INAUGURALDISSERTATION

zur

Erlangung der Doktorwürde

der

Naturwissenschaftlich-Mathematischen
Gesamtfakultät

der

Ruprecht-Karls-Universität Heidelberg

Vorgelegt von

Diplom-Mathematiker Rupert Hölzl

aus

München.

Tag der mündlichen Prüfung: 16. Dezember 2010

Thema

Kolmogorovkomplexität

Gutachter: Priv.-Doz. Dr. Wolfgang Merkle
Prof. Dr. Frank Stephan

Kolmogorov complexity

by Rupert Hölzl

English abstract: This dissertation discusses new results on Kolmogorov complexity. Its first part focuses on the study of Kolmogorov complexity without time bounds. Here we deal with the concept of non-monotonic randomness, that is randomness characterized by martingales that bet non-monotonically. We will state the definitions of several different randomness classes and then separate them from each other. We also present a a systematic survey of a wide array of traceability notions and characterize them through (auto)complexity notions. Traceabilities are a group of notions that express that a set is not far away from being computable.

The second part of the document deals with the topic of time bounded Kolmogorov complexity. First we investigate the difference between two ways of describing a word: the complexity of describing it well enough so that it can be distinguished from other words; and the complexity of describing it well enough so that the word can actually be produced from the description. While this difference is unimportant in the case of Kolmogorov complexity without time bounds it plays an essential role when time bounds are present. Next, we introduce the notion of computational depth and prove a dichotomy result about it that is reminiscent of Kummer's well-known gap theorem. Lastly, we look at the important notion of Solovay functions. Solovay functions are computable upper bounds of Kolmogorov complexity that are actually sharp infinitely often. We will use them, first, to characterize Martin-Löf randomness in a certain way and, second, to give a characterization of being jump-traceable.

Deutsche Zusammenfassung: In dieser Dissertation werden neue Ergebnisse über Kolmogorovkomplexität diskutiert. Ihr erster Teil konzentriert sich auf das Studium von Kolmogorovkomplexität ohne Zeitschranken. Hier beschäftigen wir uns mit dem Konzept nicht-monotoner Zufälligkeit, d.h. Zufälligkeit, die von Martingalen charakterisiert wird, die in nicht-monotoner Reihenfolge wetten dürfen. Wir werden in diesem Zusammenhang eine Reihe von Zufälligkeitsklassen einführen, und diese dann von einander separieren. Wir präsentieren außerdem einen systematischen Überblick über verschiedene Traceability-Begriffe und charakterisieren diese durch (Auto-)Komplexitätsbegriffe. Traceabilities sind eine Gruppe von Begriffen, die ausdrücken, dass eine Menge beinahe berechenbar ist.

Der zweite Teil dieses Dokuments beschäftigt sich mit dem Thema zeitbeschränkter Kolmogorovkomplexität. Zunächst untersuchen wir den Unterschied zwischen zwei Arten, ein Wort zu beschreiben: Die Komplexität, es genau genug zu beschreiben, damit es von anderen Wörter unterschieden werden kann; sowie die Komplexität, es genau genug zu beschreiben, damit das Wort aus der Beschreibung tatsächlich generiert werden kann. Diese Unterscheidung ist im Falle zeitunbeschränkter Kolmogorovkomplexität nicht von Bedeutung; sobald wir jedoch Zeitschranken einführen, wird sie essentiell. Als nächstes führen wir den Begriff der Tiefe ein und beweisen ein ihn betreffendes Dichotomieresultat, das in seiner Struktur an Kummers bekanntes Gap-Theorem erinnert. Zu guter Letzt betrachten wir den wichtigen Begriff der Solovayfunktionen. Hierbei handelt es sich um berechenbare obere Schranken der Kolmogorovkomplexität, die unendlich oft scharf sind. Wir benutzen sie, um in einem gewissen Zusammenhang Martin-Löf-Zufälligkeit zu charakterisieren, und um eine Charakterisierung von Jump-Traceability anzugeben.

Contents

CHAPTER 1

Introduction

Since the 1930s, mathematicians such as Gödel, Church and Turing have considered the notion of computable (or decidable) sets. That is, subsets of the natural numbers $\mathbb{N}$ that can be described in an effective way using only a finite amount of information. Trivially, every finite set is computable; but there are also many infinite sets that can be described in such a way. Being computable then means that the set somehow exhibits enough internal structure and regularity, that despite its infinite cardinality a finite amount of information suffices to describe it.

Of course, the set of subsets of $\mathbb{N}$ in uncountable whereas a countable list of all finite descriptions can be given. So it is obvious that all but countably many subsets of $\mathbb{N}$ cannot be computable.

So how difficult *is* it to describe more sets? To investigate this the notion of Kolmogorov complexity was introduced by R.J. Solomonoff, A.N. Kolmogorov and G.J. Chaitin.[1] Assume we want to describe some non-computable set A. If we look at initial segments $A \upharpoonright i$, that is, the sets $A \cap \{0, \dots, i\}$ for increasing i, how much information do we need to describe those?

Of course, we can always describe such an initial segment of length i by giving a sequence of i values in $\{0, 1\}$. But for some sets A we can actually use fewer bits than this trivial bound, namely for those sets that, while not exhibiting enough regularity to be computable, still exhibit enough structure so that we can economise.

As it turned out, there are many non-computable sets exhibiting such regularity and Kolmogorov complexity is a useful tool to investigate and describe them.

Sequences that do *not* exhibit such regularities are called random and have been the central object of study in algorithmic randomness. Many interesting insights in this area have resulted from trying to relate various notions of randomness with various notions of computational power [DH10, LV08, Nie09].

In the last decades, a variant of Kolmogorov complexity came into focus: In this variant, not all space-saving descriptions of sets A are eligible, but only those

[1] See [LV08] for a more detailed account of the history of the notion.

that can be (in some sense) quickly executed to output A. This is know as time-bounded Kolmogorov complexity. This notion can act as a liaison between the investigation of Kolmogorov complexity and that of classical structural complexity theory. The maybe most important difference between Kolmogorov complexity with time bound and that without is that Kolmogorov complexity with time bound is itself a computable function.

1.1 Summary

The purpose of this dissertation is to give some new results on Kolmogorov complexity. It essentially consists of two parts.

Part I, with the exception of a short digression in chapter 4, focuses on the study of Kolmogorov complexity without time bounds. The first chapter in this part is chapter 3, which deals with the concept of non-monotonic randomness, that is randomness characterized by martingales that bet non-monotonically. We will state the definitions of several different randomness classes and then separate them from each other.

In chapter 4 we present a systematic survey of a wide array of traceability notions and characterize them through (auto)complexity notions. Traceabilities are a group of notions that express that a set is not far away from being computable.

Part II deals with the topic of time bounded Kolmogorov complexity. Chapter 5 is concerned with the difference between two ways of describing a word: the complexity of describing it well enough so that it can be distinguished from other words; and the complexity of describing it well enough so that the word can actually be produced from the description. While this difference is unimportant in the case of Kolmogorov complexity without time bounds it plays an essential role when time bounds are present.

The next chapter, chapter 6, introduces the notion of computational depth and proves a dichotomy result about it that is reminiscent of Kummer's well-known gap theorem [DH10, Kum96].

The last chapter 7 deals with the important notion of Solovay functions. Solovay functions are computable upper bounds of Kolmogorov complexity that are actually sharp infinitely often (up to an additive constant). We will use them, first, to characterize Martin-Löf randomness in a certain way and, second, to give a characterization of being jump-traceable.

1.2 Publications

The work presented in chapter 3 has been published in the proceedings of the 6th International Conference on Computability and Complexity in Analysis in Ljubljana in 2009 [BHKM09] and will soon appear in the Journal of Logic and Computation [BHKM]. The largest part of chapter 4 has been published in the proceedings of the IFIP Conference on Theoretical Computer Science in Brisbane

in 2010 [HM10]. The work contained in chapter 5 has been published in the proceedings of the 5th International Conference on the Theory and Applications of Models of Computation in Xi'an in 2008 [HM08]. The work presented in chapter 7 and parts of chapter 6 have been published in the proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science in Nový Smokovec in 2009 [HKM09].

1.3 Thanks

This doctoral thesis would not have been possible without a whole list of people. My special thanks go to my supervisor and co-author, Priv.-Doz. Dr. Wolfgang Merkle, with whom most of the work in this document has been done and who has supported me through the whole dissertation process. Another big thank you goes to my other co-authors together with whom a significant part of the results in this document were achieved; they are Dr. Laurent Bienvenu and Thorsten Kräling. I am also grateful to Prof. Dr. Frank Stephan for being the second reviewer of this document.

Furthermore, I want to thank Prof. Klaus Ambos-Spies, the head of the Heidelberg Logic Group. Finally I want to thank Felicitas Hirsch, who made our lives at Heidelberg significantly easier, and my office mate Timur Bakibayev with whom I had many interesting mathematical discussions and who helped me with many things, not the least of which was with fixing my car.

I am very grateful for the funding for my dissertation which was provided by the Deutsche Forschungsgemeinschaft grant ME 1806/3-1.

CHAPTER 2

Preliminaries

This chapter will provide some general background, essential definitions and notations. Readers well-acquainted with the definitions and conventions used in the field of algorithmic randomness can skip this part.

For $i \in \{0,1\}$, define $\bar{i} := 1 - i$.

We look at finite strings and infinite sequences over the alphabet $\{0,1\}$, that is, at elements of the sets $\{0,1\}^{<\infty}$ and $\{0,1\}^{\infty}$, respectively. For a string x let $|x|$ denote the length of x, that is, the number l such that $x \in \{0,1\}^l$. Let ϵ denote the string of length 0.

Depending on the situation it can simplify notations if we identify the finite strings $\{0,1\}^{<\infty}$ with the natural numbers $\mathbb{N}$. To achieve this, we order the finite strings length-lexicographically, that is, we order them using the length of the string as the primary and the lexicographical ordering as the secondary criterion. Thus we arrive at the order $\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots$

Given $v \in \{0,1\}^{<\infty}$ and $w \in \{0,1\}^{<\infty} \cup \{0,1\}^{\infty}$, we write $v \sqsubseteq w$ if v is a prefix of w. Let $w(i)$ denote the i-th bit of w where by convention there is a 0-th bit and $w(i)$ is undefined if w is a word of length less than $i+1$. For $i < j$ we also write $w(i \ldots j)$ for $w(i) \ldots w(j)$.

We will often identify a set $A \in \mathbb{N}$ with a sequence $\alpha \in \{0,1\}^{\infty}$, where $\alpha(i) = 1$ if and only if $i \in A$. If, for the purpose of the exposition, we want to insist more on the set perspective we will prefer to denote these sets by upper case latin letters $A, B, \ldots$; if we want to insist more on the sequence perspective we may also use greek letters $\alpha, \beta, \ldots$

If $A \in \{0,1\}^{\infty}$ and $X = \{x_0 < x_1 < x_2 < \ldots\}$ is a subset of $\mathbb{N}$ then $A \upharpoonright X$ is the finite or infinite binary sequence $A(x_0)A(x_1)\ldots$. We abbreviate $A \upharpoonright \{0, \ldots, n-1\}$ by $A \upharpoonright n$ (i.e., the prefix of A of length n).

Logarithms to base 2 are denoted by log, and often a term of the form $\log t$ will indeed denote the least natural number s such that $t \leq 2^s$.

An *order* is a function $h : \mathbb{N} \to \mathbb{N}$ that is non-decreasing and unbounded.

For the definition of a Turing machine, an oracle Turing machine, a universal Turing machine and an additively optimal Turing machine, as well as for existence proofs for Turing machines conforming to the last two definitions, we refer the reader to Li and Vitányi [LV08].

The e-th partially computable function according to some standard acceptable numbering will be called φ_e. Partial functions map natural numbers to natural numbers, unless explicitly specified differently. We let $\mathrm{W}_0, \mathrm{W}_1, \ldots$ be the numbering of all computably enumerable (c.e.) sets, i.e., W_e is the domain of the e-th partial computable function φ_e.

The computation of a machine M on input x does not necessarily terminate. To express that it indeed does, we write $M(x)\downarrow$. To express that the computation terminates and outputs y we write $M(x)\downarrow = y$. For a set A, the jump A' is defined as $\{e \mid \varphi_e^A(e)\downarrow\}$, the halting problem with oracle access to A. We write A'' for $(A')'$ etc. Trivially, $\emptyset' = H$, where H denotes the halting problem.

A set D of strings is called prefix-free, if for any two strings $x, y \in D$, the assumption $x \sqsubseteq y$ implies $x = y$. In order to define plain and prefix-free Kolmogorov complexity, we fix additively optimal oracle Turing machines $\mathbb{V}$ and $\mathbb{U}$, where $\mathbb{U}$ has prefix-free domain. We let $\mathrm{C}_M^A(x)$ denote the Kolmogorov complexity of x with respect to a Turing machine M relative to oracle A, that is

$$\mathrm{C}_M^A(x) := \min\{|\sigma| \colon M^A(\sigma)\downarrow = x\}.$$

We let $\mathrm{C}_M(x) = \mathrm{C}_M^{\emptyset}(x)$, $\mathrm{C}^A(x) = \mathrm{C}_{\mathbb{V}}^A(x)$, and $\mathrm{C}(x) = \mathrm{C}_{\mathbb{V}}^{\emptyset}(x)$. The prefix-free Kolmogorov complexities K_N^A, K_N, K^A and K are defined likewise through a prefix-free machine N or the universal prefix-free machine $\mathbb{U}$, respectively.

Let Ω denote the probability that a random program halts when executed on $\mathbb{U}$, that is $\Omega := \sum_{\mathbb{U}(x)\downarrow} 2^{-|x|}$.

In connection with the definition of time-bounded Kolmogorov complexity, we assume that $\mathbb{V}$ and $\mathbb{U}$ both are able to simulate any other Turing machine M running for t steps in $\mathrm{O}(t \cdot \log t)$ steps for an arbitrary machine M and in $\mathrm{O}(t(n))$ steps in case M has only two work tapes. Again, for an existence proof, see the monograph of Li and Vitányi [LV08].

For a computable function $t \colon \mathbb{N} \to \mathbb{N}$ and a machine M, the Kolmogorov complexity relative to M with time bound t is

$$\mathrm{C}_M^t(x) := \min\{|\sigma| \colon M(\sigma)\downarrow = x \text{ in at most } t(|x|) \text{ steps}\},$$

and we write C^t for $\mathrm{C}_{\mathbb{V}}^t$. The prefix-free Kolmogorov complexity with time bound t denoted by $\mathrm{K}_M^t(n)$ and $\mathrm{K}^t(n) = \mathrm{K}_{\mathbb{U}}^t$ is defined likewise by considering only prefix-free machines and the corresponding universal machine $\mathbb{U}$ in place of $\mathbb{V}$.

Let $\leq^+$ denote the relation less than or equal up to an additive constant. The relations $\geq^+$ and $=^+$ are defined likewise. As usual, $\mathrm{O}(f)$ denotes a function that grows at most as fast as f, up to a multiplicative constant, and we write $\Theta(f)$ for a function g that grows equally fast as f, up to a multiplicative constant. That is,

$g = \Theta(f)$ is equivalent to $g = \mathrm{O}(f) \wedge f = \mathrm{O}(g)$. Depending on the context, we sometimes also write $\mathrm{O}(f)$ for the *set* of all functions g such that $g = \mathrm{O}(f)$, and analogously for $\Theta(f)$.

We say that a function g dominates another function f iff for almost all n we have $f(n) \leq g(n)$.

For a set of finite sequences $W \subseteq \{0,1\}^{<\infty}$, let the cylinder of W, denoted by $[W]$, be the set $\{\alpha \in \{0,1\}^{\infty} \mid \exists i : \alpha \upharpoonright i \in W\}$. For $w \in \{0,1\}^{<\infty}$ we write $[w]$ instead of $[\{w\}]$.

A Martin-Löf test (or, for short, ML-test) is a sequence of uniformly c.e. sets $U_0, U_1, U_2, \ldots$ such that for all i, $U_i \subseteq \{0,1\}^{<\infty}$ and $\mu([U_i]) \leq 2^{-i}$, where μ denotes Lebesgue measure.

A sequence $\alpha \in \{0,1\}^{\infty}$ is covered by an ML-test $(U_i)_{i\in\mathbb{N}}$ iff $\alpha \in \bigcap_{i\in\mathbb{N}} [U_i]$.

A sequence α (and, by identifying it with a sequence, a set) is called ML-random if it is not covered by any ML-test. The set of all ML-random sequences is denoted by MLR.

Theorem 2.1 (Schnorr [Sch73])**.** *The following statements are equivalent for any sequence $\alpha \in \{0,1\}^{\infty}$.*

1. *α is ML-random.*
2. *There is a constant c such that for all n, $\mathrm{K}(\alpha \upharpoonright n) \geq n - c$.*

Part I

Kolmogorov complexity without time bounds

CHAPTER 3

Non-monotonic Randomness

Intuitively speaking, a binary sequence is random if the bits of the sequence do not have effectively detectable regularities. This idea can be formalized in terms of betting strategies, that is, a sequence will be called random in case the capital gained by successive bets on the bits of the sequence according to a fixed betting strategy must remain bounded, where we assume that the game is fair and a fixed set of admissible betting strategies is understood.

The notions of random sequences that have received most attention are Martin-Löf randomness and computable randomness. Here a sequence is called computably random if no total computable betting strategy can achieve unbounded capital by betting on the bits of the sequence in the natural order, a definition that indeed is natural and suggests itself. However, computably random sequences may lack certain properties associated with the intuitive understanding of randomness, for example there are such sequences that are highly compressible, i.e., show a large amount of redundancy, see Theorem 3.4 below. Martin-Löf randomness behaves much better in this and other respects. Indeed, the Martin-Löf random sequences can be characterized as the sequences that are incompressible in the sense that all their initial segments have essentially maximal Kolmogorov complexity, and in fact this holds for several versions of Kolmogorov complexity according to celebrated results by Schnorr, by Levin and, recently, by Miller and Yu [DH10]. On the other hand, it has been held against the concept of Martin-Löf randomness that its definition involves effective approximations, i.e., a very powerful, hence rather unnatural model of computation, and indeed the usual definition of Martin-Löf randomness in terms of left-computable martingales, that is, in terms of betting strategies where the gained capital can not be computed but only effectively approximated from below, is not very intuitive.

