Kolmogorov Complexity and Recursive Events

by

George Davie

Submitted in partial fulfilment of the requirements for the degree

Philosophiae Doctor

in the
Faculty of Natural and Agricultural Sciences

University of Pretoria

Pretoria

May 2000

ACKNOWLEDGEMENTS

I would like to thank my parents, George and Bertha, for supporting and putting up with me throughout the lengthy process of my Ph.D. - See Mom, you can still read and here it is! - I would also like to thank my wife Karen and daughter Colette for being such an irresistible combination of difficult and delightful. Thanks also to friends who were interested and/or interesting; to name a few (in alphabetical order): Cornelia, Frank, Konrad, Paul, Petrus, Raphael, Susan and Tony. Thanks also to Lizette without whom I'd still not be finished. To the entire department of Mathematics etc. at UNISA, thank you for making this such a terrific place to work. Lastly and importantly thanks to my supervisor Willem Fouché for sharing his love and enthusiasm for mathematics with me over the years.

SAMEVATTING

Titel: Kolmogorov Complexity and Recursive Events

Naam: George Davie

Promotor: Willem Fouché

Departement: Departement Wiskunde en Toegepaste Wiskunde

Graad: Philosophiae Doctor

Die proefskrif bestaan uit twee hoofgedeeltes: Hoofstukke 1 tot 3 en Hoofstuk 4, onderskeidelik. In Hoofstuk 1 bespreek ons die geskiedenis en begrippe rondom willekeurigheid en Kolmogorov-kompleksiteit. In Hoofstuk 2 voer ons die hooftema van die eerste deel in, naamlik: rekursiewe gebeurtenisse in willekeurige rye. In die besonder toon ons aan dat die "saamdrukbaarheidskoëffisiënt" van 'n willekeurige ry gebruik kan word om oneindig veel lokale inligting in te win oor rekursiewe gebeurtenisse in die ry. In Hoofstuk 2 pas ons hierdie resultaat toe op waarskynlikheidsleer asook op die werk van Fouché en Potgieter. In Hoofstuk 3 vertolk ons die resultaat in die raamwerk van beskrywende versamelingsleer en ondersoek ons ook die verbande tussen hierdie resultaat en bestaande werk, veral dié van Fouché. Die tweede tema (Hoofstuk 4) is 'n ondersoek na die verband tussen die beskrywingskompleksiteit van 'n versameling (soos gemeet deur die lengte van 'n kortste program wat die versameling as uitvoer gee) en die rekursiewe (on)oplosbaarheid van die versameling. In die besonder, blyk dit dat ons in baie gevalle vir beskrywingskompleksiteit kan sien as, enersyds, die oorsaak van, en, andersyds, 'n fyner meting van, onoplosbaarheid.

SUMMARY

Title: Kolmogorov Complexity and Recursive Events

Name: George Davie

Supervisor: Willem Fouché

Department: Department of Mathematics and Applied Mathematics

Degree: Philosophiae Doctor

The thesis consists of two main parts: Chapters 1 to 3 and Chapter 4, respectively. In Chapter 1 we discuss the history and concepts behind randomness and Kolmogorov complexity. In Chapter 2 we introduce the main theme of the first part: recursive events in random sequences. In particular, we show that the "compressibility-coefficient" of a random sequence can be used to obtain an infinite amount of local information about recursive events occurring in the sequence. In Chapter 2 we apply this result to probability theory and to work of Fouché and Potgieter. In Chapter 3 we interpret the result in the context of descriptive set theory and also examine the relation between our work and existing results, particularly those of Fouché.
The second theme (Chapter 4) is an exploration of the relations between the descriptive complexity of a set (as measured by the length of a shortest program which outputs the set) and the recursive (un)solvability of the set. In particular, it turns out that we can in many cases, see descriptive complexity as a cause of, and a sharper measure than, unsolvability.

Contents

1 Complexity and randomness **6**

1.1 Complexity of finite objects 6

1.2 Reservations 8

1.3 Randomness 9

1.3.1 Introduction 9

1.3.2 Notation 10

1.3.3 Using Kolmogorov complexity to define randomness 11

1.3.4 Modifying Kolmogorov complexity 12

1.3.5 Defining randomness using prefix complexity 13

1.3.6 Martin-Löf randomness 14

1.3.7 Solovay's criterion 15

1.3.8 The two definitions of randomness are equivalent 15

1.3.9 Properties of the set of random sequences 16

1.3.10 Small reservations 17

1.3.11 An example of a random sequence: Chaitin's Ω sequence 17

1.3.12 Some properties of Ω 18

2 Compressibility **19**

2.1 The main result 19

2.2 Two applications to probability theory 22

2.3 Combinatorial configurations generated
by random sequences 30

2.3.1 Every random sequence generates a universal graph 31
2.3.2 Colouring copies of a fixed graph β in U 33
2.3.3 A finite version 35
2.4 Discrepancy of random sequences 36
2.4.1 Introduction and notation 36
2.4.2 Random matrices 37
2.4.3 Random partitions 38

3 An application to descriptive set theory 40

3.1 Recursive events in random sequences 40
3.1.1 Probability laws 43
3.1.2 Does Fouché's criterion characterize the random sequences? 44
3.1.3 Early appearances in Fouché's criterion characterize the random sequences 45
3.2 Gambling against a random sequence 46
3.3 An application to Chaitin's Ω number 46
3.4 Discussion : Randomness and extensionality 47
3.5 Various generalizations fail 49
3.5.1 Two obvious generalizations fail 49
3.5.2 The oracle form fails 50

4 Complexity and unsolvability 53

4.1 Introduction 53
4.2 Rice's theorem 54
4.3 Complexity of finite fragments of uncomputable sets 55
4.4 Convergence of the halting fractions? 58
4.5 Complexity of finite fragments of computable sets 58
4.6 A gap theorem for outputting consecutive integers 59
4.7 Comparing the methods 60
4.8 Turing completeness 62
4.9 An Ω-like real for each functional computably enumerable predicate 63
4.10 An application to independent statements and proof lengths 64

Chapter 1

Complexity and randomness

1.1 Complexity of finite objects

Certain objects or processes are more complex than others. What do we mean by this? A central aspect of this distinction is that simple objects and processes are easy to *describe* or explain and complicated objects and processes are not. The following progression seems reasonable:

$$x \text{ is simple} \leftrightarrow x \text{ is } \textit{easy} \text{ to describe} \leftrightarrow x \text{ has a } \textit{short} \text{ description.}$$

For example, if x is the following binary string (which can be seen as a code for some object or process):

$$0101010101010101010101010101010101010101,$$

then although x is quite long, we feel that x is simple since it has a short description: "20 repetitions of the string 01". If on the other hand we are given the following string y

$$0100110101011111011000010101100001111010,$$

then no such short description is apparent. If it turned out that y has no description much shorter than 40 digits, we would feel strongly that y is complex, since we would then feel justified in thinking that there is *no short recipe or simple pattern* underlying y. Since all finite information can be coded as a finite binary string, we will henceforth restrict the objects to be described to finite binary strings.

As a first approximation towards formalizing this idea we then say:

- an object is **simple** if any one of its descriptions is short

  and is

- **complex** if *no* description of the object is short.

What would qualify as a *description*? Since we would like any candidate for a description to be *effective*, a natural candidate would be that of "algorithm". Let us therefore refine the tentative definition above using the idea of "algorithm" or "program for a universal computer". Choose a reference universal computer U (see [Odifreddi, Soare] as general references for recursion/computability theory). It would then be natural to define the complexity of a finite binary string x *as the length of a shortest program for U which outputs x from an empty input.*

For x the string above, the following is then a short program for x:

```
For $i = 1$ to $20$ :
write 01.
```

No such short program is apparent for y.

This leads us naturally to our first and fundamental

Definition 1 (Kolmogorov) *Let U be a universal Turing machine and let x be a finite binary string. The **Kolmogorov complexity** of x, denoted by $C_U(x)$, is the length of a shortest program (for U) that outputs x on the empty input.*

(See [Chaitin 1, Chaitin 2, Kolmogorov 1, Solomonoff 1, Solomonoff 2].) Unless explicitly stated we will assume that our universal machine is fixed and will usually write $C(x)$ instead of $C_U(x)$. It is very convenient to consider only universal machine taking finite binary strings as inputs and giving finite binary strings as outputs. This allows direct comparison between the object (as a finite binary string) and its description, or program (as a finite binary string). We will often describe the relevant programs informally, it should be understood that the actual program would be the encoding into binary form of the algorithm we give informally.

A string is then considered *complex* if its Kolmogorov complexity is close to its length.

Note that:

- The complexity of any string cannot be much higher than the length of the string itself, i.e. $C(x) < |x| + c$, since the program "write x" will output x. That is, a string is obviously a (possibly unnecessarily long) description of itself.
- Most strings have high complexity. This follows from the fact that there are few short programs. Indeed, there are 2^n strings (as objects) of length n, and only 2^{n-1} strings (as possible descriptions) of length one less, 2^{n-2} codes of length two less and so on. Hence at most one half of strings of length n have complexity $n-1$, and at most 2^{n-l} of strings of length n have complexity $n-l$. This agrees with the intuition that most strings are complicated.

1.2 Reservations

Although Definition 1 certainly captures part of our intuition surrounding complexity, there are some caveats. In particular, (computation) *time* is ignored. This leads to the following observations:

- Consider a dovetailing of the running of all possible programs on some universal computer U. Then "the $10^{10^{10^{10}}}$ th halting program of length more than 10^{10}" would be considered "simple" since this description of the program is very short (and *much* shorter than the program it describes). We will probably however, never have any idea of what the program looks like.
- There is a "simple" proof of Fermat's last theorem (FLT). This follows from the fact that the formulation of FLT along with the axioms of ZFC (Zermelo-Fraenkel set theory with the axiom of choice), gives a program to output the proof: Search through all the proofs in ZFC until you find one with last line FLT. Hence the proof is simple. In fact, any statement provable in any of the usual axiom system (such as ZFC), has a proof of Kolmogorov complexity at most the Kolmogorov complexity of the statement of the theorem, plus a constant to cover the axioms of the system and a program that will generate and search through all proofs (again under some dovetailing).

While in both cases cited above, we are given a valid description, constructing the object from the description may require enormous resources of time.

- We usually think of complexity as being in some way organized and structured. Our algorithmic definition cannot, however, distinguish between extreme structural complexity and "true randomness" and would, for example, label most coin tosses of length 100, as maximally complex, even though we would perhaps not think of this type of "noise" as complex. This shortcoming is, however, an advantage when, in the next section, we give a *definition* of randomness for infinite binary sequences using Kolmogorov complexity.

1.3 Randomness

1.3.1 Introduction

We now turn from the meaning of simple and complex to another fundamental question: *When is an infinite (binary) sequence random?*

There are *processes* which we believe to be random and would give as *examples* of randomness. One of the most important is the ubiquitous coin toss. Because of more or less vague ideas about the symmetry of the coin, the unpredictability of the influences on the coin and the non-existence of prescience we believe that the (at each stage finite) sequence of heads and tails, forms a random sequence. This example is not much use if we want a *mathematical* definition of randomness, but we will regularly compare the motives and consequences of our attempts at a formal definition of randomness, with our intuitions regarding a coin toss.

In terms of classical logic, a set - along with all its properties - is fully determined by its elements and the method in which the set is generated plays no role. Now, since an infinite binary sequence is extensionally just a set of integers (the positions of 0s (or 1s) in the sequence), an acceptable definition of randomness would be one in which a string is random by virtue of *the corresponding set of integers only.* The question is whether one can define randomness in this way and in the context of classical (as opposed to e.g. intuitionistic) reasoning. In other words, is there a satisfactory answer to the following question: *"Which classical binary reals are random?"*

Exploring the concept of "random sequence", Borel in 1909 [Borel 1, Borel 2] introduced the idea of a *normal* sequence. A normal *binary* sequence is one in which every binary word of length k appears in the limit with the "expected" relative frequency, namely 2^{-k}. In an extensional context, normality is usually taken to be a necessary condition for an infinite sequence to be random. It is not seen as sufficient, however, since there are explicitly constructible (and hence not intuitively random) sequences which also qualify as normal, notably Champernowne's sequence (see for example, p.50 of [Li and Vitányi]).

One of the most influential and far-reaching approaches to randomness is that of von Mises, who defined an infinite sequence to be random (a "collective") if relative frequency limits exist for the sequence which are unchanged under "admissible place-selections" [von Mises]. An admissible place selection is a selection of a subsequence of an infinite sequence ω where the decision whether to select the nth digit ω_n does not depend on ω_n itself. Imagine betting against a coin toss (heads you win, tails you lose). The idea is then that any system for choosing when to bet (selecting a subsequence) cannot be successful in the limit since the relative frequencies of both heads and tails in this subsequence, will again be $1/2$. This intuitively appealing characterization is mathematically problematic, and one must necessarily restrict the possible "place selections" or the definition will give the empty set (see e.g. [Li and Vitányi]). See also [van Lambalgen 1] for a revisiting of von Mises' idea and [van Lambalgen 2, van Lambalgen 3] for wide-ranging examinations of various possible approaches to randomness with emphasis on the intensional versus extensional problem.

1.3.2 Notation

We denote the natural numbers by **N**. All our sequences will be binary. We will always denote infinite binary sequences by the symbol ω, with ω_n the nth digit of ω. We write $\omega_{1:n}$ for the initial segment of ω of length n and $\omega_{k:m}$ for the segment of ω from the kth to the mth digits. We will often call finite binary sequences x - strings or occasionally - words and will use the same subscript notation for segments of strings. We will denote the length of a string x by $|x|$. We denote the set of binary strings by $\{0,1\}^*$. The Lebesgue measure is denoted by μ and $\mu(x)$, for x a finite binary string, will denote the Lebesgue measure of the interval $[0.x, 0.x + 2^{-|x|})$. All our intervals will be of this dyadic form, i.e. representable as a finite binary string x (for the interval $[0.x, 0.x + 2^{-|x|})$). We call a binary real ω computable if there is an algorithm which on input any $n \in \mathbf{N}$ outputs $\omega_{1:n}$. An index for a binary string or computable function will mean an index of a program which has as output the string or computes the function. We will use *recursive* and *computable* interchangeably for computable by a Turing machine. We will generally use the notation p_i for programs and the Greek notation ϕ for algorithms. We refer the reader to [Odifreddi] as a general reference for computability and recursion theory, in particular for undefined terms such as recursive, primitive recursive and partial computable function. We will often abbreviate computably (recursively) enumerable by c.e. We mean by 'for given x we can find effectively an $f(x)$' that we can write an algorithm which, on input any such x, will output $f(x)$. We denote by $\log x$ the base 2 logarithm of x.

1.3.3 Using Kolmogorov complexity to define randomness

In this section we will use a variant of Kolmogorov complexity to *define* randomness. Of course there are many ways we could try to define randomness, but in the following sections some arguments will be presented to show that the definition is a good one.

We have a strong intuition that there should be no *pattern* underlying the successive outcomes of a random sequence, as epitomized in a coin toss, for example. An extremely strong condition based on Kolmogorov complexity, which could ensure this, would be if *every initial segment* of the sequence (of heads and tails, for example) was as complex as possible. This would imply the following:

- The independence of consecutive outcomes of the random process. Indeed, if consecutive outcomes were not independent, we would be able to use certain initial segments of length n to either predict or rule out certain progressions of length k. This in turn would lead to descriptions of the initial segment of length $n+k$ which would be considerably shorter than $n+k$.
- The irregularity of the sequence. Indeed, the condition would imply that we cannot discern any pattern in *any* of the initial segments, even *after the fact*. (That is, we cannot study an initial segment *after* we have generated it and find a pattern underlying the segment). Thus, irregularity would clearly result from the fact that no initial segment has a description much shorter than itself, as any pattern to an initial segment would allow

us to "compress" the segment.

