.30. Modify above construction such that a set A is produced such that there is a constant c with $H(\sigma) \leq H^A(\sigma) + c$ for all σ. In other words, the prefix-free Kolmogorov complexity relative to H is up to the constant c the same as the unrelativized one. The idea is construct a prefix-free partial B-recursive function U_H^B such that U_H^B is universal for the prefix-free Kolmogorov complexity for any oracle B. Then A is constructed as above with two adaptations:

- The weight $w(x, s)$ is the sum over all $2^{-|p|}$ such that $U_s^{A_s}(p)\downarrow \wedge U_x^{A_s}(p)\uparrow$ where it is assumed that the computation of $U_z^B(p)$ queries the oracle B only at places $y < z$.

- Instead of pairs of the form $\langle d, A_{s+1}(0)A_{s+1}(1)A_{s+1}(2)\ldots A_{s+1}(x)\rangle$, the Kraft-Chaitin sets E, F, G contain now pairs of the form $\langle |p|, U_{H,s+1}^{A_{s+1}}(p)\rangle$ whenever the second component is defined. The goal of this construction is to get $\langle H^A(\sigma), \sigma\rangle$ into $E \cup F \cup G$ for all strings σ and then to apply the Kraft-Chaitin theorem.

This construction was first carried out by Muchnik.

Nies [77] showed that the class of H-trivial sets is a natural class where many notions coincide. For a proof, please see at the paper of Nies [77] and the references cited in it.

Theorem 11.31: Nies' Characterization of H-Trivial Sets [77]. *The following conditions are equivalent for any set A.*

- *A is H-trivial, that is, $\exists c \forall n\,[H(A(0)A(1)A(2)\ldots A(n)) \leq H(n) + c]$;*
- *A is low for Martin-Löf random, that is, $\forall R\,[R$ is Martin-Löf random relative to A iff R is Martin-Löf random (unrelativized)$]$;*
- *A is low for prefix-free Kolmogorov complexity, that is, $\exists c \forall \sigma \in \{0,1\}^*\,[H(\sigma) \leq H^A(\sigma) + c]$;*
- *A is a basis for Martin-Löf randomness, that is, $\exists R \geq_T A\,[R$ is Martin-Löf random relative to $A]$;*
- *$A \leq_T \mathbb{K}$ and A is low for Ω, that is, Ω is Martin-Löf random relative to A;*
- *$\exists B\,[A \leq_T B$ and B is r.e. and B is low for $\Omega]$.*

The last characterizations implies that every H-trivial set is Turing reducible to an r.e. H-trivial set.

Remark 11.32. Miller showed a characterization for "low is Ω" which is a weak counterpart of being "low for prefix-free Kolmogorov complexity". He showed that a set A is low for Ω iff $\exists c\, \exists^\infty \sigma \in \{0,1\}^*\,[H(\sigma) \leq H^A(\sigma) + c]$. Furthermore, he showed that every infinite recursive binary tree T has an infinite branch which is low for Ω; thus the sets which are low for Ω satisfy some basis theorem, similar to the low sets and the sets of hyperimmune-free Turing degree. Another important result is that every set which is low for Ω is either recursive or of hyperimmune Turing degree.

12 Inductive inference

Inductive inference is a general framework of learning. Learning means that a learner has to identify a given set within the a class of possible choices. For example, if a child grows up, it observes more and more sentences from the parent's languages and finds eventually a concept, that is a grammar, of all syntactically correct sentences in this language. As the child has no preknowledge of the man-made language, the only thing the child might use is that the language must follow some rules which could be formalized in a certain way which is normally called a grammar. In inductive inference, this concept had been formalized within Chomsky's theory of formal languages. So the general task is the following, where "learner" is used instead of "child".

Learning Activity. The learner reads more and more data from an unknown language and modifies its hypotheses from time to time. The learner might also interact with additional sources of information like oracles and teachers.

Texts. In inductive inference, one represents the language L mostly as the set $\{a \in \mathbb{N} : \eta_a \in L\}$ and therefore one works with sets of numbers. A text is then an enumeration of a set L and one permits besides the numbers in the text also the pause symbol # for the case that no (new) data is available. Texts need not to be recursive, but every r.e. set has a primitive-recursive text.

Classes. The learner is not required to learn all possible recursively enumerable languages but has only to deal with those from a reasonable class S. Indeed, this phenomenon is already found in the real world: In Chinese or Japanese, the writing system is so difficult that it takes up to the age of 12 or 15 (various people asked gave quite different ages) until a child can read a newspaper. In English speaking countries this age might be 8. So some languages are more and other less difficult although languages a human cannot learn in lifetime just do not exist. So the child has only to deal with a small choice among all theoretically possible languages; S denotes this class. One of the major topics is "which classes S are learnable" and — as there are various formalizations of learning — whether there is a class S learnable under formalization (A) but not under the formalization (B). Often it is convenient to have classes which are described in a uniform way. There are three types of such classes:

- Uniformly recursive classes. The class is a list $L_0, L_1, \ldots$ of sets such that $\{\langle e, x\rangle : x \in L_e\}$ is recursive.
- Uniformly recursively enumerable classes. The class is a list $L_0, L_1, \ldots$ of sets such that $\{\langle e, x\rangle : x \in L_e\}$ is recursively enumerable.

- General classes. The class is a list $L_0, L_1, \ldots$ of sets such that each L_e is recursively enumerable.

The listing $L_0, L_1, \ldots$ of the sets in the class is normally not viewed upon as a part of the class, one only asks whether there is such a listing. If it exists, it might be exploited by the learner as one asks normally only whether a learner exists, not how to construct the learner.

Learners. Learners are recursive functions from $(\mathbb{N} \cup \{\#\})^*$ to $\mathbb{N}$. It is assumed that all data and all hypotheses are coded as natural numbers. A learner can be partial, but for most purposes it is possible to extend the learner such that it is total.

In certain settings, learners are augmented with an oracle [1]. Indeed, a lot of literature is dealing with the question how much additional learning power certain oracles give. That an oracle can be helpful can be seen from the fact, that it can be used to check whether the data seen so far is contained in a set W_e for some hypothesis e or not. So a learner with access to the halting-problem as an oracle can make sure that all its hypotheses are consistent with the data seen so far.

The Major Learning Criteria. A learning criterion refers to the sequence of hypothesis output by the learner in relation to the input seen so far. Here is a choice of some criteria, the literature knows much more. The first one is the most common one.

- Learning in the Limit (Lim) or Explanatory learning (Ex): The learner outputs only finitely often a new hypothesis and the last one is correct [39].