It can be shown that Martin-Löf randomness strictly implies computable randomness (see Schnorr [Sch71]). According to the preceding discussion the latter notion is too inclusive while the former may be considered unnatural. Ideally, we

would therefore like to find a more natural characterization of ML-randomness; or, if that is impossible, we are alternatively interested in a notion that is close in strength to ML-randomness, but has a more natural definition. One promising way of achieving such a more natural characterization or definition could be to use computable betting strategies that are more powerful than those used to define computable randomness.

Muchnik [MSU98] proposed to consider computable betting strategies that are non-monotonic in the sense that the bets on the bits need not be done in the natural order, but such that the position of the bit to bet on next can be computed from the already scanned bits. The corresponding notion of randomness is called Kolmogorov-Loveland randomness because Kolmogorov and Loveland independently had proposed concepts of randomness defined via non-monotonic selection of bits.

Kolmogorov-Loveland randomness is implied by [Nie09, Proposition 7.6.20] and in fact is quite close to Martin-Löf randomness, as we will see in connection with Theorem 3.16, but whether the two notions are distinct is one of the major open problems of algorithmic randomness. In order to get a better understanding of this open problem and of non-monotonic randomness in general, Miller and Nies [MN06] introduced restricted variants of Kolmogorov-Loveland randomness, where the sequence of betting positions must be non-adaptive, i.e., can be computed in advance without accessing the sequence on which one bets.

The randomness notions mentioned so far are determined by two parameters that correspond to the columns and rows, respectively, of the table in Figure 3.1. First, the sequence of places that are scanned and on which bets may be placed, while always being given effectively, can just be monotonic, can be equal to $\pi(0), \pi(1), \ldots$ for a permutation or an injection π from $\mathbb{N}$ to $\mathbb{N}$, or can be adaptive, i.e., the next bit depends on the bits already scanned. Second, once the sequence of scanned bits is determined, betting on these bits can be done according to a betting strategy where the corresponding martingale is total or partial computable, or is left-computable. The inclusions known from existing literature between the corresponding classes of random sequences are shown in Figure 3.1; see Section 3.1 for technical details and for the definitions of the class acronyms that occur in the figure.

The classes in the last row of the table in Figure 3.1 all coincide with the class of Martin-Löf random sequences by the folklore result that left-computable martingales always yield the concept of Martin-Löf randomness, no matter whether the sequence of bits to bet on is monotonic or is determined adaptively, because even in the latter, less restrictive model one can uniformly in k enumerate an open cover of measure at most $1/k$ that covers all the sequences on which some universal martingale exceeds k — which easily yields an ML-test. Furthermore, the classes in the first and second row of the last column both yield the class of Kolmogorov-Loveland random sequences, because it can be shown that total and partial adaptive betting strategies yield the same concept of random sequence [Mer03]. Finally, it follows easily from results of Buhrman et al. [BvMR$^+$00] that the class TMR of computably random sequences coincides with the class TPR of sequences that are random with respect to total

	monotonic		permutation		injection		adaptive
total	TMR	$=$	TPR	$\supseteq$	TIR	$\supseteq$	KLR
	$\cup\!\mid$		$\cup\!\mid$		$\cup\!\mid$		$\mid\!\mid$
partial	PMR	$\supseteq$	PPR	$\supseteq$	PIR	$\supseteq$	KLR
	$\cup\!\mid$		$\cup\!\mid$		$\cup\!\mid$		$\cup\!\mid$
left-computable	MLR	$=$	MLR	$=$	MLR	$=$	MLR

Figure 3.1: Known class inclusions

permutation martingales, i.e., the ability to scan the bits of a sequence according to a computable permutation does not increase the power of total martingales.

Concerning non-inclusions, it is well-known [MSU98, AS98] that it holds that

$$\mathsf{KLR} \subsetneq \mathsf{PMR} \subsetneq \mathsf{TMR}.$$

Furthermore, Kastermans and Lempp [KL10] have recently shown that the Martin-Löf random sequences form a proper subclass of the class PIR of partial injective random sequences, i.e., $\mathsf{MLR} \subsetneq \mathsf{PIR}$.

In what follows, we investigate the six randomness notions that are shown in Figure 3.1 in the range between PIR and TMR, i.e., between partial injective randomness as introduced below and computable randomness. We obtain a complete picture of the inclusion structure of these notions, more precisely we show that the notions are mutually distinct and indeed are mutually incomparable with respect to set theoretical inclusion, except for the inclusion relations that follow trivially by definition and by the known relation $\mathsf{TMR} \subseteq \mathsf{TPR}$, see Figure 3.3 at the end of this paper. Interestingly these separation results are obtained by investigating the possible values of the Kolmogorov complexity of initial segments of random sequences for the different strategy types, and for some randomness notions we obtain essentially sharp bounds on how low these complexities can be.

3.1 Permutation and injection randomness

We now review the concept of martingale and betting strategy that are central for the unpredictability approach to define notions of an infinite random sequence.

Definition 3.1. *A* martingale *is a non-negative, possibly partial, function d from $\{0,1\}^{<\infty}$ to $\mathbb{Q}$ such that for all $w \in \{0,1\}^{<\infty}$, $d(w0)$ is defined if and only if $d(w1)$ is, and if these are defined, then so is $d(w)$, and the relation $2d(w) = d(w0) + d(w1)$ holds. A martingale* succeeds *on a sequence $A \in \{0,1\}^{\infty}$ if $d(A \upharpoonright n)$ is defined for all n, and* $\limsup d(A \upharpoonright n) = +\infty$. *We denote by* $\mathrm{Succ}(d)$ *the* success set *of d, i.e., the set of sequences on which d succeeds.*

Intuitively, a martingale represents the capital of a player who bets on the bits of a sequence $A \in \{0,1\}^\infty$ in order, where at every round he bets some amount of money on the value of the next bit of A. If his guess is correct, he doubles his stake. If not, he loses his stake. The quantity $d(w)$, with w a string of length n, represents the capital of the player before the n-th round of the game (by convention there is a 0-th round) in case the first n bits revealed so far are those of w.

We say that a sequence A is computably random if no total computable martingale succeeds on it. One can extend this in a natural way to partial computable martingales: a sequence A is partial computably random if no partial martingale succeeds on it. No matter whether we consider partial or total computable martingales, this game model can be seen as too restrictive as described at the beginning of the chapter. Instead, one could allow the player to bet on bits in any order he likes (as long as he can visit each bit at most once). This leads us to extend the notion of martingale as follows.

Definition 3.2. *A* betting strategy *is a pair $b = (d, \sigma)$ where d is a martingale and σ is a function from $\{0,1\}^{<\infty}$ to $\mathbb{N}$.*

For a strategy $b = (d, \sigma)$, the term σ is called the *scan rule*. For a string w, $\sigma(w)$ represents the position of the next bit to be visited if the player has read the sequence of bits w during the previous moves. And as before, d specifies how much money is bet at each move. Formally, given a sequence $A \in \{0,1\}^\infty$, we define by induction a sequence of positions $n_0, n_1, \ldots$ by

$$\begin{aligned} n_0 &= \sigma(\epsilon), \\ n_{k+1} &= \sigma\left(A(n_0)A(n_1)\ldots A(n_k)\right) \text{ for all } k \geq 0 \end{aligned}$$

and we say that $b = (d, \sigma)$ succeeds on A if the n_i are all defined and pairwise distinct (i.e., no bit is visited twice) and

$$\limsup_{k \to +\infty} d\left(A(n_0)\ldots A(n_k)\right) = +\infty$$

Here again, a betting strategy $b = (d, \sigma)$ can be total or partial. In fact, its partiality can be due either to the partiality of d or to the partiality of σ. We say that a sequence is Kolmogorov-Loveland random if no total computable betting strategy succeeds on it. As noted by Merkle [Mer03], the concept of Kolmogorov-Loveland randomness remains the same if one replaces "total computable" by "partial computable" in the definition.

Kolmogorov-Loveland randomness is implied by Martin-Löf randomness and whether the two notions can be separated is one of the most important open problems in algorithmic randomness. As we discussed above, Miller and Nies [MN06] proposed to look at intermediate notions of randomness, where the power of non-monotonic betting strategies is limited. In the definition of a betting strategy, the scan rule is adaptive, i.e., the position of the next visited bit depends on the bits previously seen. It is interesting to look at non-adaptive games.

Definition 3.3. *In the above definition of a strategy, when $\sigma(w)$ only depends on the length of w for all w (i.e., the decision of which bit should be chosen at each move is independent of the values of the bits seen in previous moves), we identify σ with the (injective) function $\pi : \mathbb{N} \to \mathbb{N}$, where, for all n, $\pi(n)$ is the value of σ on all words of length n ($\pi(n)$ indicates the position of the bit visited during the n-th move), and we say that $b = (d, \pi)$ is an* injection strategy. *If moreover π is bijective, we say that b is a* permutation strategy. *If π is the identity, the strategy $b = (d, \pi)$ is said to be* monotonic, *and can clearly be identified with the martingale d.*

The preceeding discussion naturally leads to a number of randomness notions with non-adaptive scan rules: one can consider either monotonic, permutation, or injection strategies, and either total computable or partial computable ones. This gives a total of six randomness classes, which we denote by

$$\mathsf{TMR},\ \mathsf{TPR},\ \mathsf{TIR},\ \mathsf{PMR},\ \mathsf{PPR},\ \text{and}\ \mathsf{PIR}, \tag{3.1}$$

where the first letter indicates whether we consider total (T) or partial (P) strategies, and the second indicates whether we look at monotonic (M), permutation (P) or injection (I) strategies. For example, the class TMR is the class of computably random sequences, while the class PIR is the class of sequences A such that no partial injection strategy succeeds on A. Recall in this connection that the previously known inclusions between the six classes in (3.1) and the classes KLR and MLR of Kolmogorov-Loveland random and Martin-Löf random sequences have been stated in Figure 3.1 above.

3.2 Randomness notions based on total computable strategies

We begin our study with the randomness notions arising from the game model where strategies are total computable. As we will see in this section, in this model, it is possible to construct sequences that are random and yet have very low Kolmogorov complexity (i.e. all their initial segments are of low Kolmogorov complexity). We will see in section 3.3 that this is no longer the case when we allow partial computable strategies in the model.

Building a sequence in TMR of low complexity

The following theorem is a first illustration of the phenomenon we just described.

Theorem 3.4 (Lathrop and Lutz [LL99], Muchnik [MSU98])**.** *For every computable order h, there is a sequence $A \in \mathsf{TMR}$ and a $c \in \mathbb{N}$ such that for all $n \in \mathbb{N}$,*

$$\mathrm{C}(A \upharpoonright n | n) \leq h(n) + c.$$

Definition 3.5. *For a function f from $\mathbb{N}$ to $\mathbb{N}$ that is unbounded and non-decreasing let the* discrete inverse f^{-1} *be the function that maps k to the greatest n such that $f(n) \leq k$.*

Proof idea. Defeating one total computable martingale is easy and can be done computably, i.e., for every total computable martingale d there exists a sequence A, uniformly computable in d, such that $A \notin \mathrm{Succ}(d)$. Indeed, fix a martingale d. For any given w, one has either $d(w0) \leq d(w)$ or $d(w1) \leq d(w)$. Thus, one can easily construct a computable sequence A by setting $A \upharpoonright 0 = \epsilon$ and by induction, having defined $A \upharpoonright n$, we choose $A \upharpoonright n+1 = (A \upharpoonright n)i$ where $i \in \{0,1\}$ is such that $d((A \upharpoonright n)i) \leq d(A \upharpoonright n)$. This can of course be done computably since d is total computable, and by construction of A, the values $d(A \upharpoonright n)$ form a non-increasing sequence, meaning in particular that d does not succeed against A.

Defeating a finite number of total computable martingales is equally easy. Indeed, given a finite number $d_1, \ldots, d_k$ of such martingales, their sum $D = d_1 + \ldots + d_k$ is itself a total computable martingale (this follows directly from the definition). Thus, we can construct as above a computable sequence A that defeats D. And since $D \geq d_i$ for all $1 \leq i \leq k$, this implies that A defeats all the d_i. Note that this argument would work just as well if we had taken D to be any weighted sum $\alpha_1 d_1 + \ldots + \alpha_k d_k$, with positive rational constants α_i.

We now need to deal with the general case where we have to defeat *all* total computable martingales simultaneously. We will again proceed using a diagonalization technique. Of course, this diagonalization cannot be carried out effectively, since there are infinitely many such martingales and since we do not even know whether any one given partial computable martingale is total. The first problem can easily be overcome by introducing the martingales to diagonalize against one by one instead of all at the beginning. So at first, for a number of stages we will only take into account the first computable martingale d_1. Then (maybe after a long time) we may introduce the second martingale d_2, with a small coefficient α_2 and then consider the martingale $d_1 + \alpha_2 d_2$ (α_2 is chosen to be sufficiently small to ensure that the difference in capital between d_1 and $d_1 + \alpha_2 d_2$ is small at the time when d_2 is added). Much later we can introduce the third martingale d_3 with an even smaller coefficient α_3, and diagonalize against $d_1 + \alpha_2 d_2 + \alpha_3 d_3$, and so on. So in each step of the construction we have to consider just a finite number of martingales.

The non-effectivity of the construction arises from the second problem, deciding which of our partial computable martingales are total. However, once we are supplied with this additional information, we *can* effectively carry out the construction of A. And since for each step we need to consider only finitely many potentially

total martingales, the information we need to construct the first n bits of A for some fixed n is finite, too. Say, for example, that for the first n stages of the construction — i.e., to define $A \upharpoonright n$ — we decided on only considering k martingales $d_0, \ldots, d_k$. Then we need no more than k bits, carrying the information which martingales among $d_0, \ldots, d_k$ are total, to describe $A \upharpoonright n$. That way, we get $\mathrm{C}(A \upharpoonright n | n) \leq k + \mathrm{O}(1)$.

As can be seen from the above example, the complexity of descriptions of prefixes of A depends on how fast we introduce the martingales. This is where our orders come into play. Fix a fast-growing computable function f with $f(0) = 0$, to be specified later. We will introduce a new martingale at every position of type $f(k)$, that is, between positions $[f(k), f(k+1))$, we will only diagonalize against $k+1$ martingales, hence by the above discussion, for every $n \in [f(k), f(k+1))$, we have

$$\mathrm{C}(A \upharpoonright n | n) \leq k + \mathrm{O}(1)$$

Thus, if the function f grows faster than the discrete inverse h^{-1} of a given order h, we get

$$\mathrm{C}(A \upharpoonright n | n) \leq h(n) + \mathrm{O}(1)$$

for all n. □

The theorem also holds in a slightly stronger form which states that there is a set A such that the inequality holds for *any* computable order h and for almost all n. See Merkle [Mer08].

TMR = TPR: the averaging technique

It turns out that, perhaps surprisingly, the classes TMR and TPR coincide. This fact was stated explicitly in Merkle et al. [MMN+06], but is easily derived from the ideas introduced in Buhrman et al. [BvMR+00]. We present the main ideas of their proof as we will later need them.

Theorem 3.6. *Let $b = (d, \pi)$ be a total computable permutation strategy. There exists a total computable martingale d such that* $\mathrm{Succ}(b) \subseteq \mathrm{Succ}(d)$.

This theorem states that total permutation strategies are no more powerful than total monotonic strategies, which obviously entails TMR = TPR. Before we can prove this, we first need a definition.

Definition 3.7. *Let $b = (d, \pi)$ be a total injective strategy. Let $w \in \{0,1\}^{<\infty}$. We can run the strategy b on w as if it were an element of $\{0,1\}^\infty$, stopping the game when b asks to bet on a bit on position outside w. This game is of course finite (for a given w) since at most $|w|$ bets can be made. We define $\hat{b}(w)$ to be the capital of b at the end of this game. Formally: $\hat{b}(w) = d\left(w_{\pi(0)} \ldots w_{\pi(N-1)}\right)$ where N is the least integer such that $\pi(N) \geq |w|$.*

Note that despite the notation, $\hat{b}$ is a single function, not a pair of functions like b. Also note that if $b = (d, \pi)$ is a total computable injection martingale, $\hat{b}$ is total computable.

If $\hat{b}$ was itself a monotonic martingale, assuming the "savings property" described below would be enough to prove Theorem 3.6. In general however, $\hat{b}$ is not even a martingale, as can be seen from the following example: Suppose the starting capital is $d(\epsilon) = 1$, the scan rule is $\pi(0) = 1$, $\pi(1) = 0$ and the betting strategy is described by $d(0) = 2$, $d(00) = 4$, $d(01) = 0$ and by $d(1) = 0$, $d(10) = d(11) = 0$ — that is, d first visits the bit in position 1, betting everything on the value 0, then visits the bit in position 0, again betting everything on the value 0. We then have

$$\frac{\hat{b}(00) + \hat{b}(01)}{2} = \frac{4+0}{2} = 2 \neq 1 = \hat{b}(0),$$

which shows that $\hat{b}$ is not a martingale.