We formalize the idea of "as complex as possible" by requiring the existence of a constant $c \in \mathbf{N}$ such that no initial segment $\omega_{1:n}$ of ω has Kolmogorov complexity lower than $n - c$.

A priori there is no reason why this condition should hold for *any* infinite sequence, as we may suspect that the independence of consecutive digits does not rule out finding shorter programs for certain initial segments *after the fact*. In fact, this straightforward extension of the definition of Kolmogorov complexity for finite strings *does not work* for infinite sequences. In other words, *no* infinite sequence has the property that there is a natural number constant c, such that the Kolmogorov complexity of each initial segment of length n, is at least $n - c$. We can quite easily see why: It is intuitively clear that a random infinite sequence should contain all finite strings as subsequences, since if some finite sequence x appears nowhere in ω, we would feel that ω is avoiding x and would thus not be random. It therefore follows that, for every n, the string of 10^n 0s should appear in each random ω.

Now, just as clearly, the Kolmogorov complexity of a finite string is at most a universal constant more than the Kolmogorov complexity of the same string written *backwards* (there is a short program which will reverse any given string). Consider therefore, an initial segment $\omega_{1:n'}$ of ω for which the last 10^n digits are 0. The reverse of this string is then made up of an initial piece of length 10^n, of Kolmogorov complexity around $\log n$ ("write 10^n consecutive 0s"), followed by "write y", where y is the remainder of the segment. Clearly, the Kolmogorov complexity of this string (and hence the original segment $\omega_{1:n'}$), will be, for n large enough $< n - c$ for any previously given c.

Surprisingly, perhaps, a natural condition on the form of our descriptions, remedies this.

1.3.4 Modifying Kolmogorov complexity

Any program in any real computer language is *self-delimiting*. This means that if we write a complete program and then add anything to the end, the added part will be ignored by the universal machine. Programs are delimited by constructs such as the command "end" (Pascal etc.) or by the closure of the last open bracket (Lisp) and so on. The modification to Kolmogorov complexity, consists of precisely this:

All programs must be self delimiting.

This means that any extension of a valid program is either not a valid program, or alternatively, gives the same program (the added digits being ignored). Turing machines which accept only self-delimiting programs are called *prefix machines* . A prefix free *set* is a set of words (over $\{0,1\}$) such that no word is an initial segment of any other word. Clearly, any self delimiting set of programs must form a prefix-free set of words.

Definition 2 (Levin,Gács,Chaitin) *Let U be a universal prefix machine from $\{0,1\}^*$ to $\{0,1\}^*$. For x an arbitrary binary string, the* prefix complexity *$K_U(x)$ of x, is the length of a shortest program for U which outputs x on the empty input.*

(See [Levin 2, Gács 1, Chaitin 3].) Analogous to the case of Kolmogorov complexity, we will usually drop the subscript and write $K(x)$ instead of $K_U(x)$.

1.3.5 Defining randomness using prefix complexity

We can now extend the idea of complexity to infinite sequences (see [Chaitin 3]):

Definition 3 (Chaitin) *Fix a universal* prefix *machine U from $\{0,1\}^*$ to $\{0,1\}^*$. An infinite binary sequence ω is* complex *if there is a $c \in \mathbf{N}$ such that, for all $n \in \mathbf{N}$*

$$K_U(\omega_{1:n}) \geq n - c.$$

Under this slightly modified definition, an infinite binary sequence is complex with probability 1, and crucially, the set of complex sequences does not change if you change the universal prefix machine (of course, the constant c may change). See e.g. [Li and Vitányi]. The reason for this is roughly that, for any two universal machines U and U', we can write a program (a "compiler") that will translate a program for U into a program for U', adding at most a universal constant to the program length. We will sometimes denote the set of random sequences by $\mathcal{K}$.

Since this condition seems very close in spirit to the original one (of Kolmogorov - which did not work) and in view of the above remarks, it was proposed by Chaitin to *define the intuitively* *random infinite sequences to be the complex infinite sequences.* Note that defining randomness in this way is purely extensional in that the randomness of a sequence depends on its associated set of integers only.

As the coefficient c in the above definition plays a fundamental role in the next few chapters, we pause here for the following definition (regard the universal machine as fixed).

Definition 4 *Let ω be complex, that is, there exists a $c \in N$ such that $K(\omega_{1:n}) > n - c$, for all n. We define the* compressibility-coefficient *of ω as the least such c, and denote it by $c(\omega)$.*

We therefore have that for a complex ω there exists an n such that $K(\omega_{1:n}) \leq n - c(\omega)$ but for no $c > c(\omega)$ do we have $K(\omega_{1:n}) \leq n - c$ for any n.

We say that ω is *c-compressible* if and only if $c \leq c(\omega)$. We denote by K^c the set of infinite binary sequences with compressibility-coefficient at most c. We will often refer to the value of the compressibility-coefficient of a sequence as the *compressibility* of the sequence.

1.3.6 Martin-Löf randomness

We now discuss a different approach to randomness. A random infinite sequence should be *unexceptional.* One way to interpret this is the following: For any ω in the unit interval $[0,1]$ to be random, we would hope that any set in $[0,1]$ of measure 1, contains ω. We feel this reflects the fact that ω should never be part of a very small minority of sequences. That is, ω should be *typical* in every respect.

Of course, as it stands, this cannot hold as ω is part of the singleton set $\{\omega\}$ and $\mu\{[0,1] - \{\omega\}\} = 1$. However, if we consider this criterion in a constructive light, the situation changes. Consider an algorithm ϕ which generates a sequence of sets O_m in the following way: Each O_i consists of a (perhaps infinite) set of dyadic intervals, and on input any (m,k) we have ϕ outputting the kth interval of O_m. Suppose further that $\mu(O_i) < 2^{-i}$ and $O_i \supset O_{i+1}$. We call this a *Martin-Löf test* (for the reasons below).

That is, we have a sequence of sets O_i, of intervals, closed under downwards inclusion, such that the Lebesgue measure of O_i is bounded by 2^{-i}. If we now consider any random sequence ω, we would feel strongly that ω should "drop out" of the sequence $O_1, O_2, O_3, ...$ at some finite stage as the contrary would imply that ω is in each of a sequence of sets of which the measure is rapidly approaching 0.

Any sequence ω for which an i exists such that $\omega \notin O_j$ for all $j > i$, is said to *pass* the Martin-Löf test. This is the essence of Martin-Löf's definition of randomness ([Martin-Löf 1], [Martin-Löf 2]). To settle this important idea, we give an elementary example.

Example 5 *Let O_i be the set of all sequences starting with i 1s (for example* 111011 *is in O_3). Clearly $\mu(O_i) = 2^{-i}$ and $O_i \supset O_{i+1}$. Now for all sequences ω, except 1^ω, there is an i such that $\omega \notin O_j$ for all $j > i$. In other words, all sequences except 1^ω pass this particular Martin-Löf test.*

Definition 6 (Martin-Löf) *An infinite binary sequence ω is* Martin-Löf random *if it passes all Martin-Löf tests.*

Martin-Löf proved that an infinite binary sequence satisfies this condition with probability one.

1.3.7 Solovay's criterion

Of course, for the sets O_n in any Martin-Löf test, $\sum_n \mu(O_n)$ will converge. One can, however, show the following generalization of Martin-Löf's criterion to be equivalent to that of Martin-Löf. The generalization consists of the following: We replace the requirement that $\mu(O_i) < 2^{-i}$ with the requirement that $\sum \mu(O_i) < \infty$. There is therefore no longer a function controlling the rate at which the Lebesgue measure of O_i gets smaller. It can be shown that all Martin-Löf random sequences satisfy this stronger condition (see [Solovay, Chaitin 4, Shen']). In other words, if we have a computably enumerable sequence (O_i) of sets of intervals and the sum of the measures of the intervals is finite, then any random ω is contained in at most finitely many of the sets O_i of intervals. This criterion will play an important role in the next chapter.

1.3.8 The two definitions of randomness are equivalent

Since the definition via prefix complexity and the definition of Martin-Löf are both attractive, it is very satisfying that they define the *same set* of infinite binary sequences. Thus, we have the following

Theorem 7 (Schnorr, as referee to [Chaitin 3]). *A real number is complex if and only if it is Martin-Löf random.*

For a proof, see for example [Li and Vitányi].

1.3.9 Properties of the set of random sequences

The common set of "random sequences" thus obtained has many properties that one would hope for from such a set:

1. The set has Lebesgue measure 1, that is, almost all infinite binary sequences are random. This follows easily from the fact that our descriptions must be self-delimiting. [Martin-Löf 1].

2. Each random sequence satisfies all effective probability laws, such as the law of large numbers and the law of the iterated logarithm. (This follows from the definition of a Martin-Löf test - We will show in the next chapter how complexity in fact enables us to lift the effective content of these laws.)

3. Each of the sequences is unpredictable in the following sense: Let ϕ be any algorithm which takes as input finite binary strings and outputs either "the next digit is a 0", "the next digit is a 1" or "no prediction". Then if ϕ makes infinitely many predictions, it does no better than chance in the limit [Chaitin 4]. This is a very strong condition since the algorithm could embody any finite information about the sequence.

4. Each of the sequences is "typical" or generic in the sense of being part of no set of (effective) measure 0. This is the criterion of Martin-Löf.

5. Each of the sequences is Borel normal (see e.g. [Li and Vitányi]).

6. Each of the sequences is *von Mises-Wald-Church random* (allowing as "place-selections" in von Mises' formulation, all partial computable functions), see [Wald],[Church].

7. Other definitions of "random sequence", proposed in [Solovay](quoted in [Chaitin 4]), [Schnorr], [Levin 1], [Gács 2], turn out to define the same set.

For these and other reasons, the definition above is widely seen as a good definition of a random sequence. We will therefore call complex sequences *random* from now on.

1.3.10 Small reservations

We seem to have a most satisfactory definition of randomness. One of the consequences of the main result in the next chapter will nevertheless be that we cannot expect absolute unpredictability even from random sequences. In particular, we will show that infinitely many finitary facts about a random sequence are deducible from its compressibility-coefficient.

In the next section we define a particular random sequence, one moreover, which has an important meaning in the context of computability theory.

1.3.11 An example of a random sequence: Chaitin's Ω sequence

A moment's thought will show that we cannot label any explicit sequence as random, and that no deterministic algorithm can output a random sequence. This does not mean, however, that there are no naturally defined random sequences. One very significant such sequence is *Chaitin's omega number* ([Chaitin 4]):

Denote by the symbol Ω_U the *halting probability* for the universal prefix machine U. This is defined as follows: If $p_1, p_2, ...$ is the sequence of halting programs for U, then $\Omega_U = \Sigma_{n=1}^{\infty} 2^{-|p_n|}$.

When dealing with a fixed U we will drop the subscript and write only Ω.

Theorem 8 (Chaitin) *For a fixed a universal prefix machine U, the associated number Ω is random, that is, there is a $c \in N$ such that $K(\Omega_{1:n}) \geq n - c$ for all n.*

Proof. (See [Chaitin 4].) Since Ω is the sum of the numbers $2^{-|p|}$ where p is a halting program for U, we can write a program p' for U which on input the first n digits of Ω, finds a string of prefix complexity $> n$, by listing the first unlisted *output* (say x), that appears after the measure of the halting set has reached $\Omega_{1:n}$. Clearly $K(x) \geq n$, hence $K(\Omega_{1:n}) + |p'| \geq n$. This holds for each n, so setting $c = |p'|$, we have $K(\Omega_{1:n}) \geq n - c$ for all n. Therefore Ω is random. ■

We can see Ω as the probability that our universal machine will halt on a random input sequence ω of 0s and 1s because Ω is exactly the sum of the measures of the halting inputs.

(Any infinite sequence falling within the union of the associated *intervals*, will lead to a halting state in finite time, since U will be defined on some initial segment of the sequence.)

1.3.12 Some properties of Ω

The fact that Ω is both random and the limit of a series of which the partial sums are computably enumerable, has some interesting consequences. Basically, we can't use any tricks to find digits of Ω effectively. The proofs of the propositions that follow are similar to those in Chapter 4 and are not given here.

Firstly, the sum over the halting programs converges incredibly slowly to Ω:

- The number of dovetailed computation steps before the first n digits settle, is larger than any number specifiable in less than $n - K(n)$ digits. (This is enormous: for example more steps than a tower of $2^{n-K(n)}$ n's!).

We may try to specify the first n digits of $\Omega = \Sigma_{n=1}^{\infty} 2^{-|p_n|}$ by using the fact that once the first n digits of $\Sigma_{n=1} 2^{-|p_n|}$ become $\Omega_{1:n}$, these digits can no longer change. Hence $\Omega_{1:n}$ is obviously the only value of the first n digits which will not change again. We could therefore try to specify $\Omega_{1:n}$ by a program which waits until the first n digits stay the same for very long, and then outputs these n digits as $\Omega_{1:n}$. This also does not work :

- The number of times that the first n digits do not change for more steps than a tower of $2^{n-K(n)}$ n's, is of order $2^{n-K(n)}$.

Chapter 2

Compressibility

2.1 The main result

We have noted that the complex sequences are exactly the Martin-Löf (and Solovay) random sequences. A proof of this fact, due to Schnorr, can be found in, for example [Chaitin 4, Li and Vitányi, Calude]. Our main technique (Theorem 9) in this and the following chapter can be stated as a modification of the criterion of Solovay. Recall that a real ω is *Solovay random*, if ω is contained in at most finitely many of the O_i for any computably enumerable sequence (O_i) of sets of intervals with the property that $\sum \mu(O_i) < \infty$.

Chapters 2 and 3 are written in the context of prefix algorithms. We will try to make it clear in our proofs and discussions that all programs or algorithms we treat are either clearly elements of a self delimiting class of programs, or could trivially be made so.

Theorem 9 is the main result of Chapters 1 to 3 and can be stated roughly as follows: **If we have a computably enumerable sequence of dyadic intervals $I_1, I_2, \ldots$, and the Lebesgue measure of $\bigcup_{i=1}^{\infty} I_i$ is a computable real π, then there is a stage in the interval enumeration, computable in c, after which no ω with $c(\omega) = c$ can appear.**

We will use this result and small modifications of it extensively in this and the next chapter. We will show in Section 3.5.2 that the analogous result does not hold for the original condition of Solovay or Martin-Löf.

Theorem 9 *We can find a constant $k \in \mathbf{N}$ such that the following holds: Let (I_i) be a computably enumerable sequence of dyadic intervals with $\mu(\bigcup_{i=1}^{\infty} I_i)$ a* computable *real π. Further, let p_1 be a program enumerating the sequence (I_i) and p_2 be a program for π. Then no $\omega \in K^c$ can appear in the interval enumeration after the measure of the enumerated intervals has reached*

$$\pi_{1:|p_1|+|p_2|+c(\omega)+2\log c(\omega)+k}$$

Roughly, $c(\omega)$ establishes a cut-off point for the appearance of ω in the enumeration (I_i). Importantly, this is also a cut-off *time* in that no $\omega \in K^c$ can appear *after* this point. It follows that there is also a cut-off *length* $n(c)$, such that no initial segment longer than $n(c)$, of any ω with $c(\omega) = c$, can appear.

Proof of Theorem 9. We can assume, without loss of generality, that the length of the listed strings (as intervals) increases monotonically in time and that the intervals are disjoint, since

i) if not, we can subdivide the intervals, for example: $0.01 = 0.010 \cup 0.011$, and

ii) we are interested in the Lebesgue measure of the union of the intervals, not the sum of the Lebesgue measures of the individual intervals.