- Behaviourally Correct learning (BC): The learner outputs an infinite sequence $e_0, e_1, \ldots$ of hypotheses such that W_{e_n} equals to the set to be learnt for almost all n [6, 18].

- Finite learning (Fin): The learner outputs only one hypothesis and this one is correct; the learner outputs a special symbol "?" before it conjectures the correct hypothesis which then is never revised [7].

- Confident learning: This is learning combined with the constraint that the learner converges syntactically on every text [82]; so a confident Ex-learner converges syntactically on all texts and a confident BC-learner converges semantically on all texts; although, on some texts for sets outside the class to be learnt, these learners have to converge to a wrong hypothesis. Reasons for that are: (a) there are uncountably many sets while there are only countably many indices; (b) the class of all r.e. languages is not learnable.

- Consistent learning: This is learning in the limit combined with the additional constraint that whenever the learner outputs a hypothesis e after seeing data σ, every number x occurring in σ is covered by the hypothesis: $x \in W_e$ [10].
- Conservative learning: This is learning in the limit combined with the additional constraint that whenever the learner outputs at some time i and later j then it has seen some number x prior to conjecturing j with $x \in W_j - W_i$ [2].

The next results present the basics of inductive inference and give some overview on the main connections between the above mentioned criteria.

The class of all r.e. languages is not learnable in the limit, it is even not behaviourally correctly learnable. The next example gives some natural classes of sets which are learnable.

Example 12.1. *The class* $\{W_e : W_e \neq \emptyset \wedge e = \min(W_e)\}$ *of self-describing languages is not empty and Ex-learnable.*

Proof. There is a recursive function f such that $W_{f(e)} = \{x \geq e : x = e \vee x \in W_e\}$. By the Fixed-Point Theorem there are infinitely many e such that $W_e = W_{f(e)}$; hence the class of self-describing languages is not empty but infinite.

A learner which outputs the minimal number seen so far (and 0 if no number has been seen so far) is a learner which converges on every text of a self-describing language L to $\min(L)$; by choice of the class this minimum is an index for the language and hence this algorithm learns the class. ∎

An important property of learners is the following locking sequence criterion from Blum and Blum [10].

Theorem 12.2: Existence of Locking Sequences [10]**.** *Let M be a recursive learner and L be an r.e. set learned by M. Then there is $\sigma \in (L \cup \{\#\})^*$ such that*

$$W_{M(\sigma)} = L \wedge \forall \tau \in (L \cup \{\#\})^* [M(\sigma\tau) = M(\sigma)].$$

This σ is called a locking sequence *for L.*

Proof. Let $a_0, a_1, a_2, \ldots$ be a recursive enumeration of L. Now assume by way of contradiction that every string of elements from $L \cup \{\#\}^*$ has an extension $\sigma\tau$ with $\tau \in (L \cup \{\#\})^*$ with $M(\sigma\tau) \neq M(\sigma)$. Now one can build inductively a recursive text

$$T = a_0\tau_0 a_1\tau_1 a_2\tau_2 a_3\tau_3 \ldots$$

such that τ_n is the first string in $(L \cup \{\#\})^*$ found by exhaustive search such that

$$M(a_0\tau_0 a_1\tau_1 a_2\tau_2 \ldots a_n\tau_n) \neq M(a_0\tau_0 a_1\tau_1 a_2\tau_2 \ldots a_n).$$

But then M would diverge on T and thus not Ex-learn L, even not from recursive texts. Hence there is a sequence $\sigma = a_0\tau_0 a_1\tau_1 a_2\tau_2 \ldots a_n$ such that the corresponding extension τ_n cannot be found. It follows that $M(\sigma a_{n+1}a_{n+2}\ldots a_{n+m}) = M(\sigma)$ for all m. As M Ex-learns L one has that $W_{M(\sigma)} = L$. Thus σ is a locking sequence. ∎

The result does show even something stronger: for every recursive learner M and every r.e. set L there is either a sequence $\sigma \in (L \cup \{\#\})^*$ with $M(\sigma\tau) = M(\sigma)$ for all $\tau \in (L \cup \{\#\})^*$ or there is a recursive text on which M makes infinitely many mind changes. Although one can show that there is an equivalent of locking-sequences for BC-learning and a corresponding result, one can no longer search for these mind changes and therefore the texts on which a BC-learner converges might be nonrecursive. Indeed there are unlearnable classes such that some BC-learner still learns them from all recursive texts; this explains the necessity to consider nonrecursive texts in the case of BC-learning.

Based on the concept of locking-sequences, Angluin [2] was able to give a criterion when a uniformly recursive class is learnable from positive data.

Theorem 12.3: Angluin's Tell-Tale-Sets [2]**.** Let $\{L_0, L_1, L_2, \ldots\}$ be a uniformly recursive class. This class is then learnable in the limit iff there is uniformly recursively enumerable list $H_0, H_1, H_2, \ldots$ of finite sets such that for every i, j,

- $H_i \subseteq L_i$;
- if $H_i \subseteq L_j \subseteq L_i$ then $L_j = L_i$.

Proof. If M is an explantory learner for the uniformly recursive class $\{L_0, L_1, L_2, \ldots\}$ then one can enumerate for each e all strings $\sigma_0, \sigma_1, \sigma_2, \ldots \in (L_e \cup \{\#\})^*$ and let

$$H_e = \{x : \exists y\, \forall z < y\, \exists \tau \in (L_e \cup \{\#\})^*\, [x \text{ occurs in } \sigma_y \wedge M(\sigma_z\tau) \neq M(\sigma_z)]\}.$$

It is easy to see that the sets H_e are uniformly recursively enumerable. Furthermore, as for H_e some string σ_y in the list of all strings over $(L_e \cup \{\#\})^*$ is a locking-sequence and then only the finitely many x occurring in any of the strings $\sigma_0, \sigma_1, \ldots, \sigma_y$ can show up in H_e. So Angluin's criterion is necessary even if the underlying class in only uniformly recursively enumerable.