The trick is, given a betting strategy b and a word w, to look at the *expected value* of b on w, i.e., look at the mathematical expectation of $b(w')$ for large enough extensions w' of w. Specifically, given a total betting strategy $b = (d, \pi)$ and a word w of length n, we take an integer M large enough to have

$$\pi([0, \ldots, M-1]) \cap [0, \ldots, n-1] = \pi(\mathbb{N}) \cap [0, \ldots, n-1]$$

(i.e. the strategy b will never bet on a bit on position less than n after the M-th move), and define:

$$\mathrm{Av}_b(w) = \frac{1}{2^M} \sum_{\substack{w \sqsubseteq w' \\ |w'|=M}} \hat{b}(w')$$

Proposition 3.8 (Buhrman et al. [BvMR$^+$00], Kastermans-Lempp [KL10])**.** *The following statements hold.*

(i) The quantity $\mathrm{Av}_b(w)$ *(defined above) is well-defined i.e. does not depend on* M *as long as it satisfies the required condition.*

(ii) For a total injective strategy b, Av_b *is a martingale.*

(iii) For a given injective strategy b *and a given word* w *of length* n, $\mathrm{Av}_b(w)$ *can be computed if we know the set* $\pi(\mathbb{N}) \cap [0, \ldots, n-1]$. *In particular, if* b *is a total computable permutation strategy, then* Av_b *is total computable.*

As Buhrman et al. [BvMR$^+$00] explained, it is not true in general that if a total computable injective strategy b succeeds against a sequence A, then Av_b also succeeds on A. However, this can be dealt with using the well-known "savings trick". Suppose we are given a martingale d with initial capital, say, 1. Consider the

variant d' of d that does the following: when run on a given sequence A, d' initially plays exactly as d. If at some stage of the game d' reaches a capital of 2 or more, it then puts half of its capital on a "bank account", which will never be used again. From that point on, d' bets half of what d does, i.e. start behaving like $d/2$ (plus the saved capital). If later in the game the "non-saved" part of its capital reaches 2 or more, then half of it is placed on the bank account and then d' starts behaving like $d/4$, and so on.

For every martingale d' that behaves as above (i.e. saves half of its capital as soon as it exceeds twice its starting capital), we say that d' has the "savings property". It is clear from the definition that if d is computable, then so is d', and moreover d' can be uniformly computed given an index for d. Moreover, if for some sequence A one has

$$\limsup_{n\to+\infty} d(A\upharpoonright n) = +\infty$$

then

$$\lim_{n\to+\infty} d'(A\upharpoonright n) = +\infty$$

which in particular implies $\mathrm{Succ}(d) \subseteq \mathrm{Succ}(d')$ (it is easy to see that in fact equality holds). Thus, whenever one considers a martingale d, one can assume without loss of generality that it has the savings property (as long as we are only interested in the success set of martingales, not in the growth rate of their capital). The key property (for our purposes) of savings martingales is the following.

Lemma 3.9. *Let $b = (d, \pi)$ be a total injective strategy such that d has the savings property. Let $d' = \mathrm{Av}_b$. Then* $\mathrm{Succ}(b) \subseteq \mathrm{Succ}(d')$.

Proof. Suppose that $b = (d, \pi)$ succeeds on a sequence A. Since d has the savings property, for arbitrarily large k there exists a finite prefix $A\upharpoonright n$ of A such that a capital of at least k is saved during the finite game of b against A. We then have $\hat{b}(w') \geq k$ for all extensions w' of $A\upharpoonright n$ (as a saved capital is never used), which by definition of Av_b implies $\mathrm{Av}_b(A\upharpoonright m) \geq k$ for all $m \geq n$. Since k can be chosen arbitrarily large, this finishes the proof. □

Now the proof of Theorem 3.6 is as follows. Let $b = (d, \pi)$ be a total computable permutation strategy. By the above discussion, let d' be the savings version of d, so that $\mathrm{Succ}(d) \subseteq \mathrm{Succ}(d')$. Setting $b' = (d', \pi)$, we have $\mathrm{Succ}(b) \subseteq \mathrm{Succ}(b')$. By Proposition 3.8 and Lemma 3.9, $d'' = \mathrm{Av}_{b'}$ is a total computable martingale, and

$$\mathrm{Succ}(b) \subseteq \mathrm{Succ}(b') \subseteq \mathrm{Succ}(d'')$$

as wanted.

The strength of injective strategies: the class TIR

Theorem 3.6 implies in particular that the class of computably random sequences (i.e. the class TMR) is closed under computable permutations of the bits. We now see that this result does not extend to computable injections.

Theorem 3.10. *Let $A \in \{0,1\}^\infty$. Let $\{n_k\}_{k\in\mathbb{N}}$ be a computable sequence of integers such that $n_{k+1} \geq 2n_k$ for all k. Suppose that A is such that*

$$\mathrm{C}\left(A \upharpoonright n_k | k\right) \leq \log n_k - 3\log\log n_k$$

for infinitely many k. Then $A \notin$ TIR.

Proof. Let A be a sequence satisfying the hypothesis of the theorem. Assuming, without loss of generality, that $n_0 = 0$, we partition $\mathbb{N}$ into a sequence of intervals $I_0, I_1, I_2, \ldots$ where $I_k = [n_k, n_{k+1})$. Notice that we have for all k:

$$\mathrm{C}(A \upharpoonright I_k | k) \leq \mathrm{C}(A \upharpoonright n_{k+1} | k+1) + \mathrm{O}(1)$$

By the assumption of the theorem, the right-hand side of the above inequality is bounded by $\log n_{k+1} - 3\log\log n_{k+1}$ for infinitely many k.

Additionally, we have $|I_k| = n_{k+1} - n_k$ which by assumption on the sequence n_k implies $|I_k| \geq n_{k+1}/2$, and hence $\log|I_k| \geq \log n_{k+1} + \mathrm{O}(1)$ and $\log\log|I_k| \geq \log\log n_{k+1} + \mathrm{O}(1)$. It follows that

$$\mathrm{C}\left(A \upharpoonright I_k | k\right) \leq \log|I_k| - 3\log\log|I_k| - \mathrm{O}(1)$$

for infinitely many k, hence

$$\mathrm{C}\left(A \upharpoonright I_k | k\right) < \log|I_k| - 2\log\log|I_k|$$

for infinitely many k.

Let us call S_k the set of strings w of length $|I_k|$ such that

$$\mathrm{C}\left(w || |I_k|\right) < \log|I_k| - 2\log\log|I_k|,$$

which implies that $A \upharpoonright I_k \in S_k$ for infinitely many k. By the standard counting argument, there are at most

$$s_k = \left\lceil 2^{\log|I_k| - 2\log\log|I_k|} \right\rceil = \left\lceil \frac{|I_k|}{\log^2(|I_k|)} \right\rceil$$

strings in S_k. For every k, we split I_k into s_k consecutive disjoint intervals of equal length, see Figure 3.2 (if s_k does not exactly divide $|I_k|$, then put the excess bits at

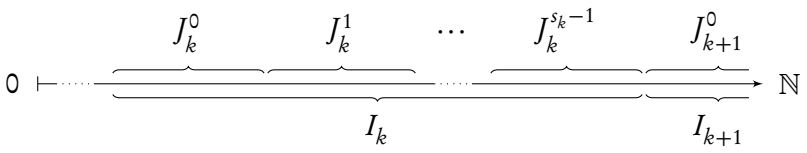

Figure 3.2: The partition into intervals.

the end of I_k into a "garbage set", which we will ignore from now on except for the fact that we account for it in the calculations in the next paragraph).

$$I_k = J_k^0 \cup J_k^1 \cup \ldots \cup J_k^{s_k-1}$$

We design a betting strategy as follows. We start with a capital of 2. We then reserve for each k an amount of $1/(k+1)^2$ to be bet on the bits in positions in I_k (this way, the total amount we distribute is smaller than 2), and we split this evenly between the J_k^i, i.e. we reserve an amount of $\frac{1}{(s_k+1)\cdot(k+1)^2}$ for every J_k^i. We then enumerate the sets S_k in parallel. Whenever the e-th element w_k^e of some S_k is enumerated, we see w_k^e as a possible candidate to be equal to $A \upharpoonright I_k$, and we bet the reserved amount $\frac{1}{(s_k+1)\cdot(k+1)^2}$ on the fact that $A \upharpoonright I_k$ coincides with w_k^e on the bits whose positions are in J_k^e. If we are successful (this in particular happens whenever $w_k^e = A \upharpoonright I_k$), our reserved capital for this J_k^e is multiplied by $2^{|J_k^e|}$, i.e. we now have for this J_k^e, a capital of

$$\frac{1}{(s_k+1)\cdot(k+1)^2} \cdot 2^{|I_k|/(s_k+1)}$$

Replacing s_k by its value (and remembering that $|I_k| \geq 2^{k-\mathrm{O}(1)}$), an elementary calculation shows that this quantity is greater than 1 for almost all k. Thus, our betting strategy succeeds on A. Indeed, for infinitely many k, $A \upharpoonright I_k$ is an element of S_k, hence for some e we will be successful in the above sub-strategy, making an amount of money greater than 1 for infinitely many k, hence our capital tends to infinity throughout the game. Finally, it is easy to see that this betting strategy is total: it simply is a succession of doubling strategies on an infinite c.e. set of words, and it is injective as the J_k^e form a partition of $\mathbb{N}$, and the order of the bits we bet on is independent of A (in fact, we see our betting strategy succeeds on *all* sets A satisfying the hypothesis of the theorem). □

As an immediate corollary, we get the following.

Corollary 3.11. *If for a sequence A there is a constant c such that we have for all n that $\mathrm{C}(A \upharpoonright n|n) \leq \log n - 4\log\log n + c$, then $A \notin \mathsf{TIR}$.*

Another interesting corollary of our construction is that the class of all computable sequences can be covered by a single total computable injective strategy.

Corollary 3.12. *There exists a single total computable injective strategy which succeeds against all computable elements of* $\{0,1\}^\infty$.

Proof. This is because, as we explained above, the strategy we construct in the proof of Theorem 3.10 succeeds against *every* sequence A such that $\mathrm{C}\left(A \upharpoonright n_k | k\right) \leq \log n_k - 3\log\log n_k$ for infinitely many k. This in particular includes all computable sequences A, for which $\mathrm{C}\left(A \upharpoonright n_k | k\right) = \mathrm{O}(1)$. □

The lower bound on Kolmogorov complexity given in Theorem 3.10 is quite tight, as witnessed by the following theorem.

Theorem 3.13. *For every computable order h there is a sequence $A \in \mathsf{TIR}$ such that* $\mathrm{C}(A \upharpoonright n \mid n) \leq \log n + h(n) + \mathrm{O}(1)$. *In particular, we have*

$$\mathrm{C}(A \upharpoonright n) \leq 2\log n + h(n) + \mathrm{O}(1).$$

Proof. The proof is a modification of the proof of Theorem 3.4. This time, we want to diagonalize against all *non-monotonic* total computable injective betting strategies. Like in the proof of Theorem 3.4, we add them one by one, discarding the partial strategies. However, to achieve the construction of A by diagonalization, we will diagonalize against the average martingales of the strategies we consider. As explained on page 27, it suffices to diagonalize against all total computable injective strategies that have the savings property, hence defeating Av_b is enough to defeat b (by Lemma 3.9). The proof thus goes as follows:

Fix a fast growing computable function f, to be specified later. We start with a martingale $D_0 = 1$ (the constant martingale equal to 1) and $w_0 = \epsilon$. For all k we do the following. Assume we have constructed a prefix w_k of A of length $f(k)$, and that we are currently diagonalizing against a martingale D_k, so that $D_k(w_k) < 2$. We then enumerate a new partial computable injective betting strategy b.

The strategy b could be total or not, and this information must later be available if we want to reproduce the contructed sequence A. Therefore this one extra bit of information must be encoded in the programs describing A, and so increases the Kolmogorov complexity of A.

If b is not total we set $D_{k+1} = D_k$. Otherwise, we set $\widetilde{d}_{k+1} = \mathrm{Av}_b$ and let d_{k+1} be a modified version of $\widetilde{d}_{k+1}$ that doesn't bet before the next position $|w_k|$. We then compute a positive rational α_{k+1} such that $(D_k + \alpha_{k+1} d_{k+1})(w_k) < 2$, and finally set $D_{k+1} = D_k + \alpha_{k+1} d_{k+1}$.

Then, we define w_{k+1} to be the extension of w_k of length $f(k+1)$ by the usual diagonalization against D_{k+1}, maintaining the inequality $D_{k+1}(u) < 2$ for

all prefixes u of w_{k+1}. The infinite sequence A obtained this way defeats all the average martingales of all total computable injective strategies, hence by Lemma 3.9, $A \in \mathsf{TIR}$.

It remains to show that A has low Kolmogorov complexity. Suppose we want to describe $A \upharpoonright n$ for some $n \in [f(k), f(k+1))$. This can be done by giving n, the subset of $\{0, \dots, k\}$ (of complexity at most $k + \mathrm{O}(1)$) corresponding to the indices of the total computable injective strategies among the first k partial computable ones, and by giving the restriction of D_{k+1} to words of length at most n. From all this, $A \upharpoonright n$ can be reconstructed following the above construction. It remains to evaluate the complexity of the restriction of D_{k+1} to words of length at most n. We already know the total computable injective strategies $b_0, \dots, b_k$ that are being considered in the definition of D_{k+1}. For all i, let π_i be the injection associated with b_i. We need to compute, for all $0 \leq i \leq k$, the martingale $d_i = \mathrm{Av}_{b_i}$ on words of length at most n. By Proposition 3.8, this can be done knowing $\pi_i(\mathbb{N}) \cap [0, \dots, n-1]$ for all $0 \leq i \leq k$. But if the π_i are known, this set is uniformly c.e. in i and n. Hence, we can enumerate all the sets $\pi_i(\mathbb{N}) \cap [0, \dots, n-1]$ (for $0 \leq i \leq k$) in parallel, and simply give the last pair (i, l) such that l is enumerated into $\pi_i(\mathbb{N}) \cap [0, \dots, n-1]$. Since $0 \leq i \leq k$ and $0 \leq l < n$, this requires $\mathrm{O}(\log k) + \log n$ bits of of information. To sum up, we get

$$\mathrm{C}(A \upharpoonright n | n) \leq k + \mathrm{O}(\log k) + \log n$$

Thus, it suffices to take f growing fast enough to ensure that the term $k + \mathrm{O}(\log k)$ is smaller than $h(n) + \mathrm{O}(1)$. □

3.3 Randomness notions based on partial computable strategies

We now turn our attention to the second row of the table in Figure 3.1, i.e., to those randomness notions that are based on partial computable betting strategies.

The class PMR: partial computable martingales are stronger than total ones

We have seen in the previous section that some sequences in TIR (and a fortiori TPR and TMR) may be of very low complexity, namely logarithmic. This is not the case anymore when one allows partial computable strategies, even monotonic ones.

Theorem 3.14 (Merkle [Mer08]). *If* $\mathrm{C}(A \upharpoonright n) = \mathrm{O}(\log n)$ *then* $A \notin \mathsf{PMR}$.

However, the next theorem, proven by Muchnik, shows that allowing slightly super-logarithmic growth of the Kolmogorov complexity is enough to construct a sequence in PMR.

Theorem 3.15 (Muchnik et al. [MSU98])**.** *For every computable order h there is a sequence $A \in \mathsf{PMR}$ such that, for all $n \in \mathbb{N}$,*

$$\mathrm{C}(A \upharpoonright n | n) \leq h(n) \log n + \mathrm{O}(1).$$

Proof. The proof is almost identical to the proof of Theorem 3.4. The only difference is that we insert *all* partial computable martingales one by one, and diagonalize against their weighted sum as before.

It may happen however, that at some stage of the construction, one of the martingales becomes undefined. We will then ignore this particular martingale from that point on. Again, as in the proof of Theorem 3.13, we will later need this information if we want to reproduce the contructed sequence A. So, as before, this information must be encoded in the programs describing A, and so increases the Kolmogorov complexity of A.

Call A the sequence we obtain by this construction. We want to describe $A \upharpoonright n$. To do so, we need to specify n, and, out of the k partial computable martingales that are inserted before stage n, which ones have diverged, and at what stage, hence an information of $\mathrm{O}(k \log n)$ (giving the position where a particular martingale diverges costs $\mathrm{O}(\log n)$ bits, and there are k martingales). Since we can insert martingales as slowly as we like (following some computable order), the complexity of $A \upharpoonright n$ given n can be taken to be smaller than $h(n) \log n + \mathrm{O}(1)$ (where h is a computable order, fixed before the construction of A). □

Again, the theorem also holds in the slightly stronger form where the inequality is true for all computable orders, see Merkle [Mer08].

The class PPR

In the case of total strategies, allowing permutation gives no real additional power, as $\mathsf{TMR} = \mathsf{TPR}$. Very surprisingly, Muchnik showed that in the case of partial computable strategies, permutation strategies are a considerable improvement over monotonic ones, as witnessed by the following theorem (quite a contrast to Theorem 3.15!).

Theorem 3.16 (Muchnik [MSU98])**.** *If there is a computable order h such that for all n we have $\mathrm{K}(A \upharpoonright n) \leq n - h(n) - \mathrm{O}(1)$, then $A \notin \mathsf{PPR}$.*

The theorem by Muchnik [MSU98] actually deals with "a priori entropy" but easily implies the above statement.

We can now see that Kolmogorov-Loveland randomness is quite close to Martin-Löf randomness: Comparing Theorem 2.1 with Theorem 3.16 shows that PPR is not far away from MLR. Since KLR lies between MLR and PPR, it has to be close to MLR as well.

Theorem 3.17. *For every computable order h there is a sequence $A \in \mathsf{PPR}$, such that there are infinitely many n where $\mathrm{C}(A \upharpoonright n|n) < h(n)$.*

Furthermore, if we have an infinite computable set $S \subseteq \mathbb{N}$, we can choose the infinitely many lengths n such that they all are contained in S.

Before we can prove the theorem we need the following lemma and corollary.

Lemma 3.18. *Let d be a partial computable martingale. Let $\mathscr{C}$ be an effectively closed subset of $\{0,1\}^\infty$ (where "effectively closed" is another expression for being a Π^0_1 class). Suppose that d is total on every element of $\mathscr{C}$. Then there exists a total computable martingale d' such that $\mathrm{Succ}(d) \cap \mathscr{C} = \mathrm{Succ}(d') \cap \mathscr{C}$.*

Proof. The idea of the proof is simple: the martingale d' will try to mimic d while enumerating the complement $\mathscr{U}$ of $\mathscr{C}$. If at some stage a cylinder $[w]$ is covered by $\mathscr{U}$, then d will be passive (i.e. defined but constant) on the sequences extending w. As we do not care about the behavior of d' on $\mathscr{U}$ (as long as it is defined), this will be enough to get the conclusion.