Now assume that some ω appears in the enumeration between the following two occurrences: The measure of the enumerated intervals reaches $\pi_{1:|p_1|+|p_2|+c+2\log c+s}$ and the measure of the enumerated intervals reaches $\pi_{1:|p_1|+|p_2|+c+2\log c+s+1}$. (Both may occur together, of course). Let the maximum of the lengths of the strings enumerated between these two occurrences be l.

Then $\omega_{1:l}$ is contained in this set of strings and hence is specifiable by giving its position in this set. Since this set of strings of maximum length l has associated Lebesgue measure less than $2^{-(|p_1|+|p_2|+c+2\log c+s)}$, there are at most $2^{l-(|p_1|+|p_2|+c+2\log c+s)}$ strings in the set. We can therefore specify the *position* of $\omega_{1:l}$ in this set, using at most

$$l - (|p_1| + |p_2| + c + 2\log c + s)$$

digits.

There is thus a program p_3, which can specify any such $\omega_{1:l}$ on an input of length bounded by

$$l - (|p_1| + |p_2| + c + 2\log c + s).$$

Besides the position, our program p_3 will of course also require the programs p_1 and p_2 and the natural numbers c and s.

Hence such a p_3 would need input lengths at most

$$|p_1| + K(c) + K(s) < |p_1| + 2\log c + 2\log s$$

to specify any such $\omega_{i:l}$. The program p_3 would therefore need at most

$$\begin{aligned} & |p_1| + |p_2| + 2\log c + 2\log s + l - (|p_1| + |p_2| + c + 2\log c + s) \\ = \; & l - c - s + 2\log s < l - c \end{aligned}$$

digits and $\omega_{1:l}$ would therefore be compressible by more than c.

We can therefore take $k = |p_3| + |p_4|$ where p_4 is any program transforming a particular 4-tuple (p_3, p_2, c, s) into a program for $\omega_{1:l}$ for the universal machine U. ∎

What exactly is the relation between this test and those of Martin-Löf and Solovay? Note firstly that we can change any test of the above form into a Martin-Löf test (and hence Solovay test) by setting O_i equal to the set of all (disjoint) intervals enumerated after the correspondence with the computable measure is accurate to i digits after the binary point. It then follows that $\mu(O_i) \leq 2^{-i}$ and $O_i \supset O_{i+1}$. Conversely, it will follow from the last section in Chapter 3 that *time* limits in terms of $c(\omega)$ do *not* hold in general, for Martin-Löf or Solovay tests. Note, however, that for Martin-Löf tests for which the total measure of enumerated intervals is a computable real, time limits will hold. In particular, if there is a computable function ϕ which on input i gives a time t_i after which the enumeration of O_i is complete, then time limits in terms of $c(\omega)$ will hold, since the conditions for Theorem 9 will apply.

The underlying thesis of this and the next chapter will be:

The value of the compressibility-coefficient of a random sequence plays a governing rôle in the behaviour of the sequence.

We will illustrate this in the context of probability theory and in the context of the work of [Fouché 1, Fouché 2, Fouché 3, Fouché and Potgieter]. In Chapter 3 we apply the result to descriptive set theory. In each of these contexts, there are natural associated **"waiting times"** which will be shown to be **computable in the compressibility coefficient** $c(\omega)$ of the relevant infinite binary sequence ω. In general, by showing that the measure of each of some class of events is computable, Theorem 9 will imply that we have computable upper bounds on the waiting times.

2.2 Two applications to probability theory

In this section we will use compressibility to refine two basic theorems of probability theory: the **law of large numbers** and the **law of the iterated logarithm**. We will stick closely to the notation of [Feller]. Before we state these two fundamental laws, some

Notation 10 *For ω an infinite binary sequence, we let S_n denote the sum of the first n digits of ω and set $S_n^* = \dfrac{S_n - \frac{n}{2}}{\frac{1}{2}\sqrt{n}}$. Let $\phi(x) = \dfrac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$ be the normal density function and $\Phi(x) = \dfrac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}y^2} dy$ be the normal distribution function. The notation $a_n \sim b_n$ means that the* ratio *of the two sides tends to* 1.

In the context of binary sequences, the strong law of large numbers (first formulated by Cantelli, see [Feller] for references), is the following:

Theorem 11 (Strong law of large numbers) *With probability one we have*

$$\frac{S_n}{n} \to \frac{1}{2}.$$

In the words of [Feller]; with probability one $\dfrac{S_n}{n} - \dfrac{1}{2}$ "becomes *and remains* small".

The law of large numbers is clearly equivalent to the following:

Theorem 12 (The law of large numbers) *For every $\varepsilon > 0$, with probability one, there occur only finitely many of the events*

$$|\frac{S_n}{n} - \frac{1}{2}| > \varepsilon.$$

The law of the iterated logarithm (due to Khintchine, see [Feller] for references) gives upper bounds for the fluctuations of $S_n^* = \frac{S_n - \frac{n}{2}}{\frac{1}{2}\sqrt{n}}$.

Theorem 13 (The law of the iterated logarithm - Khintchine) *With probability one we have:*

$$\limsup_{n\to\infty} \frac{S_n - \frac{n}{2}}{\sqrt{\frac{n}{2}\log\log n}} = 1.$$

This means: For $\lambda > 1$, with probability one, only finitely many of the events

$$S_n > \frac{n}{2} + \lambda\sqrt{\frac{n}{2}\log\log n} \tag{2.1}$$

occur; and for $\lambda < 1$, with probability one, infinitely many of the events

$$S_n > \frac{n}{2} + \lambda\sqrt{\frac{n}{2}\log\log n} \tag{2.2}$$

occur.

Note that these two laws are non-effective on (at least) two counts. Given a randomly chosen sequence ω, the laws firstly only hold with *probability one* and, secondly, we are told nothing about the *waiting times* involved. That is, nothing is stated about any of the following:

- In the strong law of large numbers. For a given ε, the largest m for which $|\frac{S_m}{m} - \frac{1}{2}| > \varepsilon$ or
- In the law of the iterated logarithm:
  - For a given $\lambda > 1$, the largest n for which $S_n > \frac{n}{2} + \lambda\sqrt{\frac{n}{2}\log\log n}$

    or

– For a given $\lambda < 1$ and n, the smallest $n' > n$ such that $S_{n'} > \frac{n'}{2} + \lambda\sqrt{\frac{n'}{2}\log\log n'}$.

Regarding the above objections: The fact that each random sequence satisfies all effective probability laws, means that we can change the statement:

P holds with probability one

to:

P holds for each random ω.

We can also use the definition of randomness to address the waiting time objection. In particular, we will show the following:

The compressibility of a random sequence determines upper bounds on the waiting times for the events involved in the probability laws.

We will prove the following stronger versions of these two laws:

Theorem 14 (Strong law - effective form) *Let ω be random and let $c(\omega) = c$. For any given ε we can effectively find an $n(c,\varepsilon)$ such that for all $n > n(c,\varepsilon)$*

$$|\frac{S_n}{n} - \frac{1}{2}| < \varepsilon.$$

Theorem 15 (Law of the iterated logarithm - effective form) *Let ω be random and let $c(\omega) = c$.*

1) *For a given $\lambda > 1$, we can find effectively an $n(c,\lambda)$ such that, for all $n > n(c,\lambda)$*

$$S_n \leq \frac{n}{2} + \lambda\sqrt{\frac{n}{2}\log\log n}. \tag{2.3}$$

2) *For a given $\lambda < 1$ and $m \in \mathbf{N}$, we can find effectively an $n(c,\lambda,m)$ such that, for some n such that $m \leq n \leq n(c,\lambda,m)$*

$$S_n > \frac{n}{2} + \lambda\sqrt{\frac{n}{2}\log\log n}. \tag{2.4}$$

Proving the effective versions above via compressibility requires explicit upper bounds on the probabilities of the various events involved in these laws. The proofs in the classic text [Feller], are well suited to this goal and will require only small modifications to prove the versions above. In particular, unless otherwise stated, we will assume all numbers to be *computable*. This should clearly not lead to a loss of generality. We will need the following:

Lemma 16 ([Feller]) *If $x_n \to \infty$ in such a way that $\frac{x_n^3}{\sqrt{n}} \to 0$ then*

$$P\{S_n^* > x_n\} \sim \frac{1}{\sqrt{2\pi}} \cdot \frac{1}{x_n} e^{-\frac{1}{2}x_n^2}.$$

Proof. (See e.g. [Feller].) Let $a_k = b(v+k; 2v, \frac{1}{2})$ where b is the binomial distribution, informally, $b(v+k; 2v, \frac{1}{2})$ is the fraction of binary sequences of length $2v$ containing exactly $v+k$ 1s. For $h = 2/\sqrt{n}$ we have (see [Feller])

$$a_k \sim h\phi(kh) \tag{2.5}$$

with error smaller than $\frac{k^3}{v^2}$. Hence

$$P\{S_n^* > x_n\} \sim \sum_{k=r_n}^{\infty} h\phi(kh) \tag{2.6}$$

where r_n is an integer such that $|r_n h - x_n| < h$.

Now $\sum_{k=r_n}^{\infty} h\phi(kh)$ lies between $1 - \Phi(x_n - 2h)$ and $1 - \Phi(x_n + 2h)$. Using the inequality

$$[x^{-1} - x^{-3}]\phi(x) < 1 - \Phi(x) < x^{-1}\phi(x) \tag{2.7}$$

(see [Feller]) we get that

$$\Phi(x_n + 2h) - \Phi(x_n - 2h) < 4h\phi(x_n - 2h) \to 0,$$

and it follows that

$$\sum_{k=r_n}^{\infty} h\phi(kh) \sim 1 - \Phi(x_n)$$

and using 2.7 again we get that

$$\sum_{k=r_n}^{\infty} h\phi(kh) \sim \frac{1}{\sqrt{2\pi}} \cdot \frac{1}{x_n} e^{-\frac{1}{2}x_n^2}.$$

It should be clear that we can, given any $\varepsilon > 0$, choose k large enough in 2.5 such that

$$(1-\varepsilon)\frac{1}{\sqrt{2\pi}} \cdot \frac{1}{x_n} e^{-\frac{1}{2}x_n^2} < \sum_{k=r_n}^{\infty} h\phi(kh) < (1+\varepsilon)\frac{1}{\sqrt{2\pi}} \cdot \frac{1}{x_n} e^{-\frac{1}{2}x_n^2}. \tag{2.8}$$

holds. ■

Proof of the law of large numbers. The idea of the proof is simply that the compressibility of a sequence must be high if, for given ε, $|\frac{S_n}{n} - \frac{1}{2}| > \varepsilon$ for a large enough n. Let $a > 1$ and let A_k be the event

$$|S_k^*| = |\frac{S_k - \frac{k}{2}}{\sqrt{\frac{k}{4}}}| \geq \sqrt{2a \log k}.$$

Now, since $\frac{(2a \log k)^{\frac{3}{2}}}{\sqrt{k}} \to 0$ we have by Lemma 16 that we can choose n large enough such that

$$P\{|S_n^*| > \sqrt{2a \log n}\} < e^{-a \log n} = \frac{1}{n^a}.$$

Since $\sum_{k=1}^{\infty} \frac{1}{k^a}$ converges for $a > 1$, we can find for any given σ, an n such that $\sum_{k=n}^{\infty} P\{|S_k^*| > \sqrt{2a \log k}\} < \sum_{k=n}^{\infty} \frac{1}{k^a} < 2^{-\sigma}$. So for given ε and a we can find an n such that the measure of the union of the events $A_k, k \geq n$, converges effectively, hence is a computable real. Theorem 9 then gives the result. ■

For the law of the iterated logarithm we also need the following

Lemma 17 (See e.g. [Feller]) *There exists a constant σ independent of n such that*

$$P\{S_n > \frac{n}{2}\} > \sigma \text{ for all } n.$$

Furthermore, if x is fixed and A is the event that for at least one k with $k \leq n$, we have $S_k - \frac{k}{2} > x$, then

$$P\{A\} \leq \sigma^{-1} P\{S_n - \frac{n}{2} > x\}.$$

Proof of the law of the iterated logarithm. The idea of the proof is simply that the compressibility of a sequence must be high if, for given $\lambda > 1$ and for a single large enough n we have

$$S_n > \frac{n}{2} + \lambda\sqrt{\frac{n}{2}\log\log n}. \tag{2.9}$$

And similarly, that the compressibility must be high, if for a given $\lambda < 1$ and for many consecutive large n, we have

$$S_n \leq \frac{n}{2} + \lambda\sqrt{\frac{n}{2}\log\log n}.$$

We now prove part (1) of Theorem 15, again following the proof in [Feller] where possible: Let $\lambda > 1$, let γ be a number between 1 and λ and let n_r be the integer nearest to γ^r. Let B_r be the event that the inequality

$$S_n - \frac{n}{2} > \lambda\sqrt{\frac{n_r}{2}\log\log n_r} \tag{2.10}$$

holds for at least one n with $n_r \leq n \leq n_{r+1}$.

Clearly 2.9 can only hold infinitely often if 2.10 holds infinitely often.

We will show that $\sum P\{B_r\}$ converges effectively.

By Lemma 17 we have that

$$P\{B_r\} \le \sigma^{-1} P\{S_{n_{r+1}} - \frac{n_{r+1}}{2} > \lambda\sqrt{\frac{n_r}{2}\log\log n_r}\} = \sigma^{-1} P\{S^*_{n_{r=1}} > \lambda\sqrt{2\frac{n_r}{n_{r+1}}\log\log n_r}\}.$$

Now $n_{r+1}/n_r \sim \gamma < \lambda$, and we can therefore choose r large enough such that

$$P\{B_r\} \le \sigma^{-1} P\{S^*_{n_{r+1}} > \sqrt{2\lambda\log\log n_r}\}.$$

By Lemma 16 we can choose r large enough such that

$$P\{B_r\} \le \sigma^{-1} e^{-\lambda\log\log n_r} = \frac{1}{\sigma(\log n_r)^\lambda}$$

and $\frac{1}{\sigma(\log n_r)^\lambda}$ is as close as we like to $\frac{1}{\sigma(r\log\gamma)^\lambda}$. Thus we can choose for any given k, an s such that $\sum_{r=s}^\infty P\{B_r\} < 2^{-k}$ and hence $\sum_{r=s}^\infty P\{B_r\}$ - the sum of the Lebesgue measures of the events B_r - is a computable number. Part (1) of Theorem 15 then follows by Theorem 9.