Now consider that such a family $H_0, H_1, H_2, \ldots$ is given and that the underlying class is uniformly recursive. The learner N for the class outputs on input $a_0, a_1, a_2, \ldots, a_n$ the least $d \leq n$ such that $i = n$ or

$$H_{d,n} \subseteq \{a_0, a_1, \ldots, a_n\} \subseteq L_d \cup \{\#\}.$$

Clearly N is total. Assume now that N has to learn L_e with e being the minimal index of the set to be learnt; that is, there is no $d < e$ with $L_d = L_e$. Given a text $a_0\, a_1\, a_2\, \ldots$ for L_e there is an n so large that $H_{d,n} = H_d$ for all $d \leq e$ and $\{a_0, a_1, a_2, \ldots, a_n\} \not\subseteq L_d \cup \{\#\}$ for all $d < e$ where $L_e \not\subseteq L_d$. Note that for those $d < e$ with $L_e \subset L_d$ it holds that $H_d = H_{d,n} \not\subseteq L_e$ and hence $H_{d,n} \not\subseteq \{a_0, a_1, a_2, \ldots, a_m\}$ for all m. Thus, for all $m \geq n$, e is the first index such that

$$H_{e,n} \subseteq \{a_0, a_1, \ldots, a_m\} \subseteq L_e \cup \{\#\}.$$

Hence $N(a_0\, a_1\, a_2\, \ldots\, a_m) = e$ for all $m \geq n$. It follows that N learns L_e and N is an Ex-learner for $\{L_0, L_1, L_2, \ldots\}$. ∎

Examples 12.4 [2, 39]**.** Angluin's criterion permits to decide the learnability of the following uniformly recursive classes.

- The class of all finite sets plus one infinite set L is unlearnable as any finite tell-tale-subset H of L has to be learned itself as well.
- The class of all cofinite sets is unlearnable as if H is the finite tell-tale-subset of $\mathbb{N}$ then there is a cofinite L with $H \subset L \subset \mathbb{N}$; so Angluin's criterion cannot be satisfied.
- Every finite class S of languages is learnable as one can consider for every $L \in S$ the tell-tale-set $H_L = \{\min(L - L') : L' \in S \wedge L \not\subseteq L'\}$ and the learner outputs always an index for that L where H_L is contained in the data seen so far and for which there is no $L' \supset L$ such that $H_{L'}$ is contained in the data seen so far as well. As there are only finitely many sets involved, all sets H_L are finite and this learning algorithm can even be carried out of some of these sets are not recursive.
- The (infinite) class of all finite languages is learnable. This is witnessed by taking $H_e = L_e$ for every e-th member of an underlying listing $L_0, L_1, L_2, \ldots$ of all finite sets.
- The class of all graphs of primitive-recursive functions is learnable. There is a uniformly recursive one-one numbering $L_0, L_1, L_2, \ldots$ of these graphs and as $L_i \not\subseteq L_j$ for any distinct i, j one can take $H_e = \emptyset$ for all e.

A further result is that the class REC of all graphs of recursive functions is not Ex-learnable. This result is not obtained by applying Angluin's theorem; instead one considers any given recursive learner. The learner has to learn all functions which almsot everywhere 0 and hence one can find an infinite recursive sequence $\sigma_0, \sigma_1, \sigma_2, \ldots$ of functions such that

- $\sigma_n \in \mathbb{N}^*$;
- σ_{n+1} extends $\sigma_n 1$;
- M conjectures after seeing the data for σ_n that the next value is 0 and not 1 and is thus not consistent with the extension σ_{n+1} of σ_n.

Then the graph of the recursive function $f = \lim_n \sigma_n$ is not learned by M.

Exercise 12.5. Which of the following classes is Ex-learnable?

- $S_1 = \{\mathbb{K} \cup \{x\} : x \notin \mathbb{K}\}$;
- $S_2 = \{\mathbb{N} - \{x, y\} : x, y \in \mathbb{N}\}$;
- $S_3 = \{L : \forall i, j \in L\, [i = j \vee W_i = L \vee W_j = L]\}$;
- $S_4 = \{D \times \mathbb{N} : D \text{ is finite}\}$.

The next theorem gives an example how to separate the learning criteria Ex and BC.

Example 12.6. Let A be a nonrecursive but r.e. set. Then the class $\{A \cup \{x\} : x \in \mathbb{N}\}$ is BC-learnable but not Ex-learnable from text.

Proof. It is easy to make a BC-learner for the class. This BC-learner can always update the hypothesis and thus code the input into the output. That is, the BC-learner M is given by the equation

$$W_{M(a_0\, a_1\, a_2\, \ldots\, a_n)} = A \cup \{a_0, a_1, a_2, \ldots, a_n\} - \{\#\}.$$

If $a_0\, a_1\, a_2\, \ldots$ is a text for $A \cup \{x\}$, then there is an m with $a_m = x$ and $W_{M(a_0\, a_1\, a_2\, \ldots\, a_n)} = A \cup \{x\}$ for all $n \geq m$. Hence M BC-learns the class.

Note that $L_x = A \cup \{x\}$ would be a recursively enumerable indexing of the class to be learnt; but a uniformly recursive indexing cannot exist as A is not recursive.

Assume now that M is an explanatory learner which at least learns A. Then there is a locking sequence σ for A and a recursive text $a_0\, a_1\, a_2\, \ldots$ for A. Now consider the set

$$B = \{x : \exists n\, [M(\sigma\, x\, a_0\, a_1\, a_2\, \ldots\, a_n) \neq M(\sigma)]\}$$

which is r.e. and disjoint to A. As A is not recursive, B cannot be the complement of A; hence there is an $x \notin A \cup B$ such that $M(\sigma\, x\, a_0\, a_1\, a_2\, \ldots\, a_n) = M(\sigma)$ for all n; that is, M converges on a text for $A \cup \{x\}$ to an index for A and does not learn $A \cup \{x\}$. It follows that the given class is not Ex-learnable. ∎

Note that one can extend the previous result to show that the class $\{A \cup D : D \text{ is finite}\}$ is also BC-learnable but not Ex-learnable. Now a further natural example for this separation is given.

Example 12.7. *The classes* $\{L_0, L_1, L_2, \ldots\}$ *and* $\{H_0, H_1, H_2, \ldots\}$ *given by*

$$L_e = \{e + x : x \leq |W_e|\} \text{ and } H_e = \{e + x : x \in \mathbb{N}\}$$

are both Ex-learnable but their union $\{L_0, H_0, L_1, H_1, L_2, H_2, \ldots\}$ *is only BC-learnable and not Ex-learnable.*

Proof. The classes $\{L_0, L_1, L_2, \ldots\}$ and $\{H_0, H_1, H_2, \ldots\}$ can both be Ex-learned by finding in the limit the smallest element e in the input and outputting a canonical index of the corresponding set L_e or H_e, respectively. Furthermore, one can BC-learn $\{L_0, H_0, L_1, H_1, L_2, H_2, \ldots\}$ as follows: On input with minimum e, maximum $e + a$ and length s one checks if $|W_{e,s}| \leq a$. If so then one outputs L_e else one outputs H_e. If $L_e \neq H_e$ then it will turn out in the limit whether the text is for L_e or for H_e and the convergence will even be syntactic. If $L_e = H_e$ then the learner might on a text for the set alternate infinitely often between an index for H_e and for L_e.