Let $d, \mathscr{C}$ be as above. We build the martingale d' on words by induction. Define $d'(\epsilon) = d(\epsilon)$ (here we assume without loss of generality that $d(\epsilon)$ is defined, otherwise there is nothing to prove). During the construction, some words will be marked as inactive, on which the martingale will be passive; initially, there is no inactive word. On active words w, we will have $d(w) = d'(w)$.

Suppose for the sake of the induction that $d'(w)$ is already defined. If w is marked as inactive, we mark $w0$ and $w1$ as inactive, and set $d(w0) = d(w1) = d(w)$. Otherwise, by the induction hypothesis, we have $d(w) = d'(w)$. We then run in parallel the computation of $d(w0)$ and $d(w1)$, and enumerate the complement $\mathscr{U}$ of $\mathscr{C}$ until one of the two above events happens:

(a) $d(w0)$ and $d(w1)$ become defined. Then set $d'(w0) = d(w0)$ and $d'(w1) = d(w1)$

(b) The cylinder $[w]$ gets covered by $\mathscr{U}$. In that case, mark $w0$ and $w1$ as inactive and set $d'(w0) = d'(w1) = d'(w)$

Note that one of these two events *must* happen: indeed, if $d(w0)$ and $d(w1)$ are undefined (remember that by the definition of a martingale, Definition 3.1, that

they are either both defined or both undefined), then this means that d diverges on *any* element of $[w0] \cup [w1] = [w]$. Hence, by assumption, $[w] \cap \mathscr{C} = \emptyset$, i.e. $[w] \subseteq \mathscr{U}$ and we will see this after finitely many steps. It remains to verify that $\mathrm{Succ}(d) \cap \mathscr{C} = \mathrm{Succ}(d') \cap \mathscr{C}$. Let $A \in \mathscr{C}$. This implies that for all $w \sqsubseteq A$, we never have $[w] \subseteq \mathscr{U}$. So during the construction of d' on A, we will always be in case (a), hence we will have for all n, $d(A \upharpoonright n) = d'(A \upharpoonright n)$. The result follows immediately. □

Corollary 3.19. *Let $b = (d, \pi)$ be a partial computable permutation strategy (resp. injective strategy). Let $\mathscr{C}$ be an effectively closed subset of $\{0,1\}^\infty$. Suppose that b is total on every element of $\mathscr{C}$. Then there exists a total computable permutation strategy (resp. injective strategy) b' such that* $\mathrm{Succ}(b) \cap \mathscr{C} = \mathrm{Succ}(b') \cap \mathscr{C}$.

Proof. This follows from the fact that the image or pre-image of an effectively closed set under a computable permutation (resp. computable injection) of the bits is itself a closed set: take $b = (d, \pi)$ and $\mathscr{C}$ as above. Let π' be the map induced on $\{0,1\}^\infty$ by π, i.e. the map defined for all $A \in \{0,1\}^\infty$ by

$$\pi'(A) = A(\pi(0))A(\pi(1))A(\pi(2))\dots$$

For any given sequence $A \in \mathscr{C}$, b succeeds on A if and only if d succeeds on $\pi'(A)$. As $\pi'(A) \in \pi'(\mathscr{C})$, and $\pi'(\mathscr{C})$ is an effectively closed set, by Lemma 3.18, there exists a total martingale d' such that $\mathrm{Succ}(d) \cap \pi'(\mathscr{C}) = \mathrm{Succ}(d') \cap \pi'(\mathscr{C})$. Thus, d' succeeds on $\pi'(A)$, or equivalently, $b' = (d', \pi)$ succeeds on A. Thus b' is as desired. □

Proof of Theorem 3.17. Again, this proof is a variant of the proof of Theorem 3.4: we add strategies one by one, diagonalizing, at each stage, against a finite weighted sum of total monotonic strategies (i.e. martingales). Of course, not all strategies have this property, but we can reduce to this case using the techniques we presented above. Suppose that in the construction of our sequence A, we have already constructed an initial segment w_k, and that up to this stage we played against a weighted sum of k total martingales

$$D_k = \sum_{i=1}^{k} \alpha_i \, d_i$$

where the d_i are total computable martingales, ensuring that $D_k(u) < 2$ for all prefix u of w. Suppose we want to introduce a new strategy $b = (d, \pi)$. There are three cases:

Case 0: the new strategy is not valid, i.e. π is not a permutation. In this case, we ignore b from now on, i.e. we set $w_{k+1} = w_k$, $d_{k+1} = 0$ (the zero martingale), and

$D_{k+1} = D_k + d_{k+1} = D_k$. As in the proofs of Theorems 3.13 and 3.15, this piece of information will be needed to reproduce the contructed sequence A later, and therefore increases the Kolmogorov complexity of A accordingly.

Case 1: the strategy b is indeed a partial computable permutation strategy, and there exists an extension w' of w such that $D_k(u) < 2$ for all prefixes u of w', and b diverges on w'. In this case, we simply take w' as our new prefix of A, as it both diagonalizes against D, and defeats b (since b diverges on w', it will not win against *any* possible extension of w'). We can thus ignore b from that point on, so we set $w_{k+1} = w'$, $d_{k+1} = 0$ and $D_{k+1} = D_k + d_{k+1} = D_k$.

Case 2: if we are not in one of the two previous cases, this means that our strategy $b = (d, \pi)$ is a partial computable permutation strategy, and that b is total on the whole Π^0_1 class

$$\mathscr{C}_k = [w_k] \cap \{X \in \{0,1\}^\infty \mid \forall n\, D_k(X \upharpoonright n) < 2\}$$

Thus, by Lemma 3.19, there exists a total computable permutation strategy b' such that $\mathrm{Succ}(b) \cap \mathscr{C}_k = \mathrm{Succ}(b') \cap \mathscr{C}_k$. And by Theorem 3.6, there exists a total computable martingale d'' such that $\mathrm{Succ}(b') \subseteq \mathrm{Succ}(d'')$. Thus, we can replace b by d'', and defeating d'' will be enough to defeat b as long as the sequence we construct is in $\mathscr{C}_k$. We thus set $d_{k+1} = d''$, $w_{k+1} = w_k$ and

$$D_{k+1} = \sum_{i=1}^{k+1} \alpha_i\, d_i$$

where α_{k+1} is sufficiently small to have $D_{k+1}(w_{k+1}) < 2$.

Once we have added a new monotonic martingale, we (as usual) computably find an extension w'' of w_{k+1}, ensuring that $D_{k+1}(u) < 2$ for all prefix u of w'', taking w'' long enough to have $\mathrm{C}\left(w'' \middle|\middle| |w''|\right) \leq h(|w''|)$. We then set $w_{k+1} = w''$, then add a $k+2$-th strategy and so on.

Note that since w'' can be chosen arbitrarily large, if we have fixed a computable subset S of $\mathbb{N}$, we can also ensure that $|w''|$ belong to S if we like.

It is clear that the infinite sequence A constructed via this process satisfies

$$\mathrm{C}(A \upharpoonright n | n) \leq h(n)$$

for infinitely many n (and, since Case 2 happens infinitely often, if we fix a given computable set S, we can ensure that infinitely many of such n belong to S). To see that it belongs to PPR, we notice that since for all k, $D_{k+1} \geq D_k$ and $w_k \sqsubseteq w_{k+1}$, we have $\mathscr{C}_{k+1} \subseteq \mathscr{C}_k$ and thus $A \in \bigcap_k \mathscr{C}_k$. Now, given a partial computable permutation

	monotonic		permutation		injection
total	TMR	$=$	TPR	$\supsetneq$	TIR
	$\rotatebox{90}{$\subsetneq$}$		$\rotatebox{90}{$\subsetneq$}$		$\rotatebox{90}{$\subsetneq$}$
partial	PMR	$\supsetneq$	PPR	$\supsetneq$	PIR

Figure 3.3: Assembled class inclusion results

strategy $b = (d, \pi)$, let k be the stage where b was considered, and replaced by the martingale d_k (according to the applicable case among the three given cases). Since by construction of A, d_{k+1} does not win against A and by definition of d_k, $\mathrm{Succ}(b) \cap \mathscr{C}_k \subseteq \mathrm{Succ}(d_k) \cap \mathscr{C}_k$, it follows that $A \notin \mathrm{Succ}(b)$. □

Now that we have assembled all our tools, we can easily prove the desired results.

Theorem 3.20. *The following statements hold.*

(i) PPR $\not\subseteq$ TIR

(ii) TIR $\not\subseteq$ PMR

(iii) PMR $\not\subseteq$ PPR

In particular, the following statements follow.

(iv) TPR $\not\subseteq$ TIR

(v) PPR $\not\subseteq$ PIR

(vi) TIR $\not\subseteq$ PPR

(vii) TIR $\not\subseteq$ PIR

(viii) TPR $\not\subseteq$ PPR

(ix) TMR $\not\subseteq$ PMR

From these results it easily follows that in Figure 3.3 no inclusion holds except those indicated and those implied by transitivity.

Proof. (i): Choose a computable sequence $\{n_k\}_k$ fulfilling the requirements of Theorem 3.10 such that $\mathrm{C}(k) \leq \log\log n_k$ for all k. Then the set $S := \{n_0, n_1, \ldots\}$ is computable. Use Theorem 3.17 to construct a sequence $A \in$ PPR such that

$C(A \upharpoonright n \mid n) < \log\log n$ at infinitely many places in S. We then have for infinitely many k

$$C(A \upharpoonright n_k \mid k) \leq C(A \upharpoonright n_k) \leq C(A \upharpoonright n_k \mid n_k) + 2\log\log n_k \leq 3\log\log n_k,$$

where the factor 2 is caused by overhead for coding pairs. It then follows from Theorem 3.10 that A cannot be in TIR.
(ii): Follows immediately from Theorems 3.13 and 3.14.
(iii): Follows immediately from Theorems 3.15 and 3.16.
(iv)–(ix): These statements follow from the previous three statements using a common pattern: If in any of the statements (i) to (iii) we replace the first set with a superset or the second set with a subset then the resulting non-inclusion statement is obviously still true.

As an example of this common pattern we prove (viii): By (iii) we have that $\mathsf{PMR} \not\subseteq \mathsf{PPR}$, which together with the fact that $\mathsf{TPR} \supseteq \mathsf{PMR}$ implies the desired result that $\mathsf{TPR} \not\subseteq \mathsf{PPR}$. □

CHAPTER 4

Traceability and complexity

The notion of traceability was first introduced by Terwijn and Zambella [TZ01] and has received a significant amount of attention in the last years. The general idea is to look at sets that are nearly but not quite computable. More explicitly, if a set is computable, obviously all functions computable in that set are computable, too. Similarly, a set is nearly computable, or traceable, if all functions computable in the set are nearly computable; where nearly computable means that for every input provided to the function we can in some sense effectively generate a list of candidate values for the image of that input under the function, where the list of candidates is in some sense small.

The various notions of a traceable set have received a significant amount of attention in the area of algorithmic randomness. On the one hand, traceability naturally comes up in connection with lowness notions, as is exemplified in the work of Terwijn and Zambella [TZ01] on Schnorr randomness and, more recently, the attempts to characterize lowness for Martin-Löf randomness and the equivalent notion of K-triviality by an appropriate version of jump traceability [BDG09, CDG08, HKM09]. On the other hand, traceability has been shown [KHMS, HM10] to interact informatively with classical notions from computability theory such as diagonally non-computable sets and with notions such as autocomplex that are defined in terms of Kolmogorov complexity of initial segments of sets.

In this chapter, we systematically investigate several variations of notions of traceability. We review standard notions of traceability and some basic results on them, giving simplified or at least more direct proofs than in the current literature, which in particular are meant to provide an intuitive picture of why the stated relations hold. One of our aims is to give a unified view of notions and results that appear in the literature, and for example we argue that a recent result on anticomplex sets by Franklin et al. [FGSW] can be seen as a variant of results on the relations between notions of complexity and i.o. traceability [KHMS].

We also introduce new notions of traceability such as infinitely often versions of jump traceability and derive an interesting collapse result. Finally, we give a result about polynomial-time bounded notions of traceability and complexity that shows that in principle the equivalences derived so far can be transferred to the time-bounded setting.

4.1 Traceability

The various traceability notions considered in the sequel are either well-known or have at least been considered implicitly in the literature, except for, to the best of our knowledge, the infinitely often versions of jump traceable and strongly jump traceable introduced in Definition 4.11 below.

Definition 4.1. *A* trace *is a sequence $(T_n)_n$ of sets. A trace $(T_n)_n$* is a trace for *a partial function f, if $f(n) \in T_n$ holds for all n such that $f(n)$ is defined. A trace $(T_n)_n$ is an* i.o. trace for *a partial function f, if there are infinitely many n such that $f(n) \in T_n$.*

We will also say, for short, that a trace traces or i.o. traces a partial function f, in case the trace is a trace or an i.o. trace, respectively, for f. For the traces $(T_n)_n$ considered in the sequel, the sets T_n will always be finite.

Recall that $W_0, W_1, \ldots$ is the standard acceptable numbering of all computably enumerable (c.e.) sets, i.e., W_e is the domain of the e-th partial computable function φ_e.

Definition 4.2. *For a function h, a trace $(T_n)_n$ is* h-bounded, *if $\#T_n \leq h(n)$ holds for all n.*

A trace $(T_n)_n$ is computably enumerable (c.e.) *if there is a computable function g such that T_n is equal to $W_{g(n)}$ for all n. A trace $(T_n)_n$ is* computable *if there is a computable function g such that T_n is equal to $D_{g(n)}$ for all n, where D_e is the finite set with canonical index e.*

Definition 4.3. *A set A is* c.e. traceable *iff there is a computable order h such that all functions $f \leq_T A$ are traced by an h-bounded c.e. trace $(T_n)_n$. A set A is* c.e. i.o. traceable *iff there is a computable order h such that all functions $f \leq_T A$ are i.o. traced by an h-bounded c.e. trace $(T_n)_n$.*

The concepts of computably traceable *and of* computably i.o. traceable *are defined similarly where in addition the traces are required to be computable instead of being merely c.e.*

For all the concepts introduced above, there are variants where Turing reducibility is replaced by weak truth-table or truth-table reducibility, e.g., we say a set A is c.e. i.o. wtt-traceable *iff there is a computable order h such that all functions $f \leq_{wtt} A$ are i.o. traced by an h-bounded c.e. trace $(T_n)_n$.*

Remark 4.4. *Stephan [Ste10] observed that a set is c.e. traceable if and only if there is a computable function h such that every $f \leq_T A$ satisfies $C(f(n)) < h(n)$ for almost all n. A similar remark holds for partial functions that can be computed with oracle A.*

This characterization has the advantage that it works without defining traces and just uses classical concepts. The disadvantage of this style of characterization is that for other traceability concepts it yields more complicated equivalences; for example the case of computable traceability would require the use of Kolmogorov complexity defined over total machines.

Terwijn and Zambella [TZ01] observed that the notions of computable and c.e. traceability remain the same if one requires in their respective definitions the existence of h-bounded traces not just for a single but for all computable orders h. The corresponding argument extends directly to the notions c.e. and computably wtt-traceable, as well as c.e. and computably tt-traceable, but also to the infinitely often versions of these notions, as is shown in the following remark. For the notion of i.o. c.e. traceable this also follows by Theorem 4.12 below, and, what is more, by Corollaries 4.23 and 4.25 for some notions even the existence of 1-bounded traces of the considered type is equivalent.

Remark 4.5. *A set A is c.e. i.o. traceable if and only if for all computable orders h all functions $f \leq_T A$ are i.o. traced by an h-bounded c.e. trace $(T_n)_n$, and a similar statement holds for the notion computably i.o. traceable, as well as for variants of these notions defined in terms of weak truth-table or truth-table reducibility in place of Turing reducibility.*

The proof uses the same technique as the proof of the analogous everywhere version of the statement [TZ01]. Let us assume we have c.e. i.o. traces bounded by a computable order g and let us construct a c.e. i.o. trace $(S_n)_n$ for some function $f \leq_T A$ bounded by some given computable order h.

Let $\hat{g}(i)$ be the least number n such that $h(n) \geq g(i)$, so $\hat{g}$ is a computable order. Thus, the mapping $\hat{f}$ defined by $i \mapsto (f(0), \ldots, f(\hat{g}(i+1)))$ is Turing-reducible to A and therefore has a trace $(T_i)_i$ with bound g.

Recall that $\hat{g}^{-1}$ denotes the discrete inverse of $\hat{g}$, which here implies that for given n, $\hat{g}^{-1}(n)$ is the largest number i such that $g(i) \leq h(n)$ (so, typically, it will be a slow growing function). Define $(S_n)_n$ by

$$S_n := \{\pi_n(x) \colon x \in T_{\hat{g}^{-1}(n)}\}$$

where π_n is the projection to the n-th coordinate.

$T_{\hat{g}^{-1}(n)}$, and then also S_n, has at most $g(\hat{g}^{-1}(n)) \leq h(n)$ members. For infinitely many i, T_i is right; that is, it contains the correct $\hat{g}(i+1)$-tuple $(f(0), \ldots, f(\hat{g}(i+1)))$. For all such i, let us look at the set P_i of all n such that $\hat{g}^{-1}(n) = i$. For all these n the set $T_{\hat{g}^{-1}(n)} = T_i$ will be the same and contains the described correct $\hat{g}(i+1)$-tuple, which, in turn, contains the correct information about the values of all $f(n)$ with $n \in P_i$.

By the projection π_n this correct information will be put into the set S_n, so S_n will be a correct trace for $f(n)$ for all such n.