We now prove part (2) of Theorem 15. Let $\lambda < 1$ and choose a number η so close to 1 that

$$1 - \eta < (\frac{\eta - \lambda}{2})^2$$

and choose for γ an integer so large that $\frac{\gamma - 1}{\gamma} > \eta > \lambda$. Put $n_r = \gamma^r$. Let $D_r = S_{n_r} - S_{n_{r-1}}$ and let A_r be the event

$$D_r - \frac{n_r - n_{r-1}}{2} > \eta\sqrt{\frac{n_r}{2}\log\log n_r}. \tag{2.11}$$

Note that the events A_r are independent. We will show that $P\{A_r\} > \frac{1}{r}$.
Indeed

$$P\{A_r\} = P\{\frac{D_r - (n_r - n_{r-1})\frac{1}{2}}{(n_r - n_{r-1})\frac{1}{4}} > \eta\sqrt{2\frac{n_r}{n_r - n_{r-1}}\log\log n_r}\}.$$

Here $n_r/(n_r - n_{r-1}) = \gamma/(\gamma - 1) < \eta^{-1}$. Hence

$$P\{A_r\} \geq P\{\frac{D_r - (n_r - n_{r-1})\frac{1}{2}}{\sqrt{(n_r - n_{r-1})\frac{1}{4}}} > \sqrt{2\eta\log\log n_r}\}.$$

Using Lemma 16 again, we can choose r large enough such that

$$P\{A_r\} > \frac{1}{\log\log n_r}e^{-\eta\log\log n_r} = \frac{1}{(\log\log n_r)(\log n_r)^{\eta}}.$$

Since $n_r = \gamma^r$ and $\eta < 1$, we can choose r large enough such that $P\{A_r\} > \frac{1}{r}$. This means that the probability that none of the events $A_r, A_{r+1}, ..., A_{r+n}$ occur, is at most $(1 - \frac{1}{r})(1 - \frac{1}{r+1})...(1 - \frac{1}{r+n})$ which is exactly $(\frac{r-1}{r+n})$. We can therefore effectively find, given λ, n_r and c, a number n such that

$$(1 - \frac{1}{r})(1 - \frac{1}{r+1}).....(1 - \frac{1}{r+n}) < 2^{-c-k}.$$

In other words, we can choose r large enough to make the intersection of the events $A_n, n > r$ arbitrarily small (note that we could describe the intersection above of measure 2^{-c-k} using only around $\log(c) + \log(k)$ digits), hence, by the ideas of Theorem 9, it follows that we can find, given $c(\omega)$, an n such that ω cannot be in $A_r \cap A_{r+1} \cap ... \cap A_{r+n}$.

Can S_{n_r-1} be so much smaller than 0 as to make S_{n_r} too small? Since the first part of the theorem states that for each $\varepsilon > 0$, we can find an N such that, with probability $1-\varepsilon$ or better, and for all $r > N$

$$|S_n - \frac{n_{r-1}}{2}| < 2\sqrt{\frac{n_{r-1}}{2}\log\log n_{r-1}}. \tag{2.12}$$

By our choice of η we get

$$4n_{r-1} = 4^r\gamma^{-1} < n_r(\eta - \lambda)^2$$

and hence 2.12 implies

$$S_{n_{r-1}} - \frac{n_{r-1}}{2} > -(\eta - \lambda)\sqrt{\frac{n_r}{2} \log\log n_r}.$$

If we therefore add this to 2.11 we get 2.4 with $n = n_r$. Part (2) of Theorem 15 follows. ■

2.3 Combinatorial configurations generated by random sequences

One of the main themes of the work in [Fouché 1], [Fouché 3], [Fouché and Potgieter] is to examine random sequences in contexts which make apparent certain properties of these sequences which are not so when seen only as binary sequences of 0s and 1s. A second theme is to use random sequences to bring to the fore the self reflexive characteristics of certain important combinatorial configurations.

An illustration of these themes is obtained by considering a random sequence as a code for an infinite graph.

In particular,

- if we let any random sequence generate an infinite graph (number the edges of the countable, complete graph $K_{\mathbf{N}}$ in some canonical way and let $\omega_n = 1$ mean that edge n is in the graph), then the graph will be isomorphic to the universal (Rado) graph ([Rado]), a most symmetric and self-reflexive object. In particular, every finite or countable graph can be imbedded into it, including itself. Further, and as illustration of the *second* theme
- if we use a random sequence to colour all copies of some fixed finite graph β in a universal graph U, then there will be a copy U' of U inside U with all the copies of β in U' of the same colour.

In a strong sense, this shows that the structure of (a random object coded as) the universal graph is invariant under a large class of partitions. Put another way, partitioning the universal graph in a random way (and bear in mind that the probability that a partition is random is one) we find that we can recover the universal graph in one of the partitions (actually both). The organization of the universal graph is therefore immune to (almost all) partitions. Since every random graph is a universal graph, this means that each random graph is extremely robust under partitions. That is, the graph is self reflexive to such a strong degree, and the organization of the graph is contained in so many different ways in the graph, that any random partition leaves us with 'the same' organization. In a sense, the structure of the graph survives any random partition.

The above results can be found in [Fouché 3]. We will use the compressibility-coefficient to lift the effective content of these results by showing, similarly to the probability laws, that the associated waiting times are computable in $c(\omega)$. Our notation is based on that of [Fouché 3] and our proofs differ mainly in the details of the induction step.

In this work, the Lebesgue measure of the appropriate enumerated set will always be *one* as we will be dealing with events which occur with probability one.

2.3.1 Every random sequence generates a universal graph

We let a random sequence ω generate an infinite graph as follows. Start with the complete graph $K_{\mathbf{N}}$ on $\mathbf{N}$ vertices, numbered $1, 2, 3\ldots$. Also number all the edges. The following "dovetail" mapping $\mathbf{N}^2 \to \mathbf{N}$ is an example:

$$\text{edge } \{1,2\} \to 1, \text{edge } \{1,3\} \to 2, \text{edge } \{2,3\} \to 3 \text{ and so on.}$$

Whatever mapping we use is required to be a *computable bijection* between $\mathbf{N}^2$ and $\mathbf{N}$ in the sense that both f and f^{-1} should be computable and onto functions. That is, there is a program for f which on input any vertex pair $\{i, j\} \in \mathbf{N}^2$, gives as output the number of the edge $\{i, j\}$ and there is a program for f^{-1}, which on input $i \in \mathbf{N}$, gives as output the vertex pair $\{k, l\}$ which has number i. This will play an important role.

A given infinite binary sequence ω then acts as a code for an infinite graph in the following way: Delete the ith edge of $K_{\mathbf{N}}$ if and only if the ith digit of ω is a 0. The graph obtained in the limit we call the graph generated by ω. As in [Fouché 3], we show that every *countable* graph can in fact be imbedded into the graph generated by ω.

Consider therefore any random sequence ω and let ω generate a subgraph G_ω of $K_{\mathbf{N}}$ as follows:

$$G_\omega \text{ contains the edge } \{m, n\} \text{ if and only if } \omega_{f(m,n)} = 1.$$

Now consider any countable graph C. We want to show that C can be imbedded into G_ω. We can clearly consider C as a graph with vertices natural numbers by relabelling the vertices if necessary.

To show that we can imbed C into G_ω we will find an infinite sequence $v_0, v_1, \ldots$ of vertices in G_ω such that, for each n, the subgraph of C spanned by $0, 1, \ldots, n$ is isomorphic to the subgraph of G_ω spanned by $v_0, v_1, \ldots, v_n$. We denote these graphs by C_n and G_n respectively.

The proof is by induction:

Basis step: Set $v_0 = 0$.

Induction step: Suppose that $v_0, v_1, \ldots, v_{n-1}$ have been defined with $v_0 < v_1 < \ldots < v_{n-1}$. That is, we have found a graph G_{n-1} isomorphic to C_{n-1}.

We will now show that we can continue this embedding to C_n. To do this, we must find a vertex v_n in the graph G_ω such that the graph spanned by vertices $v_0, v_1, \ldots v_{n-1}, v_n$ is isomorphic to the graph spanned by vertices $0, 1, 2, \ldots, n$ of C.

Let $\alpha = \alpha_0\alpha_1\ldots\alpha_{n-1}$ be the binary word of length n such that $\alpha_i = 1$ if $\{n, i\}$ is an edge of C, and 0 otherwise.

Now, in order to extend our imbedding to C_n, we systematically test as candidates for v_n the following vertices of G_ω : $v_{n-1} + 1, v_{n-1} + 2, \ldots$ (in this order).

Since ω generates the graph G_ω, we must, for each such candidate v_i, examine the digits

$$\omega_{f\{v_0,v_i\}}, \omega_{f\{v_1,v_i\}}, \ldots \omega_{f\{v_{n-1},v_i\}}$$

in ω corresponding to the (possible) edges between vertices v_i and vertices $v_0, v_1, \ldots, v_{n-1}$. The candidate v_i is then successful if and only if

$$\omega_{f\{v_0,v_i\}}\omega_{f\{v_1,v_i\}}...\omega_{f\{v_{n-1},v_i\}} = \alpha_0\alpha_1...\alpha_{n-1}.$$

Let $w_i = \omega_{f\{v_0,v_i\}}\omega_{f\{v_1,v_i\}}...\omega_{f\{v_{n-1},v_i\}}$ be the word obtained for candidate vertex v_i.

It is clear that any digit ω_m is examined in connection with at most one of the candidate vertices v_i. In other words, the positions m at which we examine digits ω_m of ω, are disjoint for different candidate vertices v_i.

Consider now those binary sequences which code for edge partitions for which none of the l consecutive candidate vertices $v_{n-1}+1, v_{n-1}+2, ..., v_{n-1}+l$ is successful. There are clearly at most

$$(1-2^{-n})^l$$

such sequences. Hence the Lebesgue measure of the computably enumerable set of infinite sequences which do code for an extension, converges to 1 as $l \to \infty$. Hence by Theorem 9 we have that *we can find in terms of* c, *a number of vertices* l *such that we must have an extension of the imbedding to* C_{n+1} *within* l *attempts if the edges are chosen by a random* ω *with* $c(\omega) \leq c$.

In particular, while [Fouché 3] shows that the imbedding of any countable graph C into the randomly generated graph G is computable in C, G and ω, it follows from the above that it is in fact *primitive* recursive in (c and) C, G and ω in the sense that we have upper bounds on the number of steps taken during each construction stage. To be precise, we have a primitive recursive relation between the stages of imbedding and the number of steps taken to complete each stage.

2.3.2 Colouring copies of a fixed graph β in U

We now show that if we colour the copies of some fixed finite graph in a universal graph U, we will get a copy U' of U in U with all copies of β in U' of the same colour.

We use a particular universal graph U for our construction, namely $I[\mathbf{N}]$, our notation for the intersection graph on the set of natural numbers (defined below). As all universal graphs are isomorphic, this will prove the above proposition.

Let $I[n]$ therefore denote the intersection graph on $[n]$, that is, the vertices of $I[n]$ are the non-empty subsets of $[n]$ and there is an edge between a and b if and only if a and b are not disjoint (and $a \neq b$).

Example 18 *Consider the graph $I[0,1,2]$. Note that replacing any number in the set $\{0,1,2\}$ by a number not in the set will give an isomorphic graph, so, for example, $I[0,1,3]$ will give the same graph, as will $I[4,1,2]$. This will play an important role in constructing a copy of $I[\mathbf{N}]$ in $I[\mathbf{N}]$.*

We now fix some finite graph β and colour all the copies of β in $I[\mathbf{N}]$ in the following way: Let ϕ be an effective and computable enumeration of all the copies of β in $I[\mathbf{N}]$. A natural example would be to systematically number, in some canonical lexical order, all the copies found in larger and larger subgraphs of $I[\mathbf{N}]$, such as $I[1], I[2], ..., I[n], ...$.

Now let ω be a random sequence and suppose that these subgraphs of $I[\mathbf{N}]$ are coloured by ω in the following way:

If the nth digit of ω is a 1, we colour the nth copy of β
by colour 1 and the same for 0.

We want to show that, for this colouring, we can find a copy of $I[\mathbf{N}]$ in $I[\mathbf{N}]$ in which *all* copies of β are the same colour.

Note that the colour of the nth β effectively gives us the nth digit of our random sequence.

We shall show that there is an infinite sequence of numbers $i_0, i_1...$ such that for each i, $I[n]$ is isomorphic to $I[i_0, i_1, ...i_n]$ and all the copies of β in $I[i_0, i_1, ...i_n]$ are the same colour. The proof is by induction:

Basis step: Set $i_0 = 0$.

Induction step:

Suppose that we have found $i_1 < i_2 < ... < i_n$ such that $I[i_1, i_2, ..., i_n]$ is isomorphic to $(\equiv)$ $I[n]$ and all the copies of β in $I[i_1, i_2, ..., i_n]$ are the same colour.

We therefore want to find a number i_{n+1} such that $I[i_1, i_2, ..., i_n, i_{n+1}] \equiv I[n+1]$ with all copies of β in $I[i_1, i_2, ..., i_n, i_{n+1}]$ of the same colour. We do this by examining the vertices $i_n + 1, i_n + 2, ...$ and checking whether any one of these is a successful candidate for i_{n+1}. (Note

that, from the example above, $I[i_1, i_2, ..., i_n, i_n + m] \equiv I[n+1]$ for every m.)
Let the number of *new* copies of β in $I[n+1]$ be equal to a number $k \in \mathbf{N}$.

Given a random and uniform colouring, the probability that a vertex $i_n + l$ is not successful is then clearly $(1-2^{-k})$. As no new copy of β in $I[i_1, i_2, ..., i_n, i_n+m]$ can be in $I[i_1, i_2, ..., i_n, i_n+m']$ for $m \neq m'$, the probability that none of the l consecutive vertices $i_n + 1, i_n + 2, ..., i_n + l$ will be successful, is

$$(1-2^{-k})^l.$$

Now vertex $i_n + m$ failing, means that the digits of ω which correspond to the colours of the new copies of β in $I[i_1, i_2, ..., i_n, i_n + m]$, are not all 0's (or 1's, depending on the colouring of the copies in $I[i_1, i_2, ..., i_n]$). Since we can find these digit positions from the number $i_n + m$, we can enumerate all strings of length s for which the appropriate digits are not correct. We can do this for each of the l consecutive numbers $i_n + 1, i_n + 2, ..., i_n + l$.

Let $m(l)$ be the largest digit position associated with the colourings in the l extensions. Then the number of sequences of length $m(l)$ which do not code for the correct colours for any of the l new vertices is at most $(1-2^{-k})^l$.

So, like the construction above where we extended the graph-imbedding, the Lebesgue measure of the c.e. set of infinite sequences here which do eventually code for the correct colours converges to 1 as $l \to \infty$. Hence by Theorem 9 *we can find in terms of c, a number l such that we must have a coding for the correct colours within l attempts if the edges are chosen by a random ω with $c(\omega) \leq c$.*

So again, while Fouché shows that constructing a copy of U inside the universal graph $I[\mathbf{N}]$ coloured by ω with all copies of β contained in it of the same colour, is computable in β and ω, it follows from the above that it is in fact *primitive* recursive in (c and) β and ω since we have upper bounds on the number of steps taken for each stage.

2.3.3 A finite version

We give a finite version of the previous construction. Let $[H, \beta]$ be the set of induced copies of the graph β in the graph H.

Theorem 19 *For given finite graphs β, H with $[H,\beta] \neq \emptyset$ and given a constant c, we can effectively find an n such that if we colour the copies of β in $I[n]$ by any binary string x of length $|\binom{I[n]}{\beta}| = n$ with $C(x) \geq n - c$, there will be copies H_0, H_1 of H in I_n such that $[H_i, \beta]$ is of colour i with respect to x.*

Proof. Given c, we will effectively find an n such that a colouring of $I[n]$ by any x of length $|[H_0,\beta]|$ ($= m$ say) with $C(x) \geq m - c$ will give an H_0 with all copies of colour 0.

Consider all the disjoint copies of H_0 and the colours of their βs. Say there are l disjoint copies of H_0 and none of them has all βs of colour 1. The probability of any single copy of H_0 containing only βs of colour 0 is 2^{-m}, so the probability that none of the l contains only such βs is $(1-2^{-m})^l$. We need only take n (hence m) large enough such that $(1-2^{-m})^{-l}$ saves more digits than $C(n)$ and the other inputs use. In other words, n such that $(1-2^{-m})^{-l} < 2^{-d}$, where the length of all other necessary inputs together is d digits. That is, since we are able to specify any particular string which does not code for a single colour H_0 by its position amongst such strings, and we can specify this set for given n on input n, we need only copy the last few steps of either of the above constructions to prove the theorem. ■

The same theorem will hold for the prefix complexity K.