Now assume by way of contradiction that there is a recursive Ex-learner for the class $\{L_0, H_0, L_1, H_1, L_2, H_2, \ldots\}$. By the necessity of Angluin's criterion in the case of uniformly r.e. classes, one can enumerate uniformly in e finite subsets $O_e \subseteq H_e$ such that $O_e \not\subseteq L_e$ whenever $L_e \neq H_e$.

So, whenever $|W_e| = \infty$ then $L_e = H_e$ and $\max(O_e) \leq |W_e|$ and whenever $|W_e| < \infty$ then $L_e \neq H_e$, $\max(O_e) \notin L_e$ and $\max(O_e) > |W_e|$. Hence one can decide in the limit whether W_e is infinite: first one finds in the limit the value $\max(O_e)$ and then one checks in the limit whether W_e has at least $\max(O_e)$ many elements. But it is well-known that this cannot be done as the set $\{e : |W_e| = \infty\}$ has the same Turing degree as $\mathbb{K}'$. From this contradiction one knows that the class $\{L_0, H_0, L_1, H_1, L_2, H_2, \ldots\}$ is not Ex-learnable. ∎

Exercise 12.8: Vacillatory Learning [17]**.** A class is vacillatorily learnable iff there is a BC-learner M for the class which outputs on every text for every language in the class only finitely many different indices. So the class $\{L_0, H_0, L_1, H_1, L_2, H_2, \ldots\}$ from Example 12.7 is vacillatorily learnable as the learner finds in the limit the correct parameter e and then either converges to an index for L_e or H_e or vacillates infinitely often between the two canonical indices for L_e and H_e.

Show that a class S is vacillatorily learnable iff there exists a learner which converges for every $L \in S$ and every text T for L to a hypothesis d such that there is an $e \leq d$ with $W_e = L$.

Show that for every r.e. but not recursive set A the class $\{A \cup \{x\} : x \in \mathbb{N}\}$ is

behaviourally correct but not vacillatorily learnable. Show that the class

$$S = \{L : L \neq \emptyset \Rightarrow \exists e < \min(L)\, \forall x > \min(L)\, [L(x) = W_e(x)]\}$$

is vacillatorily learnable but not explanatorily learnable. For the negative result, one can define a recursive function f such that $f(e) > e + i + j$ where i is an index for L_e and j an index for H_e from the classes in Example 12.7. Having this, one shows that $L_e - \{x : x < f(e)\}$ and $H_e - \{x : x < f(e)\}$ are both in S. Then one shows that if S would be Ex-learnable then $\{L_0, H_0, L_1, H_1, L_2, H_2, \ldots\}$ would be Ex-learnable as well in contradiction to the Example 12.7.

Learning in the limit is more powerful than finite learning since the class of all finite sets is learnable in the limit but not finitely learnable. This class is even conservative Ex-learnable but the next example shows that not every class which is learnable in the limit is also conservatively Ex-learnable.

Exercise 12.9 [82]**.** Show that the class of all finite sets is not confidently Ex-learnable.

Exercise 12.10 [2]**.** Show that the class of all sets $\{e, e+1, e+2, \ldots\}$ with $e \in \mathbb{N}$ and $\{e, e+1, e+2, \ldots, e+d\}$ with $e \in \mathbb{K} - \mathbb{K}_d$ is consistently and confidently Ex-learnable but not conservatively BC-learnable.

Exercise 12.11 [10]**.** Show that the class $\{\mathbb{K}, \mathbb{N}\}$ is conservatively and confidently Ex-learnable but not consistently Ex-learnable.

Remark 12.12. There are unions of Ex-learnable classes which are not BC-learnable: for example the union of the class of all finite sets and the class $\{\mathbb{N}\}$ [39]. Also, there are confidently Ex-learnable classes such that their union is not confidently Ex-learnable; an example are the classes $\{L_0, L_1, L_2, \ldots\}$ and $\{H_0, H_1, H_2, \ldots\}$ from Example 12.7.

Nevertheless, one can show that the union of two finitely many confidently vacillatorily learnable classes is again confidently vacillatorily learnable: Assume that $M_1, M_2, \ldots, M_k$ converge on all texts and that there are classes $S_1, S_2, \ldots, S_k$ such that each learner M_h converges on every text T for a language $L \in S_h$ to a number d such that there is $e \leq d$ with $W_e = L$. Then the learner N given by the equation $N(\sigma) = M_1(\sigma) + M_2(\sigma) + \ldots + M_k(\sigma)$ has the same property for $S_1 \cup S_2 \cup \ldots \cup S_k$.

Confident behaviourally correct learning is also closed under union: if the learners $M_1, M_2, \ldots, M_k$ are fed with a text T, the learner N can switch between them and outputs at each stage that one it is currently tracing. It makes its n-th switch from the learner M_i to a learner M_j at some stage s iff there is an m such that

- $n < m < s$;
- $W_{M_i(\tau),s} \cap \{0,1,2,\ldots,m\} \neq \{T(0),T(1),T(2),\ldots,T(m)\} \cap \{0,1,2,\ldots,m\} - \{\#\}$;
- $W_{M_j(\tau),s} \cap \{0,1,2,\ldots,m\} = \{T(0),T(1),T(2),\ldots,T(m)\} \cap \{0,1,2,\ldots,m\} - \{\#\}$.

One can verify that the learner always semantically converges to the correct hypothesis if at least one of the machines $M_1, M_2, \ldots, M_k$ does so; furthermore, in the case that all machines fail to identify the correct language, there are only finitely many switches as from some time on the values of n and s are so large that

$$W_{M_j(\tau),s} \cap \{0,1,2,\ldots,m\} = \{T(0),T(1),T(2),\ldots,T(m)\} \cap \{0,1,2,\ldots,m\}$$

is no longer true when N thinks about changing from i to j.

Remark 12.13. For BC-learning, consistency is not restrictive. The reason is that one can modify a given BC-learner M to a new BC-learner N such that

$$W_{N(a_0a_1a_2\ldots a_n)} = W_{M(a_0a_1a_2\ldots a_n)} \cup \{a_0,a_1,a_2,\ldots,a_n\} - \{\#\}.$$

This is always possible as the learner needs only to converge semantically, not syntactically. Hence, every two subsequent indices of N will be different. But as long as the hypotheses of M are correct, so are those of N as N adds only observed elements to these hypotheses. Furthermore, N is obviously consistent. Hence, whenever M BC-learns a set L on a text T for L, then N consistently BC-learns L on T.