To see that overall infinitely many such n exist, we still need to argue that the P_i's cannot be empty. But this is clear since for every i and for $n = \hat{g}(i)$ we have $\hat{g}^{-1}(n) = i$. □

The following theorem is attributed to Kjos-Hanssen et al. [KHMS] by Downey and Hirschfeldt [DH10], however, the assertion of the theorem does not even implicitly appear in the published versions of the corresponding article [KHMS], nor does its proof. Since the proof presented by Downey and Hirschfeldt is via a chain of equivalent statements, we consider it useful and instructive to give a direct argument here. Among the various equivalent definitions for the notion high, we will work with the one according to which a set A is high iff A computes a function that dominates every computable function.

Definition 4.6. *A set A is called* high *iff $A' \geq_T \emptyset''$.*

Proposition 4.7 (Martin [Mar66])**.** *A set A is high if and only if it computes a function that dominates every computable function.*

Theorem 4.8. *The following statements are equivalent.*

(i) The set A is computably i.o. traceable.

(ii) The set A is c.e. i.o. traceable and non-high.

Proof. (i) implies (ii): Any computably i.o. traceable set A is *a fortiori* c.e. i.o. traceable, and is also non-high because given an A-computable function g we obtain a computable function f such that $g(n) \leq f(n)$ for infinitely many n by letting $f(n) = 1 + \max T_n$ where $(T_n)_n$ is a computable trace for g.

(ii) implies (i): Let us assume we have a c.e. i.o. trace $(T_n)_n$ of a function $\ell \leq_{\mathrm{T}} A$. Define the function g such that on argument n one starts to enumerate in parallel the traces T_m for all $m \geq n$ and A-computably recognizes when for the first time for some m_n the correct value $\ell(m_n)$ is enumerated into T_{m_n}, then letting $g(n)$ be the number of computational steps of the enumeration of T_{m_n} that are required to enumerate $\ell(m_n)$. In this situation, let us say that n has found m_n. Since g is computable in A and A is non-high, there is a computable function f that at infinitely many places is larger than g, where in addition we can assume that f is non-decreasing.

We can now get a computable trace $(\widetilde{T}_n)_n$ for ℓ that is correct at infinitely many places as follows: simply let $\widetilde{T}_n$ contain all elements that are enumerated into T_n in at most $f(n)$ steps.

This trace is correct infinitely often. Indeed, *any* n finds *some* m_n, and among the corresponding pairs (n, m_n) there are infinitely many where we have

$$g(n) \leq f(n) \leq f(m_n),$$

where the second inequality uses the assumption that f is non-decreasing. For these pairs, $f(m_n)$ exceeds the number of steps needed to enumerate $\ell(m_n)$ into T_{m_n}, so for these pairs the correct value $\ell(m_n)$ will be a member of $\widetilde{T}_{m_n}$.

Finally observe that in the construction the set $\widetilde{T}_n$ is always contained in T_n, hence any uniform bound h for the trace $(T_n)_n$ is also a uniform bound for the trace $(\widetilde{T}_n)_n$ □

We review the concepts of jump traceable and strongly jump traceable, which can be seen as stricter versions of the notion of c.e. traceable where not only the total but also all partial functions computable in a given set must be traced.

Definition 4.9. *If we say that there is a trace $(T_n)_n$ for a partial function Φ we mean that $\Phi(n) \in T_n$ for all n such that $\Phi(n)$ is defined.*

A set A set is jump traceable *iff there is a computable order h such that for all functions partially computable in A there is an h-bounded c.e. trace.*

A set A is strongly jump-traceable *iff for all computable orders h it holds that for all functions partially computable in A there is an h-bounded c.e. trace.*

Remark 4.10. *It is easier for our purposes to work with the given definition. Alternatively, (strong) jump traceability can be defined by requiring that the diagonal jump function is traceable. For more details, see Downey and Hirschfeldt [DH10].*

It is well-known that the class of strongly jump-traceable sets is a proper subclass of the jump-traceable sets, in fact, the two classes are proper sub- and superclasses, respectively, of the class of K-trivial sets [BDG09, CDG08]. However, for the infinitely often versions of these two notions we get an interesting collapse of traceability notions.

Definition 4.11. *A set A is* i.o. jump-traceable *iff there is a computable order h such that for all functions partially computable in A that have an infinite domain there is an h-bounded c.e. i.o. trace.*

A set A is strongly i.o. jump-traceable *iff for all computable orders h it holds that for all functions partially computable in A that have an infinite domain there is an h-bounded c.e. i.o. trace.*

Theorem 4.12. *The following statements are equivalent.*

(i) The set A is strongly i.o. jump-traceable.

(ii) The set A is i.o. jump-traceable.

(iii) The set A is c.e. i.o. traceable.

Proof. By definition, (i) implies (ii) and (ii) implies (iii), so it suffices to show that not strongly i.o. jump traceable implies not c.e. i.o. traceable. So let A be a set that computes a *partial* function f that for some computable order h_0 cannot be i.o. traced by any h_0-bounded c.e. trace. We show that for any given computable order h there is an *A-computable* function that cannot be i.o. traced by any h-bounded c.e. trace. Fix an appropriate effective enumeration $(T^0_n)_{n\in\mathbb{N}},(T^1_n)_{n\in\mathbb{N}},\dots$ of all h-bounded c.e. traces, e.g., let T^e_n be the subset of the n-th row of W_e that contains the first $h(n)$ elements that are enumerated into this row. Furthermore, let S_n be the union of all T^e_i where $i < n$ and $e < n$ and observe that this way the cardinality of S_n is at most $c(n) = n^2 h(n)$. For all n, let T_n be equal to S_m where m is maximum such that $c(m) \leq h_0(n)$ and call the trace $(T_n)_n$ the universal h_0-bounded trace, which by construction is indeed h_0-bounded, hence does not i.o. trace f. Hence for almost all m such that $f(m)$ is defined, we have $f(m) \notin T_m$. So we obtain an A-computable function as required by mapping n to a value of the form $f(m)$ where m is chosen large enough to ensure $c(n) \leq h_0(m)$ and such that $f(m)$ is defined. Such a value can be found by dovetailing. □

In order to render the statement of results in Section 4.4 and 4.5 more intuitive, we introduce the following alternate notation for notions of not being traceable.

Definition 4.13. *A set* avoids c.e. traces *if the set is not c.e. i.o. traceable and the set* i.o. avoids c.e. traces *if it is not c.e. traceable. Similarly, a set* tt-avoids c.e. traces *if the set is not c.e. i.o. tt-traceable, and further notions such as i.o. wtt-avoiding computable traces are defined in the same manner.*

4.2 Autocomplex and complex sets

The notions of complexity and autocomplexity were first defined in an article by Kanovich [Kan70], where he showed that autocomplex sets are Turing complete and complex sets are wtt-complete for the class of c.e. sets.

Definition 4.14. *A set A is* complex *if there is a computable order h such that for all n, it holds that* $C(A \upharpoonright n) \geq h(n)$.

A set A is called autocomplex, *if there is an A-computable order h such that for all n, it holds that* $C(A \upharpoonright n) \geq h(n)$.

We omit the straightforward proof of the following known fact [DH10, KHMS]. Note that by the standard proof of Proposition 4.15 it is immediate that all the functions g that occur in the proposition can be assumed to be orders.

Proposition 4.15. *The following statements are equivalent for a set A.*

(i) *A is complex.*

(ii) *There is a computable function g such that for all n, $\mathrm{C}(A \upharpoonright g(n)) \geq n$.*

(iii) *There is a function $g \leq_{\mathrm{tt}} A$ such that for all n, $\mathrm{C}(g(n)) \geq n$.*

(iv) *There is a function $g \leq_{\mathrm{wtt}} A$ such that for all n, $\mathrm{C}(g(n)) \geq n$.*

Similarly, the following statements are equivalent for A.

(i) *A is autocomplex.*

(i) *There is an A-computable function g such that for all n, $\mathrm{C}(A \upharpoonright g(n)) \geq n$.*

(i) *There is an A-computable function g such that for all n, $\mathrm{C}(g(n)) \geq n$.*

In Section 4.5, we will see that it is interesting to consider variants of the notions autocomplex and complex where the condition $\mathrm{C}(A \upharpoonright g(n)) \geq n$ is not required for all but just for infinitely many n. In connection with the following definition, note that the notion of *not* being i.o. complex has been introduced by Franklin et al. [FGSW] under the name of anticomplex.

Definition 4.16. *A set A is* i.o. complex *iff there is a computable order g such that for infinitely many n, we have $\mathrm{C}(A \upharpoonright g(n)) \geq n$.*

A set A is i.o. autocomplex *iff there is an A-computable order g such that for infinitely many n, we have $\mathrm{C}(A \upharpoonright g(n)) \geq n$.*

The equivalent characterizations of complex suggest different ways to define i.o. complex (and similar remarks can be made for the notion i.o. autocomplex). However, it would neither be equivalent nor even make sense to define i.o. complexity by requiring that there is some computable order h such that for infinitely many n it holds that $\mathrm{C}(A \upharpoonright n) \geq h(n)$, because for slowly growing h (e.g., the map $n \mapsto \log\log n$) this inequality is satisfied for infinitely many initial segments of any set A, simply because a code for $A \upharpoonright n$ is always also a code for n. In Section 4.7, we will see that equivalent definitions in this style are still possible by considering specific variants of Kolmogorov complexity. Furthermore, the two following propositions show that in the defining condition $\mathrm{C}(A \upharpoonright g(n)) \geq n$ of i.o. autocomplexity and i.o. complexity the lower bound n can equivalently be replaced by a wide range of lower bounds in case g may depend on this bound.

Proposition 4.17. *The following assertions are equivalent.*

(i) *The set A is i.o. autocomplex.*

(ii) *There is a computable order h and an A-computable function g such that there are infinitely many n where $\mathrm{C}(A \upharpoonright g(n)) \geq \mathrm{C}(n) + h(n)$.*

(iii) For every A-computable order h there is an A-computable function g such that there are infinitely many n where $\mathrm{C}(A \upharpoonright g(n)) \geq h(n)$.

Proof. It is immediate that (i) implies (ii) and that (iii) implies (i). For a proof of the remaining implication from (ii) to (iii), fix h and g that satisfy (ii), and let h_A be any A-computable order. Let $m_0 = 0$ and for all $n > 0$ let

$$m_n = \min\{m : m_{n-1} < m \text{ and } 3h_A(n) \leq h(m)\} \quad \text{and} \quad I_n = [m_n, m_{n+1}).$$

For all n, let $\widetilde{g}(n)$ be an appropriate representation of the pair of the restriction of g to I_n and of the string $A \upharpoonright \max_{j \in I_n} g(j)$, and observe that the function $\widetilde{g}$ is A-computable. We now have that there are infinitely many j such that for the index n where $j \in I_n$, we have

$$\mathrm{C}(A \upharpoonright g(j)) \geq \mathrm{C}(j) + h(j) \geq \mathrm{C}(j) + h(m_n) \geq \mathrm{C}(j) + 3h_A(n), \tag{4.1}$$

where the first inequality is due to (ii), the second is implied by $j \in I_n$, and the third is by definition of m_n.

For each such j and n, assuming we would have access to j, we could extract $A \upharpoonright g(j)$ from $\widetilde{g}(n)$. This is because the latter is defined to contain an initial segment of A longer than $A \upharpoonright g(j)$.

This implies that, for each such j and n, it holds that $\mathrm{C}(\widetilde{g}(n)) \geq h_A(n)$. To see this, assume otherwise. Then $A \upharpoonright g(j)$ could be coded by giving a code for $\widetilde{g}(n)$ and one for j, which, together with some overhead for coding, would imply $\mathrm{C}(A \upharpoonright g(j)) \leq \mathrm{C}(j) + 2h_A(n) + \mathrm{O}(1)$, contradicting (4.1).

To finalize the proof, observe that it always holds that $\mathrm{C}(A \upharpoonright \widetilde{g}(n)) \geq \mathrm{C}(\widetilde{g}(n))$, so for infinitely many n we have $\mathrm{C}(A \upharpoonright \widetilde{g}(n)) \geq h_A(n)$. □

The following variant of Proposition 4.17 can be shown by almost literally the same proof, which we omit.

Proposition 4.18. *The following assertions are equivalent.*

(i) The set A is i.o. complex.

(ii) There is a computable order h and a computable function g such that there are infinitely many n where $\mathrm{C}(A \upharpoonright g(n)) \geq \mathrm{C}(n) + h(n)$.

(iii) For every computable order h there is a computable function g such that there are infinitely many n where $\mathrm{C}(A \upharpoonright g(n)) \geq h(n)$.

4.3 Diagonally non-computable sets

Definition 4.19. *A set A is* diagonally non-computable (DNC) *if there is a function $f \leq_T A$ such that $f(n)$ differs from $\varphi_n(n)$ whenever the latter value is defined. With an appropriate coding scheme for finite sequences of natural numbers understood, a set A is* strongly diagonally non-computable (SDNC) *if there is a function $f \leq_T A$ such that when z is a code for the sequence $e_1, x_1, \ldots, e_m, x_m$, then $f(z)$ differs for $i = 1, \ldots, m$ from $\varphi_{e_i}(x_i)$ whenever this value is defined.*

The notions of wtt-DNC, wtt-SDNC, tt-DNC, *and* tt-SDNC *are defined likewise, where in the above definitions $f \leq_T A$ is replaced by $f \leq_{wtt} A$ and $f \leq_{tt} A$, respectively.*

Note that if we can compute a function f such that for given n the value $f(n)$ differs from $\varphi_n(n)$, we can also compute a function g such that for given e, x the value $g(e,x)$ differs from $\varphi_e(x)$, because by the s-m-n Theorem one can effectively find an index i such that $\varphi_e(x)$ and $\varphi_i(i)$ are either both undefined or both defined and have the same value. By a result of Jokusch [Joc89], indeed even the notions of DNC and SDNC coincide.

Theorem 4.20. *A set A is DNC if and only if A is SDNC.*

Proof. By definition, it suffices to show that DNC implies SDNC. If A is DNC, one obtains an A-computable function f as required as follows. Given a natural number m we fix uniformly effective and uniformly effectively invertible bijections between $\mathbb{N}$ and $\mathbb{N}^m$. This allows us to interpret *any* natural number as an m-tuple of natural numbers. Then given a sequence $e_1, x_1, \ldots, e_m, x_m$ with code z, let $f(z)$ be equal to the m-tuple $(y_1, \ldots, y_m)$, where y_i differs from $(\varphi_{e_i}(x_i))_i$, the i-th component of $\varphi_{e_i}(x_i)$, whenever this value is defined. This implies that for any i, $f(z)$ differs in at least one component from $\varphi_{e_i}(x_i)$, so that $f(z) \neq \varphi_{e_i}(x_i)$ for all i. Also, it is obvious that for all i, given e_i and x_i, the value y_i is A-computable using the fact that A is DNC. It follows that $f(z)$ is A-computable. □

The following infinitely often versions of the notion DNC is due to Kjos-Hanssen et al. [KHMS]. Note that there are computable functions g such that $g(e)$ differs from $\varphi_e(e)$ for infinitely many e, hence in order to get interesting infinitely often versions of the various variants of the concept of DNC, one has to require more than just to be able to compute a function that differs from the partial diagonal function at infinitely many places.

Definition 4.21. *For a function g, let $E_g = \{e : g(e) = \varphi_e(e)\}$ be the* (diagonal) equality set *of g. A set A is* i.o. DNC *if for all computable functions z there is a function $g \leq_T A$ such that there are infinitely many n where*

$$\left| E_g \cap \{0, \ldots, z(n) - 1\} \right| \leq n.$$

The concepts of i.o. tt-DNC *and* i.o. wtt-DNC *are defined likewise, where in the definition* $g \leq_{\mathrm{T}} A$ *is replaced by* $g \leq_{\mathrm{tt}} A$ *and* $g \leq_{\mathrm{wtt}} A$*, respectively.*

By definition, a set A is DNC if and only if there is an A-computable function g such that E_g is empty, and consequently any set that is DNC is also i.o. DNC. More precisely, if a set A is DNC, then it satisfies the definition of i.o. DNC by a function $g \leq_{\mathrm{T}} A$ that does not depend on z.

4.4 Equivalences of the almost everywhere notions

The following theorem is due to Kjos-Hanssen, Merkle and Stephan [KHMS, Theorems 2.3 and 2.7]. The proof of their result given here is somewhat more direct, furthermore, their short but slightly technical proof of the implication from DNC to autocomplex is replaced by a simplified argument due to Khodyrev and Shen [She10], who rediscovered the known equivalence of DNC and SDNC and observed that SDNC easily implies autocomplex. The subsequent equivalence results are formulated in terms of avoidance as introduced in Definition 4.13 in order to render these results more intuitive.

Theorem 4.22. *The following assertions are equivalent.*

(i) The set A is autocomplex.

(ii) The set A is DNC.

(iii) The set A avoids c.e. traces.

Proof. To see that (i) implies (ii), assume that A is autocomplex. Then there is an A-computable function g such that for all n, we have $\mathrm{C}(g(n)) \geq n$. So $g(n)$ differs from $\varphi_n(n)$ for almost all n, because the latter value, if defined, has plain complexity of $\log n$ up to an additive constant, and consequently, A is DNC.

That (i) implies (iii) is shown by a similar argument: The set A can not be c.e. i.o. traceable, i.e., A must avoid c.e. traces, because otherwise by Remark 4.5 the function g would have an n-bounded c.e. trace, which would imply $\mathrm{C}(g(n)) \leq^{+} 2\log n$ for infinitely many n.

Next, to see that (ii) implies (i), assume that A is DNC and hence SDNC. Then A is autocomplex because in order to obtain for given n a value $g(n)$ where $\mathrm{C}(g(n)) \geq n$, it suffices to obtain a value that differs from all the values $\varphi_e(p)$ where the latter value is defined, p has length at most n, and e is an index for the universal machine used in the definition of the plain complexity C.