2.4 Discrepancy of random sequences

2.4.1 Introduction and notation

Let ϕ be a universal prefix machine, fixed in what follows. The paper [Fouché 2] is a continuation of the themes explored above and studies the discrepancy of random sequences when seen as coding for set systems. We use similar methods as those used above to lift the effective content of the results in [Fouché 2].

For the sake of clarity and direct comparison with [Fouché 2], we modify our notation to that in [Fouché 2]. Let $\mathcal{A}$ be a family of subsets of a finite set A and $\sum$ be a set of mappings χ from A to the set $\{-1,1\}$. The *discrepancy of $\mathcal{A}$ with respect to* $\sum$ is $\min_{\chi\in\sum} \max_{X\in\mathcal{A}} |\sum_{x\in X} \chi(x)|$. If $\sum$ is the entire set of mappings from A to $\{-1,1\}$, we will talk of the *discrepancy of* $\mathcal{A}$. Denote the discrepancy by $\sigma(\mathcal{A})$. Since we are going to view binary sequences as codes for the entries of matrices, we let $(i,j) \longmapsto <i,j>$ be a computable bijection from $\mathbf{N} \times \mathbf{N}$ onto $\mathbf{N}$. For a binary sequence ω, we define the family $\mathcal{A}(\omega) = (A_i)_{i\geq 1}$ of subsets of $\mathbf{N}$ by $j \in A_i \Leftrightarrow \omega_{ij} = 1$, where we now write ij for $< i, j >$. For $n \geq 1$, let $\mathcal{A}_n(\omega)$ be the family of sets $A_i \cap [n], i = 1, ..., n$. That is, we can see $\mathcal{A}_n(\omega)$ as the set system given by the $n \times n$ matrix in the top left-hand corner of the infinite matrix generated by ω. For brevity, when dealing directly with the matrix, we will often talk of the *discrepancy of the $n \times n$ submatrix* instead of the *discrepancy of the set system given by the submatrix.*

Fouché, in [Fouché 2], firstly examines the case where the matrix is infinite and is generated by a random sequence ω. For each of the top left-hand $n \times n$ submatrices, the discrepancy of the set system represented by this submatrix is considered, which leads to Theorem 1 of [Fouché 2]. Secondly, the case where the matrix is computable and the colouring random is examined. This leads to Theorem 3 in [Fouché 2].

In this section we use compressibility to prove sharper versions of these results, again showing the associated waiting times to be computable in the relevant compressibility coefficients.

2.4.2 Random matrices

Theorem 1 of [Fouché 2] states:

Theorem 20 *There exists a universal constant $\tau > 0$ such that, for each random string ω, there exists a natural number n_ω such that, for all $n \geq n_\omega$, the discrepancy of $\mathcal{A}_n(\omega)$ satisfies*

$$\sigma(\mathcal{A}_n(\omega)) \geq \tau\sqrt{n}.$$

In some sense then, set systems generated by random sequences have maximal discrepancy, since [Spencer] has shown that for any set system $\mathcal{A}$ consisting of n subsets of $[n]$ we have $\sigma(\mathcal{A}) \leq 6\sqrt{n}$.

The idea in using compressibility is to show that for given c, if the discrepancy $\mathcal{A}_n(\omega)$ stays low for too large n, then the compressibility of ω must be larger than c. Thus:

Theorem 21 *There exists a universal constant $\tau > 0$ such that, for each random string ω with $c(\omega) = c$ we can find a natural number n_ω such that, for all $n \geq n_\omega$, the discrepancy of $\mathcal{A}_n(\omega)$ satisfies $\sigma(\mathcal{A}_n(\omega)) \geq \tau\sqrt{n}$.*

In other words, the stage after which the discrepancy must be at least $\tau\sqrt{n}$, is computable in $c(\omega)$. We need the following lemma:

Lemma 22 *Let $(W_{i,j} : 1 \leq i,j \leq n)$ be n^2 random variables such that each $W_{i,j}$ assumes each of the values 0 and 1 with probability 1/2. For each $v = (v_1, v_2, ..., v_n) \in \{-1,1\}^n$, put $L_i(v) = \sum_{j=1}^{n} v_j W_{i,j}$. Then we can find numbers $\tau, \varepsilon, c_0 < 1/2 - \varepsilon$ and n^* such that*

$$P\{\exists_{v\in\{-1,1\}^n}\forall_{i\leq n}(|L_i(v) < \tau\sqrt{n})\} \leq (2c_0)^n.$$

Proof. Directly from Lemma 1 in [Fouché 2]. The effectivity follows from the proof of the lemma and the error terms in Section 2.2 for convergence to the normal distribution. ■

Proof of theorem 21. Consider a random sequence ω with $c(\omega) = c$. Let $\phi : \mathbf{N}\times\mathbf{N} \mapsto \mathbf{N}$ be our computable bijection. Now consider the sequence of $n \times n$ submatrices in the top left-hand corner of the generated infinite matrix. It follows from the lemma and the choice of values that for size $n \times n$, $n > n^*$, the probability that the submatrix has discrepancy lower than $\tau\sqrt{n}$ is less than $(1/2 - 2\varepsilon)^n$. We can therefore specify an $A^n, n > n^*$ with discrepancy lower than $\tau\sqrt{n}$, by giving the lexical position of A^n amongst the $n\times n$ matrices with discrepancy less than $\tau\sqrt{n}$. Now, clearly $\sum_{n=n^*}^{\infty}(1/2 - 2\varepsilon)^n$ is a computable real. Since ω codes for A^n, the result follows from Theorem 9. ■

2.4.3 Random partitions

We now consider the case where our countable matrix is computable and our partition (column vector) is random.

Notation: For A an $\omega \times \omega$ matrix over $\{0,1\}$, let A^n be the $n \times n$ submatrix in the upper left-hand corner of A. For X a countable column vector over $\{-1,1\}$, write $\bar{X}(n)$ for the first n entries of X. For an $n \times n$ matrix B and column vector X as above, we write $||BX||$ for $\sup\{|B_iX| : i = 1, ..., n\}$, where B_i denotes the ith row of B.

The following theorem appears in [Fouché 2].

Theorem 23 *Let A be a computable countable matrix over $\{0,1\}$. There is a universal constant $C > 0$ such that, for every random ω, there is some n_ω, such that, for all $n \geq n_\omega$*

$$||A^n(\omega(n))|| \leq C\sqrt{n \log n}.$$

We prove the following version of this Theorem:

Theorem 24 *Let A be a computable countable matrix over $\{0,1\}$. There is a universal constant $C > 0$ such that, for every random ω with $c(\omega) = c$ we can find effectively in c, an n_ω, such that, for all $n \geq n_\omega$*

$$||A^n(\omega(n))|| \leq C\sqrt{n \log n}.$$

We use the following

Lemma 25 [Fouché 2] *Let M be an $n \times n$ matrix and let X be a random column vector over $\{-1,1\}$ taking each of the values -1 and 1 with probability $1/2$. Then*

$$P\{\exists_{2 \leq k \leq n} |M^k \bar{X}(k)| > \sqrt{Dk \log k}\} \leq 2\sum_{k \geq 2} \frac{k^2}{k^{D/2}}.$$

Proof of theorem 24. Choose $D > 4$, then $2\sum_{k\geq 2} \frac{k^2}{k^{D/2}}$ is a computable real and the result follows from Theorem 9. ■

So even though we clearly can, given a random vector ω, tailor-make initial segments A^n of A to render the discrepancy of the set system given by A^n very high, the prefix complexity of ω and the recursiveness of A rules out the discrepancy being high infinitely often.

Chapter 3

An application to descriptive set theory

In this chapter, we make full use of Theorem 9 to deduce some results in descriptive set theory. The treatment here can be seen as the general case of the examples of the previous chapter. We also prove some results relating Theorem 9 to results obtained by W.L. Fouché (see bibliography). We then take the opportunity to remark on randomness and extensionality. Finally, we examine various possible generalizations of Theorem 9, all of which fail.

3.1 Recursive events in random sequences

Let $\mathbf{P}$ be a computable predicate defined for all finite binary strings. We can see $\mathbf{P}$ as an event which may or may not occur in an infinite binary sequence ω. We are interested in the following question: *Does* $\mathbf{P}$ *occur in* ω*? If so when?*

Of course, this class of questions is not solvable. Even the following class:

$$\text{Given } \omega, \text{ does } \mathbf{P} \text{ hold for } \omega?$$

is not solvable in $\mathbf{P}$ and ω.

Indeed, if $\mathbf{P}$ is a computable predicate, the set of reals $S_{\mathbf{P}} = \{\omega | \exists n \mathbf{P}(\omega_{1:n})\}$ is called a *semirecursive* set of reals partly because the membership problem for the set is analogous to that of the membership problem for finite sequences in computably enumerable (or semirecursive) sets of integers.

Thus, to decide whether $\omega \in S_{\mathbf{P}}$, we can in general do no better than to wait for (an initial segment of) ω to appear in the enumeration of strings x for which $\mathbf{P}(x)$ holds. If this happens then of course $\omega \in S_{\mathbf{P}}$, if not, we may be forced to wait forever.

Now many such computable events $\mathbf{P}$ occur with probability a computable real. Let $\mathcal{C}$ denote this subclass. That is $\mathcal{C} = \{\mathbf{P}|\mu(S_{\mathbf{P}})$ is computable$\}$. So consider the following class of questions : *Given ω, does $\mathbf{P} \in C$ hold for ω?*

It follows directly from Theorem 9 that this class of questions **is solvable in P,** $\mu(S_{\mathbf{P}})$ and ω for every random ω. In particular:
Given an arbitrary binary real ω. If ω is random, there is a computable function which takes as input (any) indices for $\mathbf{P} \in \mathcal{C}$ and $\mu(S_{\mathbf{P}})$ and outputs an $n \in \mathbf{N}$ such that $\mathbf{P}$ occurs within the first n digits of ω, or not in ω at all.

We will call any information of the form "$\mathbf{P}$ occurs in $\omega_{m:n}$", *local* information about ω, in order to contrast it with that of the form "$\omega \in S_{\mathbf{P}}$" which tells us nothing about *where* $\mathbf{P}$ occurs.

As a direct consequence of Theorem 9, we get:

Theorem 26 *Let $\mathbf{P}$ be a computable predicate with $\mu\{\omega|\exists n\mathbf{P}(\omega_{1:n})\}$, a computable real π. Then, given $c \in \mathbf{N}$, there is an $n \in \mathbf{N}$ such that, for all $\omega \in K^c$, $\mathbf{P}$ holds for some initial segment of ω shorter than n, or not at all. Further, there is an algorithm finding n on input indices for $c, \mathbf{P}$ and π.*

Minor modifications of the proof give the following version of Theorem 9:

Corollary 27 *Let S be a computably enumerable set of binary reals with $\mu(S)$ computable and let p be a program enumerating S. Then, given $c \in \mathbf{N}$, p and a program for $\mu(S)$, we can effectively find a stage $t(c)$ in the enumeration after which no $\omega \in K^c$ can appear for the first time. That is, ω must appear before $t(c)$ or not at all.*

This form of Theorem 9 will be useful later. In terms of individual random sequences we can state the following:

Corollary 28 *Let ω be a random sequence. Then there is an algorithm, easily constructible from $c(\omega)$, which on input of programs/indices for $\mathbf{P} \in \mathcal{C}$ and $\mu(S_{\mathbf{P}})$, outputs an n such that $\mathbf{P}$ either occurs before the nth digit of ω or not at all.*

Note: This algorithm clearly requires no access to the digits of ω. *If* one has access to an oracle for ω, the question as to whether $\mathbf{P}$ would ever occur is answerable in finite time for each $\mathbf{P} \in \mathcal{C}$.

Remark 29 Andreas Blass has given an alternative proof of the following form of Theorem 9 (e-mail correspondence): *For almost all binary sequences there is an algorithm which can decide the membership problem for all $S_{\mathbf{P}}$ for which $\mu(S_{\mathbf{P}})$ is computable, when given indices for $\mathbf{P}$ and $\mu(S_{\mathbf{P}})$.*

Proof. For this proof consider any program p_i enumerating those strings for which the predicate $\mathbf{P}_i$ holds, and consider those infinite sequences which appear in the enumeration of intervals (binary strings for which $\mathbf{P}_i$ holds) only after we are closer than 2^{-p_i} to $\mu(S_{\mathbf{P}_i})$.

Since $\sum 2^{-p_i}$ converges, it follows from the Borel-Cantelli lemma that only measure 0 of infinite binary sequences can appear this late for infinitely many predicates $\mathbf{P}_i$. For each ω in the measure one complement of this set, we can therefore keep a list of each of the finitely many $\mathbf{P}_i$'s for which ω appears this late. Given any $\mathbf{P}_i \in \mathcal{C}$ we can then consult our list and if $\mathbf{P}_i$ is not on the list we can start enumerating intervals until our element is listed or we get closer than 2^{-p_i} to $\mu(S_{\mathbf{P}_i})$. ∎

Note that one can frame this proof of Blass in terms of Solovay's criterion for infinite sequences, namely, define the sequence of sets $S_1, S_2, ...$ with S_1 the set of sequences enumerated after we are closer than 2^{-1} to $\mu(S_{\mathbf{P}_1})$ and S_2 the set of sequences which, *in addition*, are enumerated after we are closer than 2^{-2} to $\mu(S_{\mathbf{P}_2})$ and so on. Since $\sum \mu(S_{\mathbf{p}_i}) < \infty$ we have, by Solovay's criterion, that no random sequence can be in infinitely many of the sets $S_{\mathbf{p}_i}$. Blass has also remarked that $\mu(S_{\mathbf{p}_i})$ need not be recursive, but that it suffices to have an index for computing a sequence of rationals converging to $\mu(S_{\mathbf{P}_i})$ from above. This turns out however, not to be a generalization for our particular form of the result by the following converse to Theorem 9:

Proposition 30 *Let* $\mathbf{P}$ *be a computable predicate. If there is an algorithm which, given* $c \in \mathbf{N}$, *finds an* n *such that for all* $\omega \in K^c$, $\mathbf{P}$ *occurs before the* n*th digit of* ω *or not at all, then* $\mu(S_\mathbf{P})$ *is a computable real.*

Proof. By hypothesis, we can find for each c a length $n(c)$ such that all enumerated strings of greater length are compressible by more than c. As the total measure of such sequences is less than $k \cdot 2^{-c}$ (see e.g. [Li and Vitányi]), the cumulative measure at this stage must be closer than $k \cdot 2^{-c}$ to the total measure. In other words, we can on input c, compute $\mu(S_\mathbf{P})$ to an accuracy within $k \cdot 2^{-c}$, therefore $\mu(S_\mathbf{P})$ is a computable real. ■

3.1.1 Probability laws

In [Fouché 1], Fouché proves that every random sequence is contained in every recursively Π_1^0 set of measure one of binary reals. This is done by showing that we can consider the complement of any computably enumerable set of measure one, as a *Martin-Löf test*, hence every random ω must drop out of the complement at some stage. Since a Π_1^0 set of measure one is the union of a countable number of computably enumerable sets of measure one, the result follows. This result is used extensively in the papers [Fouché 1],[Fouché 3],[Fouché and Potgieter] to examine the properties of random sequences when seen as codes for combinatorial objects, some examples of which we discussed in the previous chapter.

The method in the proof of Theorem 9 leads to a stronger result than in [Fouché 1]:

Theorem 31 *Let* (O_i) *be a c.e. infinite sequence of sets of intervals with the property that* $\sum \mu(O_i) = 1$. *Let* p_1 *be a program enumerating the sequence* (O_i). *Then there is (effectively) a universal constant* k *such that every* $\omega \in K^c$ *must appear before the measure of the enumerated intervals differs from* 1 *by less than* $2^{-(|p_1|+c(\omega)+2\log c(\omega)+k)}$.