Alternative Models of Data Presentation. Besides text, there are also other sources of data which might be considered for learning.

An *informant* supplies the sequence $L(0), L(1), L(2), \ldots$ informing over the characteristic function of L instead of a text for L.

Queries permit to the learner to ask questions to a teacher which the teacher has to answer. The queries might be used to retrieve information going much beyond what can be observed; for example equivalence queries answer whether a given hypothesis is equivalent to the concept to be learnt. Queries to teachers are in particular used to study the complexity of learning in settings where there is potentially exponentially many data-items but only polynomial time to learn is permitted.

One can also consider *learning with additional information*. In this scenario, the learner recieves, besides a text, also an upper bound on the size of some index of the set to be learnt. Such additional information increases the learning-performance quite

a bit [32] although the class of all r.e. sets remains unlearnable in this setting [42].

An *oracle* can be used in addition to a text or informant. Note that an oracle is independent on the language L to be learned, but an orcale might nevertheless give information over the class S from which L is taken or permit to decide which r.e. sets contain the data seen so far and which not.

Remark 12.14: Oracles [1, 31, 59]**.** It has been investigated to which extent oracles are helpful during the learning process. That is, one considers instead of a recursive learner M an A-recursive learner M^A.

Such oracles can overcome the difference between explanatory and behaviourally correct learning: Every BC[A]-learnable class is Ex[$A \oplus \mathbb{K}$]-learnable and the class $\{\mathbb{K} \cup \{x\} : x \in \mathbb{N}\}$ needs an oracle above $\mathbb{K}$, that is, this class is Ex[B]-learnable iff $B \geq_T \mathbb{K}$.

Jain and Sharma [43] showed that for every set A there is a set $B >_T A$ such that some class is learnable Ex[B]-learnable but not BC[A]-learnable. Adleman and Blum [1] showed that a the class REC of all graphs of recursive functions is Ex[A]-learnable iff A is high; Kummer and Stephan [31, 59] showed that REC is already BC[B]-learnable for some low oracle B; so the characterization of the oracles permitting to learn REC does not carry over from Ex to BC.

Remark 12.15: Teachers and Learners [3]**.** Queries permit to the learner to ask questions to a teacher which the teacher has to answer and which relate to the concept to be learnt. In general, the answers to the queries provide more information about the concept to be learnt than the observation of texts. There are always two modes: One is just giving "Yes" or "No" but the second is justifying a "No" by a counterexample. So when the learner asks "Is L the set of all even numbers?" and the teacher answers "No" then the teacher also supplies some odd number contained in L back to the learner. Queries have always to be from query languages and the following languages are common:

- Equivalence queries: "Is $W_e = L$?";
- Subset queries: "Is $W_e \subseteq L$?";
- Superset queries: "Is $W_e \supseteq L$?";
- Disjointness queries: "Is $W_e \cap L = \emptyset$?";
- Membership queries: "Is $a \in W_e$?" for some a.

Membership queries are rare in inductive inference since one can either derive them (in the limit) from the text or one can simulate them by subset queries, superset queries or disjointness queries, the following is equivalent to the question whether $a \in W_e$:

- "Is $\{a\} \subseteq L$?";
- "Is $L \subseteq \mathbb{N} - \{a\}$?";
- "Is $\{a\} \cap L = \emptyset$?".

There is a finite learner for the class of all r.e. languages which uses equivalence queries: It just asks "Is $W_0 = L$?", "Is $W_1 = L$?", ... until the answer is "Yes" and then it outputs the index of the corresponding language.

For this reason, many people working on query learning have turned to investigate the number of queries needed. One important class considered is that of the regular languages. Here a set is *regular* iff a *deterministic finite automaton* accepts it. A *deterministic finite automaton* consists of a finite set $\{q_0, q_1, q_2, \ldots, q_n\}$ of states with starting state q_0, a set $A \subseteq \{q_0, q_1, q_2, \ldots, q_n\}$ which says which of the states are accepting and a transition function f which assigns to every state q_i and input symbol a a new state $q_j = f(q_i, a)$ which is taken after processing a. One can easily extend this to a function $\tilde{f}$ on words by letting $\tilde{f}(\eta_0) = q_0$ and $\tilde{f}(\eta_b a) = f(f(\eta_b), a)$ for all $a \in \{0, 1\}$ and $b \in \mathbb{N}$. Now the language accepted by the determinisitc finite automaton is $\{\eta_a : \tilde{f}(\eta_a) \in A\}$ and this is, as usual, identified with the corresponding subset $\{a : \tilde{f}(\eta_a) \in A\}$ of the natural numbers. For learning regular languages, Angluin [3] has obtained the following result.

Theorem 12.16 [3]**.** *There is a learning algorithm M and there is a polynomial p such that for every consistent teacher and every regular language R determined by a finite automaton of size n the dialogue between the learner and the teacher $(q_0, a_0, q_1, a_1, \ldots, q_m, a_m)$ satisfies at every stage m that the length of query q_m and m itself are both bounded by $p(\max\{n, |a_0|, |a_1|, \ldots, |a_{m-1}|\})$. Roughly spoken, M learns polynomially in the size of the target and the answers of the teacher, which M must of course read.*

Exercise 12.17. Construct a deterministic finite automaton which accepts all binary strings η_a such that a is not a multiple of 5. So the language contains $\eta_1 = 0$, $\eta_2 = 1$, $\eta_3 = 00$, $\eta_4 = 01$, $\eta_6 = 11$, $\eta_7 = 000$, $\eta_8 = 001$ and $\eta_9 = 010$ but it does not contain η_0, $\eta_5 = 10$, $\eta_{10} = 011$ and $\eta_{15} = 0000$.

Query learning is quite powerful. For example one can learn all r.e. languages with equivalence queries.

Example 12.18. One can learn the graphs of all total recursive functions finitely from superset queries but not behaviourally correct from text. The algorithm is to find the first e such that $\{\langle x, \varphi_e(x)\rangle : \varphi_e(x)\downarrow\} \supseteq L$. If $f = \varphi_e$ then this algorithm needs at most $e + 1$ queries.

Example 12.19. The class of all r.e. sets can be learned with finitely many equivalence queries by just asking these queries until the answer is "Yes".

For some further natural query-languages, one might need infinitely many queries in order to learn every r.e. set.