Finally, to prove that (iii) implies (ii), assume that the set A avoids c.e. traces, i.e., is not c.e. i.o. traceable. In order to see that A is DNC, let the diagonal trace $(T_n)_n$ be defined by $T(n) = \{\varphi_e(e)\}$ if $\varphi_e(e)$ converges and by $T(n) = \emptyset$ otherwise. By

assumption, there is an A-computable function g that is not i.o. traced by the diagonal trace, hence $g(e)$ differs from $\varphi_e(e)$, whenever the latter value is defined. □

Corollary 4.23. *A set A is c.e. i.o. traceable if and only if every A-computable function has a 1-bounded c.e. i.o. trace.*

Proof. It suffices to show the implication from left to right. By the proof of the implication from (iii) to (ii) in Theorem 4.22, if there is an A-computable function that has no 1-bounded c.e. i.o. trace, then this function witnesses that A is DNC, hence, by the same theorem, A is not i.o. c.e. traceable. □

The following variant of Theorem 4.22 is once more due to Kjos-Hanssen et al. [KHMS]. The proofs of Theorem 4.24 and its corollary are omitted because they are almost literally the same as for Theorem 4.22 and Corollary 4.23 when using the characterizations of the notion complex from Proposition 4.15 and showing separately the equivalences for truth-table and weak truth-table reducibility.

Theorem 4.24. *The following assertions are equivalent.*

(i) The set A is complex.

(ii) The set A is tt-DNC.

(iii) The set A tt-avoids c.e. traces.

The three assertions remain equivalent if one replaces in the two last assertions truth-table reducibility by weak truth-table reducibility.

Corollary 4.25. *The following assertions are equivalent.*

(i) A is not complex.

(ii) The set A is c.e. i.o. tt-traceable.

(iii) Every function $f \leq_{\mathrm{tt}} A$ has a 1-bounded c.e. i.o. trace.

(iv) The set A is c.e. i.o. wtt-traceable.

(v) Every function $f \leq_{\mathrm{wtt}} A$ has a 1-bounded c.e. i.o. trace.

4.5 Equivalence of the infinitely often notions

In Section 4.4 we have seen equivalences between first, notions of complexity and autocomplexity, second, computing diagonally non-computable functions, and third, notions of avoiding c.e. traces. The corresponding proofs were rather direct and functions g as required in the definitions of these three notions where obtained place by place in the sense that, for example, a function value $g(n)$ that has a certain complexity is obtained by considering a value $g(n)$ that is not contained in a component T_n of an appropriate trace and vice versa. Accordingly, by identical or similar arguments, we obtain infinitely often versions of these equivalence results where now, for example, for all n such that the value $g(n)$ has high complexity the value $g(n)$ avoids a corresponding set T_n and vice versa.

The two following theorems are infinitely often versions of Theorems 4.22 and 4.24. The equivalence of assertions (i) and (iii) in Theorem 4.27 for the case of weak truth-table reducibility is due to Franklin et al. [FGSW].

Theorem 4.26. *The following assertions are equivalent.*

(i) The set A is i.o. autocomplex.

(ii) The set A is i.o. DNC.

(iii) The set A i.o. avoids c.e. traces.

Proof. We first show that (i) and (iii) are equivalent, which follows by essentially the same arguments as the equivalence of being autocomplex and being DNC stated in Theorem 4.22. If A is i.o. autocomplex, then there is an A-computable function g such that for infinitely many n it holds that $\mathrm{C}(g(n)) \geq n$, and such a function g cannot have a c.e. trace that, e.g., is n-bounded, hence A is not c.e. traceable, i.e., A i.o. avoids c.e. traces. Conversely, if A i.o. avoids c.e. traces, there is an A-computable function g that has no 2^n-bounded c.e. trace, hence in particular, there are infinitely many n such that there is no word w of length strictly less than n such that $g(n) = \mathbb{U}(w)$, and consequently A is i.o. autocomplex.

In order to show that (i) implies (ii), assume that A is i.o. autocomplex. Fix any computable function z and let $m_0, m_1, \ldots$ be a strictly increasing computable sequence of natural numbers such that for all i, we have $z(m_i) < m_{i+1}$. This way the natural numbers are partitioned into consecutive intervals $I_i = [m_i, m_{i+1})$. By Proposition 4.17, choose some A-computable function g_0 such that there are infinitely many n such that $\mathrm{C}(g_0(n)) \geq \max I_n$. For all n and all j in I_n, let $g(j) = g_0(n)$. Then g is A-computable and there are infinitely many n where for all j in I_n we have

$$\mathrm{C}(\varphi_j(j)) \leq^+ \log j < j \leq \max I_n \leq \mathrm{C}(g_0(n)) = \mathrm{C}(g(j)),$$

i.e., the set E_g has empty intersection with I_n and thus contains at most $m_n = \min I_n$ numbers that are less than or equal to $z(m_n) \leq \max I_n$.

In order to demonstrate that (ii) implies (iii), we show the contrapositive, so assume that A does not i.o. avoid c.e. traces, i.e., that A is c.e. traceable. Fix some appropriate effective way of coding finite sequences of natural numbers of arbitrary length by single natural numbers. Let $(T^0_\ell)_{\ell\in\mathbb{N}}, (T^1_\ell)_{\ell\in\mathbb{N}}, \ldots$ be an appropriate effective enumeration of all c.e. traces. Let s be a computable function such that for all i and j the partial computable function $\varphi_{s(i,j)}$ on input y is computed by enumerating the numbers $c_0, c_1, \ldots$ in T^i_y until c_j is reached, where c_j is then considered as a code for a finite sequence of the form $g(0)g(1)\ldots g(\ell)$ and in case $y \leq \ell$ the output is $g(y)$.

Next define a computable function z where for all n the value $z(n)$ is chosen so large that for all $i < n$ and $j < n$ there are at least $n+1$ mutually distinct indices $e \leq z(n)$ such that the partial function φ_e is the same as $\varphi_{s(i,j)}$. Then given any function $g \leq_{\mathrm{T}} A$, let $\widetilde{g}(n)$ be a code for the finite sequence $g(0), \ldots g(z(n))$. By assumption on A, for $h\colon n \mapsto n$ there is an index i such that the c.e. trace $(T^i_\ell)_{\ell\in\mathbb{N}}$ is h-bounded and traces the function $\widetilde{g}$. For given n, let j be minimum such that $\widetilde{g}(n) = c_j$, where $c_0, c_1, \ldots$ are the numbers that are enumerated into T^i_n. Then for all sufficiently large n, there are at least $n+1$ places $e \leq z(n)$ such that

$$\varphi_e(e) = \varphi_{s(i,j)}(e) = g(e),$$

and since g was an arbitrary A-computable function and z does not depend on g, the set A is not i.o. DNC. □

Theorem 4.27. *The following assertions are equivalent.*

(i) The set A is i.o. complex.

(ii) The set A is i.o. tt-DNC.

(iii) The set A i.o. tt-avoids c.e. traces.

The three assertions remain equivalent if one replaces in the two last assertions truth-table reducibility by weak truth-table reducibility.

To prove the next theorem about high i.o. DNC sets, we first need the following proposition.

Proposition 4.28. *A set A is not i.o. autocomplex iff for all $f \leq_{\mathrm{T}} A$ and n,*

$$\mathrm{C}(f(n)) \leq^+ n.$$

Proof. Assume that A is not i.o. autocomplex, let $f \leq_T A$, and let g be the A-computable bound for the use function of the reduction Φ of f to A. We can w.l.o.g. assume that $g(n)$ also encodes n; if it does not replace it by $\langle g(n), n\rangle$, where $\langle .,. \rangle$ is an appropriate computable pairing function. Using the reduction Φ we see that $C(f(n)) \leq^+ C(A \upharpoonright g(n))$. By the definition of being not i.o. autocomplex we have $C(f(n)) \leq^+ n$, then.

For the converse direction, let f be an A-computable order, and let $g(n)$ be a code for $A \upharpoonright f(n)$. Then $g \leq_T A$; by assumption $C(g(n)) \leq^+ n$, and so A is not i.o. autocomplex. □

Theorem 4.29. *For a set A that is i.o. DNC and high, there is a single function $g \leq_T A$ such that for all computable orders z there exist infinitely many numbers n such that*

$$\Big|E_g \cap \{0, \ldots, z(n)-1\}\Big| \leq n.$$

Proof. Since A is high, there is an A-computable h such that h dominates all computable functions z. Let m be the A-computable function defined for all i by $m(i+1) = h(m(i))$ and set $I_i = [m(i), m(i+1))$ for all i. Then m dominates all computable functions, including z as in the statement of the theorem.

In case A is i.o. DNC, by Theorem 4.26, it is also i.o. autocomplex. By Proposition 4.17 this implies that there is an A-computable function k such that there are infinitely many n such that $C(A \upharpoonright k(n)) \geq^+ h(m(n))$. Call the set of these n's P and let f be the function $n \mapsto A \upharpoonright k(n)$. Define g by setting $g(k) := f(n)$ for all $k \in I_n$, that is, g is constant on each interval. It is obvious that g is A-computable.

For the infinitely many $n \in P$ we now have the following situation: On the one hand, for all $k \in I_n$, $C(g(k)) \geq^+ h(m(n))$. On the other hand, for $\ell \leq h(m(n))$ we have $C(\varphi_\ell(\ell)) \leq^+ \log \ell \leq \log h(m(n))$.

This implies that for all sufficiently large $n \in P$ and all $k \in I_n$, $g(k) \neq \varphi_\ell(\ell)$ for any $\ell \leq h(m(n))$, so I_n does not intersect E_g. Thus, it holds that for all sufficiently large $n \in P$,

$$\Big|E_g \cap \{0, \ldots, z(m(n))-1\}\Big| \leq \Big|E_g \cap \{0, \ldots, h(m(n))-1\}\Big| \leq m(n),$$

where $m(n)$ is equal to $\sum_{i=1}^{n-1} |I_i|$. □

4.6 Computable traces and total machines

We have seen above that traceability notions defined in terms of c.e. traces can be characterized by concepts such as autocomplexity that relate to the plain Kolmogorov complexity of the initial segments of a set. We will see now that these characterizations can be extended to traceability notions defined in terms of computable traces if one considers the complexity of initial segments with respect to total machines.

Remark 4.30. *Bienvenu and Merkle [BM07] have defined the notion of decidable machines, that is, machines whose domain is decidable. Obviously, every total machine is decidable, and every decidable machine can be easily converted into a total machine by first deciding whether a string is in the domain and then executing the machine as normal if that is the case, and outputting a constant otherwise.*

Definition 4.31. *A set A is* totally complex *iff there is a computable function g such that for all total machines M and almost all n, we have $\mathrm{C}_M(A \upharpoonright g(n)) \geq n$. A set A is* totally i.o. complex *iff there is a computable function g such that for all total machines M there are infinitely many n where $\mathrm{C}_M(A \upharpoonright g(n)) \geq n$.*

Theorem 4.32 can be obtained from a result of Kjos-Hanssen et al. [KHMS, Theorem 5.1] and Theorem 4.8. We omit the proof of Theorem 4.32 and give instead the very similar proof of its infinitely often version Theorem 4.33. In connection with the latter theorem, note that Franklin and Stephan [FS10] considered computably tt-traceable sets, that is, sets that do not i.o. tt-avoid computable traces, and showed that these sets are exactly the Schnorr-trivial sets.

Theorem 4.32. *A set A is totally complex if and only if A tt-avoids computable traces.*

Theorem 4.33. *A set A is totally i.o. complex if and only if A i.o. tt-avoids computable traces.*

Proof. First assume that A is not totally i.o. complex, i.e., for any computable function g there exists a total machine M such that for almost all n, we have $\mathrm{C}_M(A \upharpoonright g(n)) \leq n$. Fix any function $f \leq_{\mathrm{tt}} A$ and some tt-reduction witnessing this fact, which has use bound $u(n)$. By assumption on A, there is a total machine M such that for almost all n, we have $\mathrm{C}_M(A \upharpoonright u(n)) \leq n$. In order to obtain a computable trace $(T_n)_n$ for f that is bounded by the function $n \mapsto 2^{n+1}$, execute all codes of length up to n on M, view the outputs as initial segments of oracles, and let T_n contain all values that one obtains by simulating the fixed tt-reduction for computing f at place n with any of these oracles. Then $f(n)$ is contained in T_n for almost all n. Since the bound 2^{n+1} on the size of the sets T_n does not depend on f, the set A is computably tt-traceable.

Next assume that A does not i.o. tt-avoid computable traces, i.e., that A is computably tt-traceable, and recall that by the discussion preceding Remark 4.5 we can assume that any function wtt-reducible to A has a computable trace that is n-bounded. Given a computable function g, we need to show that there is a total machine M such that for almost all n, we have $\mathrm{C}_M(A \upharpoonright g(n)) \leq n$. We can assume that the function $n \mapsto A \upharpoonright g(n)$ has a computable trace $(T_n)_n$ where T_n has size at most n. Let M be the machine, which on input (n, i) outputs the i-th element of T_n, if this element exists, and outputs some constant otherwise. Since the set T_n has size

at most n and its canonical index can be computed from n, M is total and satisfies $\mathrm{C}_M(A \upharpoonright g(n)) \leq 2\log n \leq n$ for almost all n. □

4.7 Lower bounds on initial segments complexity

When introducing the notions of i.o. complex and i.o. autocomplex, we have argued that it does not make sense to define these notions by requiring for the set A under consideration that for a computable or A-computable order, respectively, infinitely often the order provides a lower bound for the plain Kolmogorov complexity of an initial segment of A, and the reason for this was simply that by choosing a small enough order this condition would be trivially satisfied by all sets. We will argue in this section, however, that equivalent definitions in terms of lower bounds for the complexity of initial segments can be given, if plain Kolmogorov complexity C is replaced by appropriate variants, e.g., by uniform or monotonic complexity (see Li and Vitányi [LV08] for a more detailed account of these notions). We will restrict our attention to the concept of i.o. autocomplex.

Definition 4.34. *The* length-conditioned complexity $\mathrm{C}(w|n)$ *of w is the length of the shortest program p such that $\mathbb{U}$ on input $(p, |w|)$ will output w.*

The uniform complexity $\mathrm{C}(w;n)$ *of w is the length of the shortest program p such that for all $i \leq |w|$, $\mathbb{U}$ on input (p,i) will output the first i bits of w, while $\mathbb{U}$ may do anything on inputs (p,i) with $i > |w|$.*

The monotonic complexity $\mathrm{C}_{\mathrm{mon}}(w)$ *is the length of the shortest program p such that $\mathbb{U}$ on input p will output some extension of w.*

From these definitions, the following chain of inequalities is immediate.

$$\mathrm{C}(w|n) \leq^+ \mathrm{C}(w;n) \leq^+ \mathrm{C}_{\mathrm{mon}}(w) \leq^+ \mathrm{C}(w) \tag{4.2}$$

Definition 4.35. *A set A is* length-conditionedly i.o. autocomplex *iff there is an A-computable order h such that for infinitely many n, we have $h(n) \leq \mathrm{C}(A \upharpoonright n|n)$.*

A set A is uniformly i.o. autocomplex *iff there is an A-computable order h such that for infinitely many n, we have $h(n) \leq \mathrm{C}(A \upharpoonright n;n)$.*

A set A is monotonically i.o. autocomplex *iff there is an A-computable order h such that for infinitely many n, we have $h(n) \leq \mathrm{C}_{\mathrm{mon}}(A \upharpoonright n)$.*

In connection with the following theorem, recall that the first, and hence also the second and third assertion are equivalent to A not being c.e. traceable.

Theorem 4.36. *The following assertions are equivalent.*

(i) The set A is i.o. autocomplex.

(ii) The set A is monotonically i.o. autocomplex.

(iii) The set A is uniformly i.o. autocomplex.

These three equivalent assertions are all implied by

(iv) The set A is length-conditionedly i.o. autocomplex.

Proof. By the chain of inequalities (4.2), it is immediate that (iv) implies (iii) and that (iii) implies (ii).

In order to see that (ii) implies (i), it suffices to show that c.e. traceable implies not monotonically i.o. autocomplex. So let an A-computable order h be given, where we can w.l.o.g. assume that $\mathrm{range}(h) = \mathbb{N}$. We want to show that for (almost) all n it holds that $\mathrm{C}_{\mathrm{mon}}(A \upharpoonright n) \leq h(n)$.

Let h^{-1} be the discrete inverse of h and look at the A-computable function $f : m \mapsto A \upharpoonright h^{-1}(m+1)$. Then this function has a c.e. trace $(T_m)_m$ with bound m.

This implies that for all m we have

$$\mathrm{C}_{\mathrm{mon}}(A \upharpoonright h^{-1}(m+1)) \leq^+ \log m + \log m \leq^+ m,$$

by coding the index m of T_m and a description inside T_m. In other words we have that for n in the range of h^{-1} we have $\mathrm{C}_{\mathrm{mon}}(A \upharpoonright n) \leq h(n-1)$.

It now remains to look at n not in the range of h^{-1}. For these, define

$$n^{>} = \min\{\ell \in \mathrm{range}(h^{-1}) \mid \ell > n\}.$$

Then, due to the properties of monotone complexity, we have

$$\mathrm{C}_{\mathrm{mon}}(A \upharpoonright n) \leq \mathrm{C}_{\mathrm{mon}}(A \upharpoonright n^{>}) \leq h(n^{>} - 1) = h(n),$$

so the bound holds everywhere.

In order to see that (i) implies (iii), it suffices to show that any set that is not uniformly i.o. autocomplex is c.e. traceable. Given such a set A and an A-computable function f, it suffices to construct a c.e. trace for f that is bounded by the fixed function $h(n) = 2^n$.

Let u be the use function of some oracle Turing machine computing f from A, where we can assume that u is strictly increasing. Let g be a sufficiently slowly growing A-computable order such that $g(u(n)) < n = \log h(n)$. Such an order exists because u is A-computable. Since A is not i.o. uniformly autocomplex, for this g and for almost all n we have that $\mathrm{C}(A \upharpoonright n; n) \leq g(n)$, hence

$$\mathrm{C}(A \upharpoonright u(n); u(n)) \leq g(u(n)) < \log h(n)$$

holds for almost all n. Consequently, some program of length strictly less than $\log h(n)$, say p, is the correct uniform description of an initial segment of A that is long enough to compute $f(n)$.