Proof. Consider any binary real ω which has not appeared in the enumeration by this time. Any such sequence has an initial segment specifiable as part of the complement of the enumerated strings of length l, say.

We can therefore specify any such initial segment using p_1, c as well as its position in this complement. Since the complement has Lebesgue measure less than $2^{-(|p_1|+c(\omega)+2\log c(\omega)+k)}$ we need at most

$$\begin{aligned} & |p_1| + 2\log c + l - (|p_1| + c + 2\log c) \\ = \; & l - c \text{ digits.} \end{aligned}$$

Hence some initial segment $\omega_{1:l}$ of ω is compressible by at least c. The value of k can be found in an analogous way to that of Theorem 9. Therefore, for each computable event $\mathbf{P}$ which occurs with probability one and each $c \in \mathbf{N}$, we can find a length n such that $\mathbf{P}$ must occur in $\omega_{1:n}$ in all $\omega \in K^c$. ■

Consequently, not only must every probability law $\mathbf{P}$ hold, but we can also find, in terms of the compressibility, upper bounds on the waiting time for each event $\mathbf{P}$ in the Π_1^0 set. Our results on the law of large numbers and the law of the iterated logarithm are examples of this.

Let us call the criterion of being in every semirecursive set of measure one, *Fouché's criterion* for random sequences.

For certain simple predicates $\mathbf{P}$ we can see directly that the prefix complexity of a segment must be low if $\mathbf{P}$ does not occur for a long time. One example is $\{x \in S_{\mathbf{P}} \leftrightarrow x \text{ contains a } 0\}$. Given any c, we could easily find an m such that no segment $\omega_{1:m}$ of an $\omega \in K^c$ can consist only of 1s. This is because the probability that the initial segment contains no 0s is easy to find and decreases rapidly with increasing m.

However, many more complicated predicates will not lend themselves to this type of treatment. The advantage of Theorem 9 is the idea that strings which have not yet satisfied $\mathbf{P}$ at a late stage in the enumeration, are sandwiched between the total measure and the measure of the listed segments which is converging to a computable real.

3.1.2 Does Fouché's criterion characterize the random sequences?

In other words: Let ω be an infinite sequence appearing in all computably enumerable sets of intervals with Lebesgue measure one. Must ω be random? The answer is no.

Theorem 32 *There is a nonrandom sequence ω which is contained in all computably enumerable sets of measure one.*

Proof. Let $\{\mathbf{P}_i\}$ be the entire sequence of predicates which hold for Lebesgue measure one of all infinite binary sequences. Let p_i be a shortest program for $\mathbf{P}_i$ (this is clearly not an enumerable list!). We will now construct a non-random sequence ω which satisfies all the predicates $\mathbf{P}_i$:

We construct the sequence in stages, starting with the segment s_1 consisting of $2|p_1|$ 0s. Now enumerate those finite strings satisfying $\mathbf{P}_1$ until some extension e_1 of s_1 appears. Extend s_1 to e_1 and append $2|p_2|$ 0s to e_1 to obtain a string s_2 and iterate the process. That is, we enumerate those strings satisfying the predicate $\mathbf{P}_2$ until some extension e_2 of s_2 appears. Then extend s_2 to e_2. We repeat this process for each $\mathbf{P}_i$.

Clearly, since each of the sets has measure one, some extension of s_i must appear in the enumeration of strings satisfying $\mathbf{P}_{i+1}$. Hence ω has an initial segment s_i in each of the computably enumerable sets.

It is furthermore clear that the compressibility of ω is unbounded since we need only around $|p_1| + ... + |p_i|$ digits to obtain $\omega_{1:2(|p_1|+..+|p_i|)}$. ■

3.1.3 Early appearances in Fouché's criterion characterize the random sequences

Theorem 9 implies that a random sequence should appear *early* (with a short initial segment) relative to the length of each associated predicate $\mathbf{P}_i$, or not at all. For measure one, it must appear (early) in all the enumerable sets of measure. We are led to ask: If a sequence appears early with regard to the length of predicates of measure one, is it random? The answer is yes.

Theorem 33 *Any infinite sequence ω which satisfies each effective probability law $\mathbf{P}_i$ early, is random.*

Proof. We prove the contrapositive, that a non-random sequence appears *late* in certain enumerations of measure one. Particularly, ω does not satisfy any of the allowed appearance schedules in Theorem 9.

Let ω therefore be non-random. Hence there exists, for each $c \in \mathbf{N}$, an initial segment of length $n(c)$ such that $K(\omega_{1:n(c)}) < n(c) - c$. In particular, there is a program p_c of length $n - c$ which outputs the initial segment of ω of length n.

We associate with each such program p_c a program p'_c (with which we can associate a predicate $\mathbf{P}_c$ holding with probability one): The program p'_i enumerates intervals as follows: p'_i first uses p_c to obtain (but not yet enumerate) $\omega_{1:n}$. The program p'_i then enumerates all other sequences of length n and only then enumerates $\omega_{1:n}$.

Now p'_c clearly need have length l no more than $|p_c| + k = n - c + k$ with k independent of c. Clearly l drops arbitrarily far below n. Therefore ω appears in the enumeration only after we are closer than 2^{-n} to 1. All $\omega \in K_r$ would have had to appear before we were closer than 2^{-n+c-r} to one. Since $n - c + k$ drops arbitrarily far below n for increasing c, none of the schedules in Theorem 9 can be met. ∎

3.2 Gambling against a random sequence

It easily follows from the method of Theorem 9, that for any given random ω and events $\mathbf{P}$ which occur infinitely often with probability one (such as the occurrence of some fixed subsequence), $c(\omega)$ will enable us to find, for each m, an m' such that $\mathbf{P}$ must occur somewhere in $\omega_{m:m'}$.

This means that there exists, for each random infinite sequence, a gambling strategy which enables us to win an unlimited amount of money by gambling against the sequence. Our only requirements are $c(\omega)$ and a generous banker/benefactor who is willing to repeatedly loan us arbitrarily large sums of money. Note that we can *guarantee* a payback time at each time of borrowing.

Indeed, imagining the process as a coin toss, we can use $c(\omega)$ to find at the mth toss, an m', then borrow $2^{m'-m}$ units, betting repeatedly on "heads", starting our bet at 1 unit and doubling it at each toss until "heads" appears - this is sure to happen before the m'th toss. We can repeat this process indefinitely.

It is somewhat ironic that it is the very randomness of the sequence which guarantees the existence of the strategy.

3.3 An application to Chaitin's Ω number

Clearly, finding $c(\omega)$, even given an oracle for a random sequence ω, is in general highly non-effective since, intuitively, we would need to use an oracle for the halting problem on each of the infinitely many initial segments of ω.

However, there is an important special case in which we can effectively find an upper bound for $c(\omega)$ without having even an oracle for ω. This is the case $\omega = \Omega$. Indeed, it follows directly from Theorem 8 that $|p|$ is in fact find an upper bound for $c(\Omega)$.

Since we can easily write p for any reasonable universal machine U and we need no knowledge of the digit values of Ω, we can *effectively* find a c such that $\Omega \in K^c$.

So although we are denied any knowledge of individual digits in the sequence Ω (see [Chaitin 4]), we have access to an infinite amount of local information about Ω. For example, we can find:

(i) an $n \in \mathbf{N}$ such that $\omega_m = 1$ for some value $m \leq n$,

(ii) an $n \in \mathbf{N}$ such that for some $m \leq n$, the number of 1s in $\omega_{1:m}$ is exactly $\frac{m}{2}$,

(iii) for each k, a value for n such that there is a run of k consecutive 0s in $\omega_{1:n}$,

(iv) for each m and word s in $\{0,1\}^k$, a value m' such that the word s appears in $\omega_{m:m'}$.

In [Chaitin 4], a Diophantine equation is constructed which has finitely many solutions if the nth bit of Ω is a 0 and infinitely many if the nth bit is a 1. Similar considerations to those above hold for this equation.

The halting probability Ω has widely been seen as a striking example of a completely random object which arises naturally in mathematics, but about which we can probably not know much more than that Ω must possess all statistical properties, such as being normal and satisfying the law of large numbers, these properties following from the fact that Ω is random. (see [Bennet and Gardner],[Chaitin 4],[Li and Vitányi],[Rosenberg and Salomaa]).

It is therefore perhaps unexpected, that we can deduce, in a uniform and completely effective way, an infinite amount of local information about Ω.

3.4 Discussion : Randomness and extensionality

Theorem 9 shows that certain aspects of random sequences are predictable. In fact, it should be clear from the proof of Theorem 9, that for each random ω, there is a *canonical* algorithm (incorporating the compressibility-coefficient of ω) which on input (any program (index) for) a computable predicate **P** holding with probability one, gives as output an upper bound for the occurrence of the event described by **P**. This stands in contrast to the fact that for each random ω there is *no* algorithm that can successfully predict digit values.

Note that it is easy to see that a much weaker version of Theorem 9 holds, namely that random binary sequences exist for which we can know that each of an infinite sequence of predicates hold. For example: find an n_1 such that more than measure $1-k$ contain a 1 before n_1, now find an n_2 such that more than measure $1-k_2$ contain a 1 between n_1 and n_2, and so on, with n_i chosen in such a way that $(1-k)(1-k_2)...$ converges to a number larger than 0. Then this set certainly contains some random sequences (since the set of random sequences has measure one) and for these random sequences we can make the set of predictions "a 1 appears between each n_i and n_{i+1}". *But we cannot do this for all computable predicates since the set of computable predicates is not a computable set. So we cannot "run through" all possibilities.*

The type of predictability discussed in the first paragraph certainly seems to rule out requiring that a random string has a particular compressibility-coefficient, such as in statements of the form: "Let ω be random with compressibility 1".

Further, since each $\omega \in \mathcal{K}$ is considered random, we should be able to think of each such ω as an outcome of a coin toss. In particular, taking initial segments of increasing length of an $\omega \in \mathcal{K}$ should reflect our intuitions regarding a developing coin toss. It would however be completely contrary to the spirit of a real coin toss to think of an *a priori* fixed constant c which fixed upper bounds for all computable events which have computable probability of occurring in the coin toss. This is because we feel that, whatever finite information we are given *now*, a real coin toss will develop *independently* of it. This is closely associated to the idea that a real coin toss is at every stage *free*.

Now the point is not that the "real" random sequences are some other *extensional* set, but that the problem may be the extensional viewpoint itself which treats infinite sequences as completed objects and allows us to talk about global properties of sequences (such as $c(\omega)$). Not surprisingly then, expecting a sequence to conform to some a priori compressibility-coefficient seems to impinge on the independence and freedom of the sequence when we again view it as *developing*. Since $c(\omega)$ is not uniquely determined by any finite initial segment, one must question whether talking about *global* constants such as $c(\omega)$ makes sense in the context of random sequences. It is perhaps particularly striking that $c(\omega)$, which is a type of certificate of incompressibility intended to *guarantee* randomness, can be used to the opposite end, namely to predict aspects of the sequence!

3.5 Various generalizations fail

3.5.1 Two obvious generalizations fail

We show in this section that some more or less plausible generalizations of Theorem 9 fail. We firstly show that there are (non-random) *binary sequences* for which Theorem 9 fails. For this section we use the following notation: We say that ω is c-compressible *before* m if $K(\omega_{1:n}) \leq n-c$ for some $n < m$, that ω is c-compressible *after* m if $K(\omega_{1:n}) \leq n-c$ for some $n > m$ and that ω is c-compressible *only after* m if $K(\omega_{1:n}) \leq n-c$ for some $n > m$ but for no $n \leq m$. We use the same notation for strings. We firstly give an example of a (non-random) infinite binary sequence for which Theorem 9 fails:

Example 34 Let ω be the infinite binary sequence which is defined as follows: Let U be any universal machine, not necessarily prefix. Now dovetail the computations of all programs p for U and record the halting inputs in a binary string, as follows: For each step i of the dovetailed computation, if no halting state is reached at step i, add a 0 to the binary string. If a halting state is reached at step i, (program p halts, say) then append to the string a 0 followed by p consecutive 1's followed by a 0. It is clear that ω will then contain a string p' if and only if p' halts. Note that a sequence contains each string of this form with probability one. Any algorithm which could give an upper bound on the waiting time for each event $\mathbf{P}$, would solve the halting problem for U. This gives a contradiction.

Secondly, we present an example of a *computably enumerable set* (with non-computable measure) for which Theorem 9 fails.

Example 35 Let U be a universal prefix machine and consider the sequence of halting programs $p_1, p_2...$ and the associated computably enumerable set $S = \bigcup_{i=1}^{\infty} 0.p_i$. Now, if we could effectively find for each c a stage $t(c)$ such that no $\omega \in K^c$ could appear for the first time after

$t(c)$, then Ω_U would be a computable real by reasoning similar to that in Proposition 30. But not only is Ω_U not computable, it is in fact itself random. This gives a contradiction.

The above is perhaps not surprising, since $\mu(S_{\mathbf{P}})$ played a crucial role in the proof of Theorem 9, and in the above example we have no access to the non-computable $\mu(S_{\mathbf{P}})$. Therefore a more natural generalization would be the following

Oracle form of Theorem 9: Can we, given c, a program for a computable predicate $\mathbf{P}$ and an *oracle* for $\mu(S_{\mathbf{P}})$, find an $n(c)$ such that, for all $\omega \in K^c, \mathbf{P}$ must hold in $\omega_{1:n(c)}$ or not at all?

There are other reasons we may expect this to hold. For U a non-prefix universal machine, the corresponding compressibility of elements in any computably enumerable set with finite Lebesgue measure must increase, and since Kolmogorov complexity and prefix complexity are asymptotically equal $(K(x) < C(x) + 2\log C(x))$, one could expect that the same may hold for prefix complexity.

We can prove, however, that the oracle form also fails by showing that there exists computably enumerable sets of intervals for which upper bounds on the waiting times for the appearance of $\omega \in K^c$ need not even exist. This also shows that neither the criterion of Solovay or Martin-Löf give explicit appearance schedules for $\omega \in K^c$ in terms of c. The reader will undoubtedly not be interested in the fact that the result below took the author eighteen months to prove.

3.5.2 The oracle form fails

Corollary 20 implies the following: For each computable enumeration of intervals with total measure a computable real, there exists for each $c \in \mathbf{N}$, a stage t_c such that any real listed for the first time after stage t_c, must be at least c-compressible. *Our counter example will be the enumeration of strings which are themselves 1-compressible.*

If the oracle form is to hold for our (counter)example, we should have for each c, a stage t_c, such that each ω enumerated in the set of 1-compressible sequences after stage t_c, must in fact be c-compressible too. For large c this seems unlikely as we feel that we should be able to modify an ω with $c(\omega) = 1$, at *arbitrarily large* n in such a way that $K(\omega_{1:n}) < n - 1$ *for the first time.*

Let $n(t_c)$ be the greatest string length up to stage t_c. Let S_1 be the set of sequences which are 1-compressible only after $n(t_c)$ and let S_c be the set of sequences which are c-compressible only after $n(t_c)$. We will show that $\mu(S_1) > \mu(S_c)$ for c large enough. This will suffice to show that the oracle form fails.

An upper bound for $\mu(S_c)$ follows from the following well known lemma (see e.g. [Li and Vitányi]).

Lemma 36 *There is a constant k_1 such that for each n, the number of sequences x of length n for which $K(x) \leq n + K(n) - r$, is at most $k_1 \cdot 2^{n-r}$. Hence $\mu(S_c) \leq k_1 \cdot \Sigma_{n=n(t_c)}^{\infty} 2^{-K(n)-c}$.*

We now find a lower bound for the measure of S_c.