Theorem 12.20. *The class of all r.e. sets can be learned in the limit with infinitely many queries for each of the following query languages: disjointness queries, subset-queries and supersetqueries.*

Algorithm for Disjointness-Queries. The algorithm converges to the first e such that there is no x with $W_e(x) \neq L(x)$. This algorithm has to be translated into the various query-languages, here the example for disjointness queries where the queries are underlined.

1. Let $a = 0$; Let $I = \emptyset$; Let $J = \emptyset$; Let $e = 0$;
2. If $\underline{L \cap \{a\} \neq \emptyset}$ Then $I = I \cup \{a\}$ Else $J = J \cup \{a\}$;
3. Let $a = a + 1$;
4. If $I = \emptyset$ Then Conjecture $\emptyset$; Goto 2;
5. If $J = \emptyset$ Then Conjecture $\mathbb{N}$; Goto 2;
6. Choose indices i, j such that
$$W_i = \begin{cases} \emptyset & \text{if } I \not\subseteq W_e; \\ \mathbb{N} & \text{if } I \subseteq W_e; \end{cases}$$
$$W_j = \begin{cases} \emptyset & \text{if } J \cap W_e = \emptyset; \\ \mathbb{N} & \text{if } J \cap W_e \neq \emptyset; \end{cases}$$
7. If $\underline{W_i \cap L = \emptyset}$ or $\underline{W_j \cap L \neq \emptyset}$ Then Let $e = e + 1$;
8. Conjecture W_e; Goto 2;

Theorem 12.21. *If a class is learnable from a learner using texts and finitely many disjointness queries for any language learnt then there is a further learner for the same class using text only and no queries at all.*

The intuitive reason for this result is that one can figure out in the limit whether $L \cap W_e$ is empty or not. As long as no element of the meet has shown up in the text and be enumerated within the given time, a negative answer is assumed but when such an element comes up, this answer is revised to a positive one. This permits to simulate such a learner by a learner only using the text.

References

[1] Lenny Adleman and Manuel Blum. Inductive inference and unsolvability. *The Journal of Symbolic Logic*, 56:891–900, 1991.

[2] Dana Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[3] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[4] Marat Arslanov. On some generalizations of a fixed-point theorem. *Soviet Mathematics* (*Iz. VUZ*), *Russian*, 25(5):9–16, 1981, *English translation*, 25(5):1–10, 1981.

[5] Janis Bārzdiņš. Prognostication of automata and functions. *Information Processing* '71, (1) 81–84. Edited by C. P. Freiman, North-Holland, Amsterdam, 1971.

[6] Janis Bārzdiņš. Two Theorems on the Limiting Synthesis of Functions. *Theory of Algorithm and Programs* I:82–88. Latvian State University, 1974.

[7] Janis Bārzdiņš and Rūsiņš Freivalds. On the prediction of general recursive functions. *Soviet Mathematical Doklady*, 13:1224–1228, 1972.

[8] Richard Beigel. Gaps in Bounded Query Hierarchies. *Proceedings of the 14th Annual IEEE Conference on Computational Complexity*, 4-6 May 1999, Atlanta, Georgia, USA. *IEEE Computer Society*, 124-141, 1999.

[9] Manuel Blum. A machine independent theory of the complexity of recursive functions. *Journal of the Association of Computing Machinery*, 14:322–336, 1967.

[10] Lenore Blum and Manuel Blum. Towards a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[11] Valeriy K. Bulitko. Reducibility by linear Zhegalkin tables. *Siberian Mathematical Journal*, *Russian*, 21:23–31, 1980, *English translation*, 21:332–339, 1980.

[12] Vadim Bulitko. On completeness of pseudosimple sets. *Journal of Universal Computer Science*, 1:151–154, 1995.

[13] Mark Burgin. *The Rise and Fall of the Church-Turing Thesis.* Manuscript, http://arxiv.org/ftp/cs/papers/0207/0207055.pdf.

[14] Crsitian S. Calude and Gregory J. Chaitin. Randomness everywhere. *Nature*, 400:319-320, 1999.

[15] Cristian S. Calude and André Nies, Chaitin Ω numbers and strong reducibilities. *Journal of Universal Computer Science*, 3:1162–1166, 1997.

[16] Georg Cantor. Über eine Eigenschaft des Inbegriffes aller reellen algebraischen Zahlen. *Journal für die Reine und Angewandte Mathematik* 77:258–262, 1874.

[17] John Case. The power of vacillation in language learning. *SIAM Journal on Computing*, 28(6):1941–1969, 1999.

[18] John Case and Chris Lynes. Machine inductive inference and language identification. *Proceedings of the nineth International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 140:107–115, 1982.

[19] John Case and Carl Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[20] Gregory Chaitin. *Algorithmi Information Theory.* Cambridge Tracts in Theoretical Computer Science I, Cambride University Press, 1987.

[21] Martin Davis (editor). *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions.* Raven Press, 1965.

[22] Richard Dedekind. *Was sind und was sollen die Zahlen?* Braunschweig, 1888.

[23] Alexander Dëgtev. tt- and m-degrees. *Algebra and Logic*, *Russian*, 12:143–161, 1973, *English translation*, 12:78–89, 1973.

[24] Alexander Dëgtev. Three theorems on tt-degrees. *Algebra and Logic*, *Russian*, 17(3):270–281, 1978, *English translation*, 17:187–194, 1978.

[25] Alexander Dëgtev. On (m,n)-computable sets. *Algebraic Systems (Edited by D.I. Moldavanskij).* Ivanova Gos. Univ., *Russian*, pages 88–99, 1981.

[26] Alexander Dëgtev. Comparison of linear reducibility with other reducibilities of tabular type. *Algebra and Logic*, *Russian*, 21:511–529, 1982, *English translation*, 21:339–353, 1982.

[27] James C.E. Dekker. Two notes on r.e. sets. *Proceedings of the American Mathematical Society*, 4:495–501, 1953.

[28] James C.E. Dekker. Productive sets. *Transactions of the American Mathematical Society*, 78:129–149, 1955.

[29] James C.E. Dekker and John Myhill. Retraceable sets. *Canadian Journal of Mathematics*, 10:357–373, 1958.

[30] Rod Downey and Liang Yu. Arithmetical Sacks forcing. *Archive for Mathematical Logic*, 45:715–720, 2006.

[31] Lance Fortnow, William Gasarch, Sanjay Jain, Efim Kinber, Martin Kummer, Steven Kurtz, Mark Pleszkoch, Theodore Slaman, Robert Solovay and Frank Stephan. Extremes in the degrees of inferability. *Annals of Pure and Applied Logic*, 66:231–276, 1994.