One problem that we still need to address is that, when executing p in order to compute the first i bits of A, we need to provide to the universal machine not only p but also i. The work-around is to take advantage of the uniform nature of the complexity notion present here. We start the computation of $f(n)$ and as soon as we query the i-th bit of the oracle, we execute p with additional input i in order to obtain the first i bits of A. Since the program p is correct and uniform, we will indeed get the initial segment of A of length i. Should we later query the oracle again at a position $j > i$, we execute p again with additional input j etc. A set T_n of size at most $h(n)$ that contains $f(n)$ can then be enumerated by processing as just described and in parallel all programs of length strictly less than $\log h(n)$. □

Together with Theorem 4.26 we get the following easy corollary.

Corollary 4.37. *No c.e. traceable set can be length-conditionedly i.o. autocomplex.*

Similarly, one can introduce notions such as uniformly or monotonically i.o. complex and can derive the equivalence with the notion i.o. complex.

4.8 Tiny use and autocomplexity

Franklin et al. [FGSW] characterized being not i.o. complex in terms of so-called reductions with tiny use, as can be seen in the following definitions and theorem. To keep consistency with their notation, in this section we identify binary words with natural numbers as described in the introduction — so Turing machines operate on natural numbers etc. The Kolmogorov complexity will still be defined as the length of a shortest binary description.

Definition 4.38 (Franklin et al. [FGSW])**.** *For sets A and B say that A is* reducible to B with tiny use *($A \leq_{\mathrm{T(tu)}} B$), iff for every computable order h, $A \leq_{\mathrm{T}} B$ with use function bounded by h.*

Definition 4.39 (Franklin et al. [FGSW])**.** *For sets A and B say that A is* uniformly reducible to B with tiny use *($A \leq_{\mathrm{uT(tu)}} B$), iff there is a single Turing reduction Φ where $\Phi^B = A$ whose use function is dominated by every computable order.*

Definition 4.40 (Franklin et al. [FGSW])**.** *For a set A we define the function g_A to map k to the smallest number n such that for all $m \geq n$ we have $\mathrm{C}(A \upharpoonright m) > k$. In other words, g_A is the point from which on we always need more than k bits to describe an initial segment of A.*

For a string x let x^* denotes the smallest number with the property $\mathbb{U}(x^*) = x$. Then define $A^* := \{(A \upharpoonright g_A(k))^* \mid k \in \mathbb{N}\}$ as the set that contains for all k a shortest code for the longest initial segment of A that is describable with k bits.

Theorem 4.41 (Franklin et al. [FGSW])**.** *The following are equivalent for a set A.*

(i) *There exists a set B with $A \leq_{\mathrm{T(tu)}} B$.*

(ii) *A is not i.o. complex.*

(iii) *g_A dominates every computable function.*

(iv) $A \leq_{\mathrm{uT(tu)}} A^*$.

We now transform this result into a result characterizing being not i.o. autocomplex using a notion of tiny use relative to an oracle, while adapting the proof of Franklin et al. in a straight-forward way.

Definition 4.42. *For sets A, B and C say that A is* reducible to B with C-tiny use *($A \leq_{\mathrm{T}(C\text{-tu})} B$), iff for every C-computable order h, $A \leq_{\mathrm{T}} B$ with use function bounded by h.*

Definition 4.43. *For sets A, B and C say that A is* uniformly reducible to B with C-tiny use *($A \leq_{\mathrm{uT}(C\text{-tu})} B$), iff there is a single Turing reduction Φ where $\Phi^B = A$ whose use function is dominated by every C-computable order.*

Definition 4.44. *A function f is called A-*dominant *iff it dominates every A-computable function.*

For a reduction Φ that reduces A to B, denote by $\Phi(B \upharpoonright n)$ the longest initial segment of A such that the initial segment $B \upharpoonright n$ of B is long enough to answer all oracle queries that occur during the computation of the reduction.

The following proposition is straight-forward to prove.

Proposition 4.45. *(i)* *$A \leq_{\mathrm{T}(C\text{-tu})} B$ if and only if for every C-computable order g, there is a Turing reduction $\Phi^B = A$ such that the map $n \mapsto |\Phi(B \upharpoonright n)|$ dominates g.*

(ii) *$A \leq_{\mathrm{uT}(C\text{-tu})} B$ if and only if there is a Turing reduction $\Phi^B = A$ such that the map $n \mapsto |\Phi(B \upharpoonright n)|$ dominates every C-computable order g.*

It was shown in [FGSW] that g_A and A^* are $A \oplus \emptyset'$-computable.

Theorem 4.46. *The following are equivalent for any set A.*

(i) *There exists a set B with $A \leq_{\mathrm{T}(A\text{-tu})} B$.*

(ii) *A is not i.o. autocomplex.*

(iii) *g_A is A-dominant.*

(iv) $A \leq_{\mathrm{uT}(A\text{-tu})} A^*$.

Proof. That (iv) implies (i) is immediate.

(i) implies (ii): Let's assume $A \leq_{\mathrm{T}(A\text{-tu})} B$ and suppose that $f \leq_{\mathrm{T}} A$. Then there exists a reduction Γ with $\Gamma^A = f$ whose use is bounded by an A-computable function g. From Proposition 4.45 it follows that there is a reduction Φ with $\Phi^B = A$ such that $n \mapsto |\Phi(B \upharpoonright n)|$ dominates g. So $\Phi(B \upharpoonright n)$ is a long enough segment of the oracle B for the computation of $f(n)$. So $\mathrm{C}(f(n)) \leq^+ \mathrm{C}(\Phi(B \upharpoonright n), n) \leq^+ \mathrm{C}(B \upharpoonright n) \leq^+ n$. Then it follows from Proposition 4.28 that A cannot be i.o. autocomplex.

(ii) implies (iii): Assume that A is not i.o. autocomplex and let f be an increasing A-computable function. By definition, for almost all n, $\mathrm{C}(A \upharpoonright f(n)) \leq n$. By definition of g_A, for almost all n, $g_A(n) > f(n)$.

(iii) implies (iv): For all sets A we have $A \leq_{\mathrm{T}} A^*$, because in order to decide $x \in A$ we just have to look for a word in A^* describing a long enough initial segment of A. Such a word will always be found eventually.

Let Φ be the described reduction. We now show that under the assumption that g_A is A-dominant this reduction already witnesses that $A \leq_{\mathrm{uT}(A\text{-tu})} A^*$. To prove that statement, we need to show that $n \mapsto |\Phi(A^* \upharpoonright n)|$ dominates every increasing A-computable function f. W.l.o.g. assume that f is non-decreasing. Fix c to be largest number with the property that there are infinitely many k such that $g_A(k) = g_A(k+1) = \ldots = g_A(k+c-1)$.

Let g denote the computable function $k \mapsto 2^{k+1}$, which dominates the map $k \mapsto (A \upharpoonright g_A(k))^*$. Since g_A is A-dominant, for almost all k, $f(g(k+c)) < g_A(k)$ holds. Suppose that k is large enough and that it satisfies

$$(A \upharpoonright g_A(k))^* < n \leq (A \upharpoonright g_A(k+c))^*. \tag{4.3}$$

This implies $n \leq g(k+c)$ and so $f(n) < g_A(k)$. Because of the first inequality in (4.3), $|\Phi(A^* \upharpoonright n)| \geq g_A(k)$ and so $|\Phi(A^* \upharpoonright n)| \geq f(n)$ as required. □

4.9 Time bounded traceability and complexity

In this section we will make a short digression into the realm of time bounded Kolmogorov complexity while staying in the traceability context. We will return to the topic of time bounded Kolmogorov complexity in Part II. Here, we will show that for appropriately chosen notions of complexity and traceability, the relations between these two notions can be transferred to the time-bounded setting, more precisely, to a setting of polynomial time bounds. Recall that for $t : \mathbb{N} \to \mathbb{N}$, time bounded Kolmogorov complexity is defined by

$$\mathrm{C}^t(x) := \min\{|\sigma| : \mathbb{U}(\sigma) = x \text{ in at most } t(|x|) \text{ steps}\}.$$

Consider a coding of finite sets of natural numbers where the code of a set D consists of the concatenation of the binary expansion of elements of D in the

natural order, where all the bits in the binary expansions are doubled and the binary expansions are separated from each other by the word 01. In the sequel, we will identify a finite set D with its code. Instead of looking at the Kolmogorov complexity of initial segments we will examine the Kolmogorov complexity of strings $A \upharpoonright D$ where D is a finite subset of $\mathbb{N}$.

Definition 4.47. *A set A is* i.o. poly-complex *iff there is a computable order h such that for all polynomials p there are infinitely many sets D where we have for $t = p(|D| + |\max D|)$ that $\mathrm{C}^t(A \upharpoonright D \mid D) \geq h(\max D)$.*

Definition 4.48. *A set A is* polynomial-time tt-traceable *iff for all computable orders h, we have that for every function $f \leq^{\mathsf{P}}_{\mathrm{tt}} A$ there is an h-bounded trace $(T_n)_n$ such that for given n, the list of elements of T_n can be computed (or, say, printed) in time polynomial in the length of n.*

Theorem 4.49. *The following statements are equivalent.*

(i) A is not i.o. poly-complex.

(ii) A is polynomial-time tt-traceable.

Proof. (i) implies (ii): Let h be the desired trace bound, where we can assume that $h(n) \leq n$ and that h can be computed in polynomial time by switching to a delayed version of h, and let $f \leq^{\mathsf{P}}_{\mathrm{tt}} A$ be the function to be traced. Let q be the polynomial time bound of some fixed tt-reduction from f to A, and let $D(n)$ be the query set of this reduction at place n, where we can assume that $D(n)$ always contains n.

Now the mapping $g : n \mapsto \lfloor \log h(n) \rfloor$ is surely a computable order, so by assumption for some p and almost all n we have for $t = p(|D(n)| + |\max D(n)|)$ that $\mathrm{C}^t(A \upharpoonright D(n) \mid D(n)) < g(n)$. Since t and $g(n)$ are both polynomial in the length of n, polynomial time in the length of n suffices to run the universal machine on all programs p of length strictly less than $g(n)$ with conditioning $D(n)$ for at most t steps each, interpreting the outputs obtained this way as oracles and to simulate the given reduction at place n with all of these oracles in order to obtain at most $2^{g(n)} - 1 \leq h(n)$ values that are put into the set T_n.

(ii) implies (i): We have to show for a given computable order h that there is a polynomial p such that for almost all finite sets D it holds for $t = p(|D| + |\max D|)$ that $\mathrm{C}^t(A \upharpoonright D \mid D) < h(\max D)$. Let f be the function which, for all n that are a code for some finite set D, maps n to $A \upharpoonright D$, where $f(n) = 0$ in case n is not such a code. By definition of the coding, computing $f(n)$ from A requires at most $\log n$ queries to A of length at most $\log n$. So $f \leq^{\mathsf{P}}_{\mathrm{tt}} A$, say with polynomial time bound q.

Since the length of the code for a finite set D is effectively bounded in $\max D$, we can fix a computable order h' such that for any finite set D with code n, we

have $h'(n) \leq h(\max D)$. By assumption on A, let $(T_n)_n$ be an h'-bounded trace for f with polynomial time bound, i.e., for any finite set D with code n the value $f(n) = A \upharpoonright D$ occurs among the at most $h'(n) \leq h(\max D)$ elements of T_n and $C^t(A \upharpoonright D \mid D) \leq h(\max D)$ with t polynomial in $|n|$. With

$$|n| \leq |D| \cdot |\max D| \leq (|D| + |\max D|)^2$$

it follows that t is polynomial in $|D| + |\max D|$, as desired. □

Part II

Kolmogorov complexity with time bounds

CHAPTER 5

Distinction Complexity

In general, the Kolmogorov complexity of a word w is the length $|d|$ of a shortest program d such that d determines w effectively. In a setting of unbounded computations, this approach leads canonically to the usual notion of plain Kolmogorov complexity and its prefix-free variant. In a setting of resource-bounded computations though, there are several notions of Kolmogorov complexity that are in some sense natural — and none of them is considered canonical.

A straight-forward approach is to cap the execution time and/or used space by simply not allowing descriptions that take too long or too much space for producing the word we want to describe. This notion has the disadvantage that for a fixed resource-bound there is no canonical notion of universal machine.

Another approach, which has received considerable attention in the literature, was introduced by Levin [Lev84], where, in contrast to the notion just mentioned, arbitrarily long computations are allowed, but a large running time increases the complexity value. More precisely, with some appropriate universal machine U understood, in Levin's model the Kolmogorov complexity of a word d is the minimum of $|d| + \log t$ over all pairs of a word d and a natural number t such that U takes time t to check that w is the word determined by d. As for other notions of resource-bounded Kolmogorov complexity, here one can differentiate between generation complexity and distinction complexity [AKRR03, Sip83], where the former asks for a program d such that w can actually be computed from d, whereas the latter asks for a program d that distinguishes w from other words in the sense that given d and any word u, one can effectively check whether u is equal to w.

The question of how generation and distinction complexity relate to each other in the setting of Levin's notion of resource-bounded Kolmogorov complexity has been investigated by Allender et al. [AKRR03]. A relevant notion in this context is that of the degree of ambiguity of a language in a non-deterministic complexity class, where we consider a language more ambiguous if for the machines recognizing it there are inputs on which the machines have a larger number of accepting paths.

More explicitly, Allender et al. consider a notion of solvability for non-deterministic computations that — for a given resource-bounded model of computation — amounts to require that for any non-deterministic machine N there is a deterministic machine that exhibits the same acceptance behavior as N on all inputs for which the number of accepting paths of N is not too large, e.g., is at most logarithmic in the number of all possible paths. They demonstrate that for any word the generation complexity is at most polynomial in the distinction complexity if and only if all computations in exponential time, whose ambiguity is limited, can be done deterministically in exponential time. We extend their work to two similar equivalences. First, generation complexity is at most linear in distinction complexity if and only if all unambiguous computations in linearly exponential time can be done deterministically in linearly exponential time. Second, the conditional generation complexity of a word w given a word y is at most linear in the conditional distinction complexity of w given y if and only if all unambiguous computations in polynomial time can be done deterministically in polynomial time. This implies that if the mentioned relation between conditional generation complexity and conditional distinction complexity holds P is equal to UP, where the latter is the class of problems $L \in \mathsf{NP}$ for which there is a machine that accepts the problem with exactly one accepting path for every input in L. Both equivalences remain valid if one replaces unambiguity with limited ambiguity in an appropriate sense to be defined. Combining this result with a result by Fortnow and Kummer [FK96] about the promise problem (1SAT, SAT), one obtains that conditional generation and distinction complexity for the classical definitions of time bounded Kolmogorov complexity are close if and only if they are close in Levin's model just described.

Finally, we prove unconditionally that in the setting of space-bounded complexity — that is for complexity measures Ks and KDs that logarithmically count the used space instead of the running time of a description — generation complexity is at most linear in distinction complexity.

The notion of generation complexity considered below differs from Levin's original notion insofar as one has to generate only single bits of the word to be generated but not the word as a whole. This variant has already been used by Allender et al. [AKRR03]; their results mentioned above, as well as the results demonstrated below extend to Levin's original model by almost identical proofs.

For a complexity class C, we will refer by C-machine to any machine M that uses a model of computation and obeys a time- or space-bound such that M witnesses $\mathrm{L}(M) \in \mathrm{C}$ with respect to the standard definition of C. For example, an NE-machine is a non-deterministic machine that runs in linearly exponential time.

In this chapter we don't use the standard universal Turing machine $\mathbb{V}$, but fix a special universal machine U that receives as input encoded tuples of words. For example, (x, y, z) will be encoded by $\widetilde{x}01\widetilde{y}01\widetilde{z}$ where the word $\widetilde{u}$ is obtained by doubling every symbol in u, i.e., $\widetilde{u} = u(0)u(0)u(1)u(1)\ldots u(|u|-1)u(|u|-1)$.

5.1 Known results

Definition 5.1 (Levin [Lev84], Allender et al. [AKRR03])**.** Time-bounded generation complexity Kt *and* distinction complexity KDt *are defined by*

$$\mathrm{Kt}(x) = \min\left\{ |d| + \log t \;\middle|\; \begin{array}{c} \forall b \in \{0,1,*\}\colon \forall i \le |x|\colon U(d,i,b) \\ \text{runs for } t \text{ steps and accepts} \\ \text{iff the } i\text{-th bit of } (x*) \text{ is } b. \end{array} \right\},$$

$$\mathrm{KDt}(x) = \min\left\{ |d| + \log t \;\middle|\; \begin{array}{c} \forall y \in \{0,1\}^{|x|}\colon U(d,y) \text{ runs for} \\ t \text{ steps and accepts iff } x = y \end{array} \right\}.$$

Observe that in the definition of Kt-complexity the symbol $*$ has to be generated as an end marker for the word x.

Remark 5.2. *The notion of* Kt*-complexity introduced in Definition 5.1 was proposed by Allender et al. in [AKRR03] as a variation of Levin's original definition, where the latter requires to generate whole words instead of individual bits. Levin's original definition has the advantage of assuring that for all x, it holds that*

$$\mathrm{KDt}(x) \le \mathrm{Kt}(x) + \log|x|.$$

In connection with Theorem 5.16 we also use the following conditional complexity notions.

Definition 5.3. *The* conditional time-bounded distinction complexity Kt *and* conditional generation complexity KDt *are defined by*

$$\mathrm{Kt}(x|y) = \min\left\{ |d| + \log t \;\middle|\; \begin{array}{c} \forall b \in \{0,1,*\}\colon \forall i \le |x|\colon U(d,i,b,y) \\ \text{runs for } t \text{ steps and accepts} \\ \text{iff the } i\text{-th bit of } (x*) \text{ is } b. \end{array} \right\},$$

$$\mathrm{KDt}(x|y) = \min\left\{ |d| + \log t \;\middle|\; \begin{array}{c} \forall z \in \{0,1\}^{|x|}\colon U(d,z,y) \text{ runs for} \\ t \text{ steps and accepts iff } z = x \end{array} \right\}.$$

We will shortly review Theorem 17 from Allender et al. [AKRR03] before we will state our extensions.