Lemma 37 *There is a constant k_2, such that for each m, the measure of binary sequences ω, which are 1-compressible only after m, is larger than $k_2 \cdot \Sigma_{n=m}^{\infty} 2^{-K(n)}$. Hence*

$$\mu(S_1) \geq k_2 \cdot \Sigma_{n=n(t_c)}^{\infty} 2^{-K(n)}.$$

Proof. Since fewer than $k_1 \cdot 2^{n-K(n)-c}$ strings of length n have prefix complexity lower than $n - c$ and $\sum_{n=0}^{\infty} 2^{-K(n)} < 1$, more than measure $\frac{k_1}{2}$ of infinite binary sequences are not 1-compressible (at all). Hence, for each m, more than measure $\frac{k_1}{2}$ of binary strings of length m, are not 1-compressible *before* m. Let N_m be the set of strings of length m which are not 1-compressible before m. Now each $x \in N_m$ has a program p_x which outputs x and has length $|x| + l$, where $l \leq K(m)$ (see e.g.[Li and Vitányi]). We will now take each of these strings x and extend them to 1-compressible strings x_e.

Consider the prefix program p (of length r say) which on input p_x, outputs x followed by a 1 and a string of 0s up to length $|p_x| + r + 1$. This extension x_e of x of length $|p_x| + r + 1$ has $K(x_e) \leq |p_x| + r$ and is thus (at least) 1-compressible.

Hence all infinite extensions of x_e are at least 1-compressible. Now, since $|p_x| \leq |x| + K(m)$, it will suffice to add a 1 and $K(m) + r$ zeros to every string in N_m. Each of *these* strings of length $m + K(m) + r + 1$ will then have prefix complexity at most $m + K(m) + r$ and all *infinite* extensions of these strings will be 1-compressible (only) after m. We do this for every set $N_n, n > m$ and denote the set obtained from N_m by N'_m.

We now show that the sets N'_i are disjoint. There are only two possibilities for sets N'_h and N'_j with $h < j$. Either $j \le h + K(h) + r + 1$ and we have the following (note carefully the (non)overlap of the 10......0 sequences):

$$\begin{array}{l} \alpha_1......\alpha_h \overbrace{10......0}... \\ \beta_1............\beta_j \overbrace{10.........0}... \end{array}$$

with the sets disjoint because of the 1 preceding the 0 sequence in $\omega \in N'_j$, or we have $j > h + K(h) + r + 1$ as follows:

$$\begin{array}{l} \alpha_1...\alpha_h \overbrace{10...0}... \\ \beta_1.....................\beta_j \overbrace{10...0}... \end{array}$$

in which case each $\omega \in N'_h$ is 1-compressible before j and can therefore not be in N'_j. There is therefore a constant k_2 such that $\mu(\bigcup_{n=m}^{\infty} N'_n) \ge k_2 \cdot \sum_{n=m}^{\infty} 2^{-K(n)}$ for each m. ■

Hence $\mu(S_1) > \mu(S_c)$ for c large enough, and the oracle form fails.

Note that this also implies that there are Martin-Löf (and Solovay) tests for which, given c, no stage exists after which no $\omega \in K^c$ can appear. Indeed, consider the result above and take O_i as the i-compressible sequences. Then $\mu(O_i) < 2^{-i}$ and $O_i \supseteq O_{i+1}$.

Chapter 4

Complexity and unsolvability

4.1 Introduction

In this chapter we examine the relation between complexity and unsolvability. To demonstrate the usefulness of (Kolmogorov) complexity in studying unsolvability, we start the chapter by using complexity to give a proof of Rice's theorem. We find that complexity gives us more information than the classical proofs of Rice's theorem (via an imbedding into the halting problem or via the recursion theorem). Since complexity is a finitary concept (as opposed to unsolvability, which is strictly infinitary) we can (and do) examine the relation between the unsolvability of a set and the complexity of *finite subsets* of the set. We also prove a "gap-theorem" for outputting consecutive integers. We then give some examples of problems which are easy to show unsolvable via complexity but are (at least) awkward to show unsolvable by an imbedding into the halting problem. We also examine the relation between our method and Turing completeness. Finally we deduce some easy results relating to proof lengths and independent statements. Unless explicitly stated, we work in the context of Kolmogorov (not prefix -) complexity. We will often refer to initial segments of sets and mean by this initial segments of the associated characteristic functions.

4.2 Rice's theorem

Notation: We will denote universal (additively optimal) Turing machines by the notation U and denote by p_n the program for U with index n. We will denote various classes of programs by C, and by C_n, the members of C up to length n. We will denote the special class of *all* programs by P, with P_n the obvious set. A set $X \subseteq \mathbf{N}$ is *decidable* or *solvable* if there is an algorithm ϕ which can decide, for all $x \in \mathbf{N}$, on input x, whether or not $x \in X$.

Let F be any class of partial computable functions and let C^F be the class of programs p, such that p computes some function in class F. Rice's theorem states that $\{n : p_n \in C^F\}$ is a computable set if and only if $C^F = \emptyset$ or C^F is the class of all partial computable functions.
In other words we cannot recursively sort *programs* according to the *functions* they compute. Rice's theorem is generally proved in one of two ways, either the problem of classifying the programs is reduced to the *halting problem*, or the *recursion theorem* is used to show that no algorithm exists which could sort the programs correctly. See for example [Rosenberg and Salomaa].

We give a proof of Rice's theorem using the idea of "shortest program for a function". For this, we extend the idea of Kolmogorov complexity (of strings) to functions. The method is simple and can be used in a variety of situations.

Theorem 38 *Let $\emptyset$ be the nowhere defined function and let C be any class of functions not containing $\emptyset$. Then $\{n : p_n \in C\}$ is not a computable set.*

In other words, the nowhere defined function is recursively inseparable from any other class of functions. The crucial part of the proof will be that every program in C halts on some input, which allows us to diagonalize against C.

Proof. Suppose by contradiction that there is a program p which can decide on input the index i of a program p_i, whether or not p_i is in C. We can then construct the following sequence of programs (p'_n) : On input any integer i, p'_n first uses p to enumerate all programs in C_n (there is an effective mapping from a program to its index so p can do this). Let p_c be the *first* program enumerated in C_n. For n small, C_n may be empty, we assume n large enough that this is not the case.

After p'_n has used p to find all programs up to length n which are elements of C, it dovetails the computation steps for each of the programs in C_n on all possible inputs until each of them has halted on some input. This is certain to happen as none of them compute the nowhere defined function. Let the total number of steps used in this dovetailing be t_n. The program p'_n then loops for *another* t_n steps, and then *simulates* p_c *on input* i.

Since the length of p'_n need be no longer than $|p| + \log n + k$, and p'_n computes the same function as p_c, we must have that p'_n is in C_n.

The program p'_n however, differs from all the programs in C_n since, for all $p \in C(n)$, there exists an i such that p halts on i within $2t_n$ steps, while p'_n runs longer than $2t_n$ steps on any input. We obtain a contradiction. ■

Corollary 39 *Rice's theorem.*

Proof. Any classification of programs according to the functions they compute must place all the programs computing the nowhere defined function in the same class. We can then diagonalize against the programs in the other class, as above. If neither class contains the nowhere defined function we can diagonalize against either class. ■

Henceforth we will refer to the "new" program we construct, given a class of programs we assume by way of contradiction, to be computable, as *the diagonal program.*

4.3 Complexity of finite fragments of uncomputable sets

Usually, classes of programs are shown to be uncomputable by reducing them to the halting problem. This method of course only applies when we are dealing with an infinite class. A moment's thought on the construction above leads us to the observation that if we are classifying programs, and the complexity of the classification increases with increasing program length, the classification should be uncomputable. In this sense we can see complexity as a cause of unsolvability. In Section 4.5 we show, however, that initial segments of *solvable* sets can also be maximally complex.

In this section we take a complementary perspective and show that it is a property of many unsolvable problems (or sets) that each initial segment of the set has *maximal* complexity, given the fact that it is c.e.. It follows from the proof of Theorem 2.7.2 in [Li and Vitányi], that the complexity of the characteristic function χ of the halting set has $C(\chi_{1:n}) > \log n$ for all n. This implies that the complexity of the number of halting programs up to length n is $> n - c$ for all n and universal c. Since the proof of Theorem 38 works for any index complete set, we have the following generalization of this result:

Theorem 40 *Let C^1 and C^2 be two disjoint classes of partial computable functions and let p'_n be a program which correctly classifies all programs up to length n as in C^1 or in C^2. Then*

$$C(p'_n) \geq n - c,$$

where c depends on C^1 and C^2 only. In particular, c depends only on the lengths of the programs used to enumerate C^1 and C^2.

Proof. Let p be a program which uses p'_n to diagonalize against the programs in C^1_n or C^2_n. Since p can get n from p'_n, p need have length no greater than $|p'_n| + k$, k independent of n. Also, since p must differ from all programs in C^1_n (or C^2_n), p must have length $> n$. Hence $C(p'_n) + k > n$ thus $C(p'_n) > n - k$. ■

Corollary 41 *Given a program p which supposedly recursively divides the set of all programs into two function classes, we can effectively find a length $n(|p|)$ such that p incorrectly classifies some program of length $< n$.*

In general, let $\mathbf{P}$ be some computable property which we suspect most halting programs have. Let p be a shortest program for testing programs for property $\mathbf{P}$. Then at least $2^{n-\log n-|p|-O(1)}$ of strings of length n that halt, do not have property $\mathbf{P}$. In some sense then, halting programs seem to have very little in common except the fact that they halt.

Restricting the halting problem to programs shorter than n, is a special case:

Corollary 42 *There exists a constant c such that any program p'_n which correctly classifies the programs in P_n as halting or not (on the empty input), must have length at least $n - c$.*

This means that the halting problem for programs shorter than n is in some sense as complicated as possible. Indeed, it will suffice to solve the halting problem for P_n to give the *exact number* of the programs in P_n which do halt (we then dovetail the running of these programs until this number have halted - we then know no more will halt). We therefore need *at most* $n+1$ digits to specify the number of these halting programs since the number of programs (binary strings) of length at most n is $< 2^{n+1}$ and we therefore need at most $n+1$ digits to specify this number. Further, by the above result we also need *at least* $n-c$ digits. This enforces the idea that there is very little pattern to the halting set. From the fact that we can construct p'_n from the exact number of halting programs of length at most n, we get the following:

Corollary 43 *The number h_n of (non) halting programs in P_n, has Kolmogorov complexity at least $n-c$.*

Proof. Since we can construct p'_n from this number h_n, i.e. $n-c < C(p'_n) < C(h_n) + c_1$, we must have that $C(h_n) > n - c - c_1$. ∎

This leads immediately to

Corollary 44 *The number h_n of (non) halting programs in P_n is at least 2^{n-c}.*

Proof. Since $C(h_n) > n-c$, it follows that the number h_n must have at least $n-c-c_2$ digits (c_2 universal), and hence it must be larger than 2^{n-c-c_2}. ∎

We now show, analogous to the fact that there must be many (non)halting programs in P_n, for each n, that many computably enumerable properties must be well represented amongst programs up to length n. This is because, as in the halting set, we can use the exact number of programs up to length n with a particular property, to find all such programs up to length n. If there were few such programs, we could use the small number to diagonalize against the set of programs as in our main proof, giving a contradiction. The only way around this is that the number of such programs is maximally complex and hence of length around n.

Theorem 45 *Let* $\mathbf{P}$ *be a c.e.* functional *property of programs enumerated by a program p and let $|p| = c$. If both* $\mathbf{P}$ *and not-*$\mathbf{P}$ *hold for at least one program each, the number x_n of programs in P_n which have/don't have the property* $\mathbf{P}$ *is at least $2^{n-c-c_1}, c_1$ independent of n.*

Proof. Let p'_n output x_n. We can then use p'_n to construct a diagonal program p with property $\mathbf{P}$. Since we can get n from p'_n, p need have length no more than $|p'_n| + k$, k independent of n. Since p must differ from all programs in P_n, p must have length $> n$. Hence $|p'_n| + k > n$ or $|p'_n| > n - k$. Since a candidate for p'_n is just x_n itself, we must have that $C(x_n) > 2^{n-c}$. ∎

Corollary 46 *Let* **P** *be a c.e.* functional *property of programs with associated program p with $|p| = c$. If both* **P** *and not-***P** *each hold for at least one program, the number of programs x of length* exactly n *which have/don't have the property* **P** *is at least 2^{n-c-c_1}, c_1 independent of n.*

Proof. All we need do is change **P** to **P** $'$ where **P** $'$ is: "**P** holds and the program is of length n". The rest of the proof is the same except that we may have to pad the diagonal program up to length n. Since $\log n$ becomes arbitrarily smaller than n, we will always have space to start the program with a string which says "Ignore until digit m" for suitable m. ∎

4.4 Convergence of the halting fractions?

It is clearly unlikely that the sequence of halting fractions $(\frac{|\{i : p_i \text{ halts } p_i \in P_n\}|}{2^n})_n$ converges. Whether or not this is the case, the sequence cannot converge to any *computable real* and in fact, no computable real can be a *limit point* for this sequence of fractions. This is because each such fraction must have maximal complexity and hence cannot even start with a simple segment (for example, a long initial segment of a computable real) since this would mean that an initial segment (with length of very low complexity) of some fraction for n large, would be easy to specify and would therefore diminish the complexity of the entire fraction.

4.5 Complexity of finite fragments of computable sets

In this section we apply complexity to finite subsets of *computable* sets and give an example of a trivially computable set for which *each initial segment* has maximal complexity.

Consider the set of natural numbers which are outputs of programs. This is clearly **N**, a computable set. We find, however, that restricting the set to outputs of programs in P_n, gives a finite subset of **N** with maximal complexity.

Theorem 47 *Let O_n be the set of outputs on the empty input, over all programs in P_n. That is, s is in O_n if a program in P_n outputs s on the empty input. Then O_n has Kolmogorov complexity larger than $n - c$ for c universal.*

Proof. Similar to the proofs above. ∎

It follows that the number of different such outputs must have complexity around $n - c$ and hence be at least 2^{n-c}. It also follows that the number of different *functions* computed by programs in P_n is large, i.e. around 2^{n-c}.

4.6 A gap theorem for outputting consecutive integers

We show that there are consecutive integers m and $m+1$ such that the shortest running time of any program in P_n to output $m+1$, is enormously longer than the shortest running time of any program in P_n to output m.

Theorem 48 *For all n large enough the following holds, for c universal. Let $t(m)$ be the shortest running time to output m over all the programs in P_n. Then there are integers m and $m+1$, both outputs of programs in P_n such that*

$$t(m+1) - t(m) > s \text{ for all } s \text{ with } C(s) < n - c.$$

Proof. Let x_n be any number larger than the largest time-gap between the first outputs of consecutive integers. In other words, if any program in P_n outputs an integer s in t steps and $s+1$ is not outputted by any program in P_n running for $t + x_n$ steps, then $s+1$ is not an output of any program in P_n. We show that $C(x_n) > n - c$. Note that such x_n exist.

Consider the program p : "dovetail the running of the programs in P_n till you find a newly outputted integer m such that, even after running all the other programs in P_n for this running time plus x_n, you do not get as output $m+1$. Now *output* $m+1$".

Note that there is such a p with $|p| < |x_n| + c$.

It follows from the definition of x_n that p is not in P_n, which means that p must have length at least $n+1$. Hence $n < |x_n| + c$ or $|x_n| > n - c$.

Clearly m will have Kolmogorov complexity around n. ■

One may well suspect that the shortest program for $m+1$ is really, "that integer one larger than the one outputted before gap x_n"!