[32] Rūsiņš Freivalds and Rolf Wiehagen. Inductive inference with additional information. *Elektronische Informationsverarbeitung und Kybernetik* 15:179–185, 1979.

[33] Richard Friedberg. Two recursively enumerable sets of incomparable degrees of unsolvability. *Proceedings of the National Academy of Sciences*, 43:236–238, 1957.

[34] Richard Friedberg. A criterion for completeness of degrees of unsolvability. *The Journal of Symbolic Logic*, 22:159–160, 1957.

[35] Richard Friedberg. Three theorems on recursive enumeration. *The Journal of Symbolic Logic*, 23(3):309–316, 1958.

[36] Richard Friedberg and Hartley Rogers. Reducibilities and completeness for sets of integers. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 5:117–125, 1959.

[37] William Gasarch and Mark Pleszkoch. Learning via queries to an oracle. *Proceedings of the Second Annual Conference on Computational Learning Theory* (COLT), 214–229, 1989.

[38] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik* 38:173–198, 1931.

[39] Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[40] Lane Hemaspaandra, Harald Hempel and Jörg Vogel. *Optimal Separations for Parallel versus Sequential Self-Checking: Parallelism Can Exponentially Increase Self-Checking Cost.* Technical Report TR 691, Department of Computer Science, University of Rochester, May 1998.

[41] Sanjay Jain, Daniel Osherson, James Royer and Arun Sharma. *Systems that Learn: An Introduction to Learning Theory.* The MIT Press, Cambridge, Massachusetts, 1999. Second Edition of Reference [82].

[42] Sanjay Jain and Arun Sharma. Learning with the knowledge of an upper bound on program size. *Information and Computation*, 102:118–166, 1993.

[43] Sanjay Jain and Arun Sharma. On the non-existence of maximal inference degrees for language identification. *Information Processing Letters*, 47:81–88, 1993.

[44] Carl G. Jockusch. *Reducibilities in Recursive Function Theories.* PhD Thesis, Massachusetts Institute of Technology, 1966.

[45] Carl G. Jockusch. Semirecursive sets and positive reducibility. *Transactions of the American Mathematical Society*, 131:420–436, 1968.

[46] Carl G. Jockusch. Relationships between reducibilities. *Transactions of the American Mathematical Society*, 142:229–237, 1969.

[47] Carl G. Jockusch. Degrees of functions with no fixed points. *Congress of Logic, Philosophy, and Methodology of Science VIII*, North-Holland, pages 191–201, 1989.

[48] Carl G. Jockusch. Degrees of generic sets. *London Mathematical Society Lecture Notes*, 45:110–139, 1981.

[49] Carl G. Jockusch, Manuel Lerman, Robert Soare and Robert Solovay. Recursively enumerable sets modulo iterated jumps and extensions of Arslanov's Completeness criterion. *The Journal of Symbolic Logic*, 54:1288–1323, 1989.

[50] Carl G. Jockusch and Robert Soare. Π^0_1 classes and degrees of theories. *Transactions of the American Mathematical Society*, 173:33–56, 1972.

[51] Stephen Cole Kleene. *Introduction to Metamathematics.* North-Holland, 1952.

[52] Stephen Cole Kleene and Emil Leon Post. The uppersemilattice of degrees of recursive unsolvability. *Annals of Mathematics*, 59:379–407, 1954.

[53] Stephen Cole Kleene. Hierarchies of number-theoretic predicates. *Bulletin of the American Mathematical Society*, 61:193–213, 1955.

[54] Andrei N. Kolmogorov and Vladimir A. Uspensky. Algorithms and randomness. *Theory of Probability and Applications*, 32:389–412, 1987.

[55] Antonín Kučera and Theodore Slaman. Randomness and recursive enumerability. *SIAM Journal of Computing*, 31:199–211, 2001.

[56] Martin Kummer. Numberings of $\mathcal{R}_1 \cup F$. *Computer Science Logic 1988*, *Springer Lecture Notes in Computer Science* 385:166–186, 1989.

[57] Martin Kummer. An easy priority-free proof of a theorem of Friedberg. *Theoretical Computer Science* 74:249–251, 1990.

[58] Martin Kummer. A proof of Beigel's cardinality conjecture. *The Journal of Symbolic Logic*, 57:677–681, 1992.

[59] Martin Kummer and Frank Stephan. On the structure of degrees of inferability. Journal of Computer and System Sciences, Special Issue COLT 1993, 52:214–238, 1996.

[60] Martin Kummer and Frank Stephan. Recursion theoretic properties of frequency computation and bounded queries. *Information and Computation*, 120:59–77, 1995.

[61] Alistair H. Lachlan. Lower bounds for pairs of recursively enumerable degrees. *Proceedings of the London Mathematical Society*, 16:537–569, 1966.

[62] Alistair H. Lachlan. Complete recursively enumerable sets. *Proceedings of the American Mathematical Society*, 19:99–102, 1968.

[63] Alistair H. Lachlan. On the lattice of recursively enumerable sets. *Transactions of the American Mathematical Society*, 130:1–37, 1968.

[64] Leonid Levin. On the notion of a random sequence. *Soviet Mathematics Doklady* 14:1413–1416, 1973.

[65] Donald Martin. Completeness, the recursion theorem and effectively simple sets. *Proceedings or the American Mathematical Society*, 17:838–842, 1966.

[66] Donald Martin. Classes of recursively enumerable sets and degrees of unsolvability. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 12:295–310, 1966.

[67] Per Martin-Löf, The definition of random sequences. *Information and Control*, 9:602–619, 1966.

[68] Yuri V. Matiyasevich. Diofantovost' perechislimykh mnozhestv. *Doklady Akademii Nauk SSSR*, 191:297-282, 1970 (Russian). English translation: Enumerable sets are Diophantine, *Soviet Mathematics Doklady*, 11:354-358, 1970.

[69] Yuri V. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, Cambridge, Massachusetts, 1993.

[70] Timothy McNicholl. *The inclusion problem for generalized frequency classes*. PhD thesis, Department of Mathematics, George Washington University, Washington DC, May 1995.

[71] Joseph S. Miller. Every 2-random real is Kolmogorov random. *The Journal of Symbolic Logic*, 69:907–913, 2004.

[72] Joseph S. Miller and Liang Yu. On initial segment complexity and degrees of randomness. Transactions of the American Mathematical Society, to appear.

[73] Webb Miller and Donald Martin. The degrees of hyperimmune sets. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 14:159–166, 1968.

[74] Albert Abramovich Muchnik. Negative answer to the problem of reducibility in the theory of algorithms. *Doklady Akademii Nauk S. S. S. R.*, 108:194–197, 1956.