Definition 5.4. *We say that a machine M recognizes a set L* with polynomial advice *iff there is a polynomial p and an* advice function $a\colon \mathbb{N} \to \{0,1\}^{<\infty}$ *such that $a(n) \le p(n)$ and that M recognizes the set*

$$\{(x, a(|x|)) \mid x \in L\}.$$

Note that $a(|x|)$ provides M with information that is "helpful" in determining the answer to the question whether x is in L, but that this helpful information may only depend on $|x|$, not on the content of x — otherwise it could simply be the value of $L(x)$ and the notion would become trivial. Recognition with access to linear advice *is defined similarly, where the length of the advice is bounded by a linear function.*

Definition 5.5 (Allender et al. [AKRR03])**.** *We say that* FewEXP search instances are EXP-solvable *if, for every* NEXP*-machine N and every k, there is an* EXP*-machine M with the property that if N has fewer than $2^{|x|^k}$ accepting paths on input x, then M produces on input x some accepting path as output if there is one.*

We say that FewEXP decision instances are EXP-solvable *if, for every* NEXP*-machine N and every k, there is an* EXP*-machine M with the property that if N has fewer than $2^{|x|^k}$ accepting paths on input x, then M accepts x if and only if N accepts x.*

We say that FewEXP decision instances are EXP/poly-solvable *if, for every* NEXP*-machine N and every k, there is an* EXP*-machine M having access to advice of polynomial length, such that if N has fewer than $2^{|x|^k}$ accepting paths on input x, then M accepts x if and only if N accepts x.*

The notion of solvability can be equivalently characterized in terms of promise problems [CHV93, FK96]. This will be discussed further in connection with Theorem 5.19 by Fortnow and Kummer.

Remark 5.6. *Note that by definition* EXP *solvability of* FewEXP *decision instances implies* FewEXP $=$ EXP*, where* FewEXP *is the the class of problems $L \in$ NEXP recognized by a non-deterministic Turing machine that has at most $2^{|x|^k}$ many accepting paths on every input.*

It is unknown whether the reverse implication holds as well. This is because the definition of EXP *solvability of* FewEXP *decision instances does not require the considered machines to have a limited number of accepting paths on* all *inputs.*

Theorem 5.7 (Allender et al. [AKRR03])**.** *The following statements are equivalent:*

(i) *For all x, $\mathrm{Kt}(x) \in (\mathrm{KDt}(x))^{\mathrm{O}(1)}$.*

(ii) FewEXP *search instances are* EXP*-solvable.*

(iii) FewEXP *decision instances are* EXP*-solvable.*

(iii') FewEXP *decision instances are* EXP/poly*-solvable.*

(iv) *For all $A \in$ P and for all $y \in A^{=l}$ it holds that*

$$\mathrm{Kt}(y) \in (\log |A^{=l}| + \log l)^{\mathrm{O}(1)}.$$

In words this means, that if generating words does not require "much larger" descriptions than distinguishing them from other words, then witnesses for certain non-deterministic computations with few witnesses can be found deterministically, and vice versa.

5.2 Tools

In what follows we will use a corollary of the following two results by Buhrman et al., which have also been used in [AKRR03]. They allow us to build distinguishing descriptions for numbers from division residues of those numbers.

Lemma 5.8 (Buhrman et al. [BFL01]). *Let $n \in \mathbb{N}$ be large enough and let*

$$A := \{x_1, x_2, \ldots, x_{|A|}\} \subseteq \{l, l+1, \ldots, l+n-1\},$$

where the $x_1, \ldots, x_{|A|}$ are pairwise different. Then for all sufficiently large n and all $x_i \in A$ and at least half of the prime numbers $p \leq 4 \cdot |A| \cdot \log^2 n$ it holds for all $j \neq i$ that $x_i \not\equiv x_j \pmod{p}$.

Proof. Assume that there are $\log n$ primes p_k in the specified range, such that for some pair of indices i and j with $i \neq j$ (w.l.o.g. with $x_i < x_j$) we have for all $k \leq \log n$ that $x_i \equiv x_j \pmod{p_k}$.

Then according to the uniqueness property (modulo $\prod p_k$) in the Chinese Remainder Theorem we would have

$$x_j \geq x_i + \prod_{k=1}^{\log n} p_k \geq x_i + 2^{\log n} \geq x_i + n$$

This implies $x_j \notin S$, a contradiction. Therefore, for all pairs i and j with $i \neq j$ there are less than $\log n$ prime numbers with the specified property.

It follows that for every x_i there are less than $|A| \cdot \log n$ primes, such that for any other x_j there occurs equality of the residue. The prime number theorem implies that (asymptotically) there are

$$\frac{4 \cdot |A| \cdot \log^2 n}{\ln(4 \cdot |A| \cdot \log^2 n)} = \frac{2 \cdot |A| \cdot \log^2 n}{\ln(2\sqrt{|A|} \log n)} = 2 \cdot |A| \cdot \log n \cdot \frac{\log n}{\ln(2\sqrt{|A|} \log n)}$$

prime numbers in the specified range, which is asymptotically larger than $2 \cdot |A| \cdot \log n$.

Therefore, for all sufficiently large n, at least half of the prime numbers in the specified range are fit to provide a x_i-residue that can serve as a unique description for x_i. □

Lemma 5.9 (Buhrman et al. [BFL01]). *Let* $A \subseteq \{s_l, s_{l+1}, \ldots, s_{l+n-1}\} \subseteq \{0,1\}^{\leq k}$. *Then for all* x *in* A,

$$\mathrm{KDt}^A(x) \leq 2\log|A| + \mathrm{O}(\log\log n) + \mathrm{O}(\log k)$$

where $\mathrm{KDt}^A(x)$ *denotes the* KDt*-complexity of* x *relative to oracle* A.

In particular, if $A \in \mathsf{P}$, *then for all* x *in* A,

$$\mathrm{KDt}(x) \leq 2\log|A| + \mathrm{O}(\log\log n) + \mathrm{O}(\log k).$$

Proof. A KDt-program for x using oracle A and hard-coded p_x and $x \bmod p_x$ from Lemma 5.8 might look like this:

1. input y;
2. if $(A(y) = 0)$ reject

   else if $(y \bmod p_x = x \bmod p_x)$ accept

   else reject.

The length of this program is

$$\begin{aligned} |p_x| + |x \bmod p_x| + \mathrm{O}(1) &= 2\log(4 \cdot |A| \cdot \log^2 n) + \mathrm{O}(1) \\ &= 2\log|A| + 4\log(2\log n) + \mathrm{O}(1) \\ &= 2\log|A| + \mathrm{O}(\log\log n) \end{aligned}$$

using the fact that $|x \bmod p_x| \leq |p_x|$.

The running time for words y of length $|x|$ (and those are the ones we have to consider according to the definition of KDt) is essentially determined by the polynomial running time for the modulo operation. Since running time counts logarithmically this results in $\mathrm{O}(\log k)$.

If $A \in \mathsf{P}$, then we can check whether $y \in A$ directly instead of using an oracle. Since running time counts logarithmically, the polynomial running time for this operation does not introduce any further terms into the upper bound on $\mathrm{KDt}(x)$. □

We will use a special case of this statement for our purpose.

Corollary 5.10. *Let* $A \subseteq \{0,1\}^{<\infty}$, $y \in \{0,1\}^{<\infty}$ *and* $l \in \mathbb{N}$. *Let*

$$A_{y,l} := A \cap \{x \mid y \sqsubseteq x \wedge |x| = l\}.$$

Then it holds that for all x *in* $A_{y,l}$,

$$\mathrm{KDt}^{A_{y,l}}(x) \leq 2\log|A_{y,l}| + \mathrm{O}(\log l)$$

In particular, if there is a machine that on input y, l and x decides in polynomial time whether x is in the set $A_{y,l}$, then for all x in $A_{y,l}$,

$$\mathrm{KDt}(x|y) \leq 2\log|A_{y,l}| + \mathrm{O}(\log l).$$

Proof. With the notation of Lemma 5.9 we have $n \leq 2^l$ and $k = l$. The claim follows using the lemma. □

5.3 The linearly exponential case

We will now state our variants of Theorem 5.7, which are proven in a similar way as the original statements by Allender et al.

Definition 5.11. *We say that* FewE search instances are E-solvable *if, for every* NE*-machine N and every k, there is an* E*-machine M with the property that if N has fewer than $2^{k\cdot|x|}$ accepting paths on input x, then M produces on input x some accepting path as output if there is one.*

We say that FewE decision instances are E-solvable *if, for every* NE*-machine N and every k, there is an* E*-machine M with the property that if N has fewer than $2^{k\cdot|x|}$ accepting paths on input x, then M accepts x if and only if N accepts x.*

We say that FewE decision instances are E/lin-solvable *if, for every* NE*-machine N and every k, there is an* E*-machine M having access to advice of linear length, such that if N has fewer than $2^{k\cdot|x|}$ accepting paths on input x, then M accepts x if and only if N accepts x.*

We say that UE decision instances are E-solvable *if, for every* NE*-machine N and every k, there is an* E*-machine M with the property that if N has at most one accepting path on input x, then M accepts x if and only if N accepts x.*

Theorem 5.12. *The following statements are equivalent:*

(i) *For all words x, $\mathrm{Kt}(x) \in \mathrm{O}(\mathrm{KDt}(x))$.*

(ii) FewE *search instances are* E*-solvable.*

(iii) FewE *decision instances are* E*-solvable.*

(iii') UE *decision instances are* E*-solvable.*

(iii") FewE *decision instances are* E/lin*-solvable.*

(iv) *For all $A \in \mathsf{P}$, for all words y and for all $l \in \mathbb{N}$ it holds that for*

$$A_{y,l} := A \cap \{x \mid y \sqsubseteq x \wedge |x| = l\}$$

and for all $x \in A_{y,l}$,

$$\mathrm{Kt}(x) \in \mathrm{O}(\log|A_{y,l}| + \log l + |y|).$$

Proof. (i) implies (iv): Given y and l, we can decide membership of x in $A_{y,l}$ in time polynomial in $|x|$. To do this, we first check whether $y \sqsubseteq x$ and whether x has the correct length l. If yes, calculate $A(x) = A_{y,l}(x)$ using the fact that $A \in \mathsf{P}$.

Therefore we can apply Corollary 5.10 to see that

$$\mathrm{KDt}(x) \leq 2\log|A_{y,l}| + \mathrm{O}(\log l) + \mathrm{O}(|y|)$$

where the last term accounts for supplying y as part of the program. Using assumption 1 the claim follows.

(iv) implies (ii): Fix a non-deterministic Turing machine N running in linearly exponential time 2^{kn}, where we can assume that N branches binarily. Let

$$D := \{yx \mid x \in \{0,1\}^{2^{k\cdot|y|}} \text{codes an accepting computation of } N \text{ on } y\}.$$

Obviously, $D \in \mathsf{P}$.

Now fix any y such that M on input y has at most $2^{k\cdot|y|}$ accepting paths. Then the set $D_y := D \cap \{yx \mid |x| = 2^{k\cdot|y|}\}$ contains at most $2^{k\cdot|y|}$ words and by assumption (iv) it follows that for all x in D_y,

$$\mathrm{Kt}(yx) \in \mathrm{O}(\log|D_y| + \log(|y| + 2^{k\cdot|y|}) + |y|) = \mathrm{O}(|y|)$$

So, in order to find an accepting path of M on input y, if there is one, it suffices to search through all words yx with $\mathrm{Kt}(yx) \leq \mathrm{O}(|y|)$. This can be done in linearly exponential time, as required.

The implications from (ii) to (iii), from (iii) to (iii'), and from (iii) to (iii") are trivial.

(iii") implies (i): Let N be an NE-machine which on input $(d, 1^t, i, b, n)$ guesses a word $x \in \{0,1\}^n$, simulates 2^t steps of the computation of $U(d,x)$ and then accepts iff $U(d,x)$ accepts and $x(i) = b$.

If d is a *distinguishing* description for a word $x \in \{0,1\}^n$, then for all sufficiently large t there is exactly one accepting path of N on input $(d, 1^t, i, x(i), |x|)$ and none for $(d, 1^t, i, \overline{x(i)}, |x|)$ for any i. By assumption (iii"), there is an E-machine M, which computes $N((d, 1^t, i, x(i), |x|))$ deterministically for such d and t given some advice h of linear length.

The specification of t and $|x|$ (both encoded in binary), M, d, and h therefore constitutes a Kt-program for x. Since d was a KDt-program, we can assume that $|d| \leq \mathrm{KDt}(x)$. Since 2^t was the time bound for this program, we can assume that $t \leq \mathrm{KDt}(x)$. Since the Kt-program's running time depends linearly exponentially on t, and running time is counted logarithmically, the Kt-program's running time increases $\mathrm{Kt}(x)$ by a value that is linear in $\mathrm{KDt}(x)$. $\mathrm{KDt}(x)$ is always greater than $\log|x|$, because otherwise x could not even be read completely, and therefore x

could not be correctly distinguished. Also, by definition of linear advice, we have $|h| \in O(|d| + |1^t| + \log|x|)$.

All this, together with the fact that running time counts logarithmically, results in

$$\mathrm{Kt}(x) \leq O(\mathrm{KDt}(x)).$$

(iii') implies (i): Identical, except that we don't consider FewE decision instances but UE decision instances (but the construction still works) and that we don't have to worry about h. □

Remark 5.13. *Theorem 5.12 remains valid by essentially the same proof when formulated for Levin's original notion of* Kt *instead of the variant of Allender et al. In the proofs that (iii') or (iii") imply (i) we would have had to generate all bits of x instead of just one bit. This would increase the running time by factor $|x|$, which, due to the logarithmic counting of running time, would result only in an additional additive term* $\log|x| \leq \mathrm{KDt}(x)$ *for the* Kt*-complexity.*

Let UE denote the class of problems $L \in$ NE recognized by a non-deterministic Turing machine that has at most one accepting path on any input.

Corollary 5.14. *If for all x,* $\mathrm{Kt}(x) \in O(\mathrm{KDt}(x))$*, then* UE = E.

Proof. According to the theorem, the assumption implies that UE decision instances are E-solvable. Since a language in UE contains only such instances, the claim follows. □

5.4 The polynomial case

Of course, a more interesting result than the one given in the last section would be a similar result for the polynomial time case. That is, an equivalence between the statement that distinction complexity and generation complexity are close to each other and the statement that non-deterministic polynomial time recognizing of a set with few accepting paths can be done deterministically. Such a result would concern the relation between the complexity classes P and UP, and knowing more about this relation might be interesting in connection with the important P versus NP question.

The following result is an analogon of the equivalence stated in Theorem 5.12 for polynomial time. To achieve this result we need to consider conditional complexities.

Definition 5.15. *We say that* FewP search instances are P-solvable *if, for every* NP *machine N and every k there is a* P *machine M with the property that if N has fewer than $|x|^k$ accepting paths on input x, then M produces on input x some accepting path as output if there is one.*

We say that FewP decision instances are P-solvable *if, for every* NP *machine N and every k there is a* P *machine M with the property that if N has fewer than $|x|^k$ accepting paths on input x, then M accepts x if and only if N accepts x.*

We say that UP decision instances are P-solvable *if, for every* NP*-machine N and every k, there is a* P*-machine M with the property that if N has at most one accepting path on input x, then M accepts x if and only if N accepts x.*

Theorem 5.16. *The following statements are equivalent:*

(i) *For all words x and y, $\mathrm{Kt}(x|y) \in \mathrm{O}(\mathrm{KDt}(x|y))$.*

(ii) FewP *search instances are* P*-solvable.*

(iii) FewP *decision instances are* P*-solvable.*

(iii') UP *decision instances are* P*-solvable.*

(iv) *For all $A \in \mathsf{P}$ it holds that for $A_{y,l} := A \cap \{x \mid y \sqsubseteq x \wedge |x| = l\}$ and for all $x \in A_{y,l}$*

$$\mathrm{Kt}(x|y) \in \mathrm{O}(\log |A_{y,l}| + \log l).$$

Proof. (i) implies (iv): We have access to y through the conditioning. If we also have access to l, we can decide membership of x in $A_{y,l}$ in polynomial time. To do this, we first check whether $y \sqsubseteq x$ and whether y has the correct length l. If yes, compute the value $A(x) = A_{y,l}(x)$ using the fact that $A \in \mathsf{P}$. Corollary 5.10 then yields

$$\mathrm{KDt}(x|y) \leq 2 \log |A_{y,l}| + \mathrm{O}(\log l).$$

Using assumption 1 the claim follows.

(iv) implies (ii): Let N be any non-deterministic machine running in polynomial time n^k, where we can assume that N branches binarily. Let L denote $\mathrm{L}(N)$. Let

$$D := \{yx \mid x \in \{0,1\}^{|y|^k} \text{ codes an accepting computation of } N \text{ on } y\}.$$

Obviously, $D \in \mathsf{P}$. Now fix any y such that M on input y has at most $|y|^k$ accepting paths. Then the set $D_y := D \cap \{yx \mid |x| = |y|^k\}$ contains at most $|y|^k$ words and by assumption (iv) it follows that for all yx in D_y,

$$\begin{aligned} \mathrm{Kt}(yx|y) &\in \mathrm{O}(\log |D_y| + \log(|y| + |y|^k)) \\ &= \mathrm{O}(\log |y|) \end{aligned}$$

So, in order to find an accepting path of M on input y, if there is one, it suffices to search through all words x with $\mathrm{Kt}(x|y) \leq \mathrm{O}(\log |y|)$. This can be done in polynomial time, so that $L \in \mathsf{P}$ as was to be shown. Note that it causes no problems

that we have to deal with conditional complexity here. This is because when we are searching for an accepting path x for a word y we obviously have access to y.

(ii) implies