4.7 Comparing the methods

The condition for Rice's theorem is that the classes into which we divide the programs should be complete index sets. This means that *all* programs computing a function in the given class, must be in the class. An example of a class which is *not* index complete is "those programs defined on their own index": Let p_j be a member of this class, hence $p_j(j)$ is defined. It is clearly not enough for a different program p_k to be defined on j, to be in the class. In other words, the class may contain an index of a program computing a function f, but not all programs computing f. This requirement is often not crucial for an imbedding into the halting problem (the way the standard proof of Rice's theorem goes) but it does make the imbedding easier. An example may be helpful:

Let us denote by A the class of programs which compute functions which are not nowhere-defined. We will use A as our basic undecidable class, as it is easy to show that A is unsolvable using our method, and using A avoids the self-referential details of the usual basic unsolvable class of programs (those programs defined on their own index).

Consider the process of showing that the class B of programs computing functions which are defined on 1, to be unsolvable. The standard method of imbedding B into the halting problem (but using our basic unsolvable class A) would associate to each program p, a program p' such that p is in A if and only if p' is in B. Clearly then, if we could decide B we could also decide A. We would generally construct p' to be a program which halts on input 1 if and only if p halts on any input. To do this, we could write p' to be some program which on input 1 simulates a dovetailing of p on all possible inputs and halts if and when p halts on some input. Clearly, p' will halt on input 1 if and only if p halts on any input. The index completeness of B assures us that

$$\text{however we construct } p', \text{ as long as } p' \text{ halts on input 1, it is in } B, \qquad (\mathbf{A})$$

so B has permissive entry criteria.

We will shortly give an example of a set of programs which we have explicitly designed to thwart this freedom but which is easily shown unsolvable via program length. Note first that the following two sets of programs are not index complete but are easily shown to be unsolvable using program length:

Example 49 *The set of programs defined on their own index (Note that this is not a complete index set): As in the proofs above, consider all those programs in P_n which are defined on their own index. Now run all of these programs till they halt on their index and then write a diagonal program that runs longer than any of these but halts for all inputs (since we want to be sure it halts on its own index).*

Many classes involving running time (the number of steps that a program takes for certain tasks) are especially easy to deal with.

Example 50 *The set of programs which are defined on at least two consecutive inputs and run at least twice as long on the second input than on the first: All we need do is find the maximum multiple of the first running time that is run on the second input and write a short diagonal program to exceed this.*

The following example is perhaps harder to imbed into the halting problem than the two above.

Example 51 *Let B be the set of indices i, where p_i is the first program (under some standard dovetailing) to halt with a new output: that is, for some $x_i \in \mathbf{N}$, p_i is the first program to halt and output x_i.*

Note that the example above allows us very little freedom to simulate the halting/non-halting question for an arbitrary program. In particular, we encounter problems with **A** above: If we attempt to imbed the halting problem into the above example, we cannot easily associate to any program p, a program p' such that p' is in B if and only if p is in A. This is because B has stringent entry criteria which depend on the *running time* of p' and not only on its output, and a long simulation of a program p by p' would already disqualify p' as an element of B.

We have made some attempts to set up problems which are even more difficult to reduce to the halting problem:

Example 52 *Let B be the (c.e.) set of all pairs (n, i) where i is an output string of some program of length at most n and we further have that all (n, j) for $j < i$ have been enumerated.*

Direct attempts at imbedding this problem in the halting problem are thwarted for similar reasons as in the example above.

Remark 53 *It is awkward to find the exact conditions for which the method will work. On the one hand we can say that it suffices, given a short program which gives an exhaustive list of programs in P_n with property* $\mathbf{P}$, *to be able to construct (or weaker, show there exists)a program or programs different to all these, with property* $\mathbf{P}$, *and of length $< n$. On the other, although our sets need not be index complete, i.e. exhaustive with regard to* the functions they compute, *they must at least be exhaustive* amongst a certain class of programs, *since we must be sure that the diagonal program is forced to be one of them. In the next section we will make some remarks that seem to imply that this method will only work for Turing complete sets of programs. This is in contrast to the examples above, where it seemed that the method may be able to show classes which are not Turing complete (see [Odifreddi],[Soare]), to be unsolvable.*

4.8 Turing completeness

Can we diagonalize against c.e. sets which are *not* reducible to the halting problem? (Particularly, can we show c.e. sets of degree less than that of the halting problem to be unsolvable using program length?) The following remarks seem to indicate that any set for which our method works is reducible to the halting problem, albeit not necessarily in the form of a straightforward imbedding.

Consider in each of the above sets C, that program p in P_n which is enumerated *last* amongst all the programs in C_n, say after t_p steps. Clearly if p had complexity much lower than n, we *could* use p to create a diagonal program p' in C_n. Hence $C(p) > n - c$ for each n and universal constant c. Now since $C(p) > n-c$, and any $t \geq t_n$ would specify p uniquely, we must have that $C(t) > n - c$, for all $t > t_n$. But then t_n as a number of steps would suffice to solve the halting problem for all programs in P_{n-k} for k some universal constant, since no program in P_m can run for more steps than t_m if $C(t) > m - c$, for all $t > t_m$, by the definition of Kolmogorov complexity.

A similar argument shows that if a c.e. set is such that the sum of the halting elements up to length n has complexity greater than $n - c$ for each n, the set is complete. This is because such a sum cannot be reached in fewer than t_n steps where $C(t) > n - c$, for all $t > t_n$, hence the running time must be longer than any number outputted by a program shorter than $n - c$. But any *such* number solves the halting problem for programs of length at most $n - c - k$. Hence in this case **maximally complex implies complete**.

In [Li and Vitányi] the remark is made that almost every c.e. set which is complete under the usual reducibilities, has the property that $C(\chi_{1:n}) > \log n$ for all n. For Turing reducibilities we can therefore say a little more.

Theorem 54 *There is an algorithm ϕ which will transform the characteristic function χ of a c.e. set S into a sequence with Kolmogorov complexity $C(\chi_{1:n}) > \log n$ for all n if and only if S is Turing complete.*

Proof. (Only if) By hypothesis, we would have that $C(\chi_{1:2^n}) > n$ for all n and hence the time t_n it takes to compute $\chi_{1;2^n}$ must have Kolmogorov complexity $C(t_n) > n$. Hence we could solve the halting problem up to length n given any time $t > t_n$. Hence S is complete.
(If) If we have a Turing complete c.e. set S then, by definition, there exists an algorithm transforming its characteristic sequence into that of the halting sequence, which has characteristic sequence $C(\chi_{1:n}) > \log n$ for all n. ∎

It follows from this theorem that "modulo the computable functions" all complete c.e. sets must have a certain density of elements in that there must be an algorithm which maps the characteristic sequence to that of the halting sequence, and this set has a high density of elements.

4.9 An Ω-like real for each functional computably enumerable predicate

Recall Chaitin's halting probability Ω : That is, if $p_1, p_2, ...$ is the sequence of halting programs for a universal prefix machine U, then $\Omega_U = \Sigma_{n=1}^{\infty} 2^{-|p_n|}$. Now consider such a universal *prefix machine* U and any c.e. functional property $\mathbf{P}$ of programs and consider for (p_i), the sequence of programs enumerated with property $\mathbf{P}$, the sum $S = \sum_x 2^{-|p_i|}$. Clearly, we can write a program p, which on input the first n digits of S, creates a diagonal program p' of complexity larger than n, with property $\mathbf{P}$, but running longer than all of the p_i in P_n. Hence $K(p') \geq n$, so $K(S_{1:n}) + |p| \geq n$. This holds for each n so setting $c = |p|$ we get that $K(S_{1:n}) \geq n - c$ for all n. Hence S is random.

4.10 An application to independent statements and proof lengths

Short statements with no short proofs have been examined for various axiomatic systems (see for example [Pudlák]). In this section we will make some informal, elementary remarks on proof lengths using complexity.

Let x_n be the number of halting programs in P_n.

Proposition 55 *Since $C(x_n) > n - c$ for a universal $c \in \mathbf{N}$, each of the statements "The number of halting programs in P_n is x_n" will be independent of any of the usual axiom systems, for n large enough.*

If this were not the case, we could on input of a large n (length $\log n$) and a search program, systematically search through all proofs in the axiom system to find the first statement of this form and output (the maximally complex) x_n. Clearly this is closely related to the analogous result for statements of the form "x is maximally complex" (see for example [Li and Vitányi] for a discussion of the latter type of statement).

Note also that

Proposition 56 *Each of the statements "The number of halting programs in P_n is at least s_n" is provable for each $s_n \leq x_n$ for the usual axiom systems, but for s_n close to x_n the proofs of such statements must be incredibly long. Indeed, consider as an example $s_n = x_n$. Then we cannot find the associated statement as the last line in any proof of which the length is specifiable in fewer than n bits!*

This follows from the following description of the number of halting programs in P_n : Search, for a fixed n, through all proofs up to length m which have as last line a statement of the form "The number of halting programs in P_n is at least s_n", now output the largest s_n appearing in such a statement.

Clearly if m was large enough, s_n *would* equal x_n, hence the fact that x_n is a maximally complex number means that m cannot be specifiable by a program much shorter than n digits.

Hence the only proof of "The number of halting programs in P_n is at least x_n" is of enormous length. In fact, it is quite easy to see that the length of this proof is comparable to the running time of the longest running program in P_n!

Bibliography

[Bennet and Gardner] C. H. BENNET and M. GARDNER, 'The random number omega bids fair to hold the mysteries of the universe' *Scientific American* 241 (1979) 20-34C.

[Borel 1] É. BOREL, 'Les probabilités dénombrables et leurs applications arithmétiques' *Rend. Circ. Mat. Palermo* 27 (1909) 247-271

[Borel 2] É. BOREL, *Leçons sur la théorie des fonctions* (Gauthier-Villars, Paris, 2nd ed., 1914)

[Calude] CALUDE, *Information and Randomness* (Springer-Verlag, 1994)

[Chaitin 1] G. J. CHAITIN, 'On the length of programs for computing finite binary sequences' *J. Assoc. Comput. Mach.* 13 (1966) 547–569

[Chaitin 2] G. J. CHAITIN, 'On the length of programs for computing finite binary sequences: statistical considerations' *J. Assoc. Comput. Mach.* 16 (1969) 145–159

[Chaitin 3] G. J. CHAITIN, 'A theory of program size formally identical to information theory' *J. Assoc. Comput. Mach.* 22 (1975) 329-340

[Chaitin 4] G. J. CHAITIN, *Algorithmic Information Theory* (Cambridge University Press, 1987)

[Church] A. CHURCH, 'On the concept of a random sequence' *Bull. Amer. Math. Soc.* 46 (1940) 130-135

[Davie] G. DAVIE, 'Discrepancy of Complex Sequences via Compressibility' *JCMCC* to appear.

[Feller] W. FELLER, *An introduction to probability theory and its applications* (Wiley, 1968) 3rd ed.

[Fouché 1] W. L. FOUCHÉ, 'Descriptive Complexity and Reflective Properties of Combinatorial Configurations' *J. London Math. Soc.* 54 (1996) 199-208

[Fouché 2] W. L. FOUCHÉ, 'Discrepancies of Hypergraphs of high Kolmogorov complexity' *JCMCC* to appear.

[Fouché 3] W. L. FOUCHÉ, 'Identifying randomness given by high descriptive complexity' *Acta Appl. Math.* 34 (1994) 313-328

[Fouché and Potgieter] W. L. FOUCHÉ and P. H. POTGIETER, 'Kolmogorov Complexity and Symmetric Relational Structures' *J. Symbolic Logic* 63 (1998) 1083-1094

[Gács 1] P. GÁCS, 'On the symmetry of algorithmic information' *Soviet Math. Dokl.* 15 (1974) 1477-1480; Correction Ibid. 15 (1974) 1480

[Gács 2] P. GÁCS, 'Exact expressions for some randomness tests' *Zeitschrift für Math. Logic Grundlag. Math.* 26 (1980) 385-394

[Hinman] P. G. HINMAN, *Recursion-Theoretic Hierarchies* (Springer-Verlag, 1978)

[Kolmogorov 1] A. N. KOLMOGOROV, 'Three approaches to the quantitative definition of information' *Problems Inform. Transmission* 1 (1965) 1-7

[Kolmogorov 2] A. N. KOLMOGOROV, 'Logical basis for information theory and probability theory' *IEEE Trans. Inform. Theory* IT-14 (1968) 662-664

[Levin 1] L. A. LEVIN, 'On the notion of a random sequence' *Soviet Math. Dokl.* 14 (1973) 1413-1416

[Levin 2] L. A. LEVIN, 'Laws of information conservation (non-growth) and aspects of the foundation of probability theory' *Problems Inform. Transmission* 10 (1974) 206-210

[Li and Vitányi] M. LI and P. VITÁNYI, *An Introduction to Kolmogorov Complexity and Its Applications* (Springer-Verlag, 1993)

[Martin-Löf 1] P. MARTIN-LÖF, 'The definition of random sequences' *Inform. and Control* 9 (1966) 602-619

[Martin-Löf 2] P. MARTIN-LÖF, 'Complexity oscillations in infinite binary sequences' *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 19 (1971) 225-230

[Odifreddi] P. ODIFREDDI, *Classical Recursion Theory* (North-Holland, 1989)

[Pudlák] P. PUDLÁK, 'The lengths of proofs' *Handbook of proof theory* (North-Holland, 1998)

[Rado] R. RADO, 'Universal graphs and universal functions' *Acta Arith.* 9 (1964) 393-407

[Rosenberg and Salomaa] G. ROSENBERG and A. SALOMAA, *Cornerstones of Undecidability* (Prentice Hall, 1993)

[Schnorr] C. P. SCHNORR, 'Process complexity and effective random tests' *J. Comput. System Sci.* 7 (1973) 376-388

[Shen'] A. Kh. SHEN', 'Connections between different algorithmic definitions of randomness' *Soviet Math. Dokl.* 38(2) (1989) 316-319

[Soare] R. I. SOARE, *Recursively Enumerable Sets and Degrees* (Springer-Verlag, 1987)

[Solomonoff 1] R. J. SOLOMONOFF, 'A preliminary report on a general theory of inductive inference' *Technical report ZTB-138, Zator Company, Cambridge, Mass.* November 1960.

[Solomonoff 2]	R. J. SOLOMONOFF, 'A formal theory of inductive inference' part 1 and part 2 *Inform. Contr.* 7 (1964) 1-22 and 224-254
[Solovay]	R. SOLOVAY, *Lecture notes on algorithmic complexity* (UCLA 1975 Unpublished)
[Spencer]	J. SPENCER, 'Six standard deviations suffice' *Trans. Amer. Math. Soc.*289 (1985) 679-706
[van Lambalgen 1]	M. VAN LAMBALGEN, 'Von Mises' definition of random sequences reconsidered' *The Journal of Symbolic Logic* 52 (1987) 725-755
[van Lambalgen 2]	M. VAN LAMBALGEN, 'The axiomatization of randomness' *The Journal of Symbolic Logic* 55 (1990) 1143-1167
[van Lambalgen 3]	M. VAN LAMBALGEN, 'Independence, randomness and the axiom of choice' *The Journal of Symbolic Logic* 57 (1992) 1274-1304
[von Mises]	R. VON MISES, 'Grundlagen der Wahrscheinlichkeitsrechnung' *Mathematische Zeitschrift* 5 (1919) 52-99
[Wald]	A. WALD, 'Die Wiederspruchsfreiheit des Kollektivbegriffes der Wahrscheinlichkeitsrechnung' *Ergebnisse eines mathematischen Kolloquiums* 8 (1937) 38-72