[75] John Myhill. Creative sets. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 1:97–108, 1955.

[76] John Myhill and John C. Shepherdson. Effective operations on partial recursive functions. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 1:310-317, 1955.

[77] André Nies. Lowness properties of reals and randomness. *Advances in Mathematics*, 197:274–305, 2005.

[78] André Nies, Frank Stephan and Sebastiaan A. Terwijn. Randomness, relativization and Turing degrees. The Journal of Symbolic Logic 70:515–535, 2005.

[79] Piergiorgio Odifreddi. *Classical Recursion Theory*. North-Holland, Amsterdam, 1989.

[80] Piergiorgio Odifreddi. *Classical Recursion Theory, Volume II.* Elsevier, Amsterdam, 1999.

[81] Piergiorgio Odifreddi. Private Communication, 2007.

[82] Daniel Osherson, Michael Stob and Scott Weinstein. *Systems That Learn, An Introduction to Learning Theory for Cognitive and Computer Scientists.* Bradford — The MIT Press, Cambridge, Massachusetts, 1986.

[83] Emil Leon Post. Finite combinatory processes. Formulation I. *The Journal of Symbolic Logic*, 1:103–105, 1936.

[84] Emil Leon Post. Recursively enumerable sets of positive integers and their decision problems. *Bulletin of the American Mathematical Society*, 50:284–316, 1944.

[85] Henry Gordon Rice. Classes of enumerable sets and their decision problems. *Transactions of the American Mathematical Society* 74:358–366, 1953.

[86] Robert W. Robinson. Simplicity of recursively enumerable sets. *The Journal of Symbolic Logic*, 32:162–172, 1967.

[87] Hartley Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

[88] Gerald E. Sacks. Recursive enumerability and the jump operator. *Transactions of the American Mathematical Society*, 108:223–239, 1963.

[89] Gerald E. Sacks. On the degrees less than $\mathbf{0}'$. *Annals of Mathematics*, 77:211–231, 1963.

[90] Gerald E. Sacks. A maximal set which is not complete. *Michigan Mathematical Journal*, 11:193–205, 1964.

[91] Gerald E. Sacks. The recursively enumerable degrees are dense. *Annals of Mathematics*, 80:300–312, 1964.

[92] Gerald E. Sacks. *Degrees of Unsolvability.* Annals of Mathematics Studies 55, Princeton University Press, Princeton, New Jersey, 1966.

[93] Gerald E. Sacks. *Higher Recursion Theory.* Perspectives in Mathematical Logic, Springer-Verlag, Heidelberg, 1990.

[94] Claus Peter Schnorr. *Zufälligkeit und Wahrscheinlichkeit. Springer Lecture Notes in Mathematics*, 1971.

[95] Claus Peter Schnorr. Process complexity and effective random tests. *Journal of Computer and System Sciences* 7:376–388, 1973.

[96] Arun Sharma. A note on batch and incremental learnability. *Journal of Computer and System Sciences*, 56:272–276, 1998.

[97] Joseph Shoenfield. On degrees of unsolvability. *Annals of Mathematics*, 69:644–653, 1959.

[98] Joseph Shoenfield. Undecidable and creative theories. *Fundamenta Mathematicae*, 49:171–179, 1961.

[99] Joseph Shoenfield. The form of negations of a predicate. *Proceedings of the Thirteenth Symposium in Applied Mathematics of the American Mathematical Society*, pages 131–134, 1962.

[100] Joseph Shoenfield. A theorem on minimal degrees. *The Journal of Symbolic Logic*, 31:539–544, 1966.

[101] Joseph Shoenfield. *Degrees of Unsolvability.* North-Holland, 1971.

[102] Thoralf Albert Skolem. *Begründung der elementaren Arithmetik durch die rekurrierende Denkweise ohne Anwendung scheinbarer Veränderlicher mit unendlichem Ausdehnungsbereich.* Videnskapsselskapet Skrifter 6, 1923.

[103] Robert Soare. *Recursively Enumerable Sets and Degrees. A Study of Computable Functions and Computably Generated Sets.* Springer-Verlag, Heidelberg, 1987.

[104] Ray Solomonoff. A formal theory of inductive inference, part 1 and part 2. *Information and Control*, 7:1–22, 224–254, 1964.

[105] Ray Solomonoff. The discovery of algorithmic probability. *Journal of Computer and System Sciences*, 55:73–88, 1997.

[106] Robert Solovay. *Draft of paper on Chaitin's work.* Unpublished notes, 215 pages, 1975.

[107] Clifford Spector. On degrees of unsolvability. *Annals of Mathematics*, 64:581–592, 1956.

[108] Frank Stephan. On one-sided versus two-sided classification. *Forschungsberichte Mathematische Logik* 25 / 1996, Mathematisches Institut, Universität Heidelberg, 1996.

[109] Frank Stephan. On the structures inside truth-table degrees. *Forschungsberichte Mathematische Logik 29 / 1997, Mathematisches Institut, Universität Heidelberg*, Heidelberg, 1997.

[110] Michael J. Suslin. Sur une définition des ensembles mesurables B sans nombres transfinis. *Comptes rendus hebdomadaires des séances de l' Académie des Sciences* (Paris), 164:88–91, 1917.

[111] Boris A. Trakhtenbrot. Tabular representation of recursive operators. *Doklady Akademii Nauk S. S. S. R.*, 101:417–420, 1955.

[112] Boris A. Trakhtenbrot. Finite automata and the logic of one place predicates. *Siberian Mathematical Journal*, 3:103–131, 1962 [in Russian].

[113] Boris A. Trakhtenbrot. On autoreducibility. *Soviet Mathematics Doklady* 11:814–817, 1970.

[114] Alan M. Turing. On computable numbers with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936 and correction, 43:544–546, 1937.

[115] Alan M. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 45:161–228, 1939.

[116] Vladimir Andreevich Uspenskii. Some remarks on r.e. sets. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 3:157–170, 1957.

[117] Andrew Wiles. Modular elliptic curves and Fermat's last theorem. *Annals of Mathematics*, 141:443–551, 1995.

[118] C. E. Mike Yates. A minimal pair of recursively enumerable degrees. *The Journal of Symbolic Logic*, 31:159–168, 1966.

[119] Paul Young. On reducibility by recursive functions. *Proceedings of the American Mathematical Society*, 15:889–892, 1964.

[120] Paul Young. A theorem on recursively enumerable classes and splinters. *Proceedings of the American Mathematical Society*, 17:1050–1056, 1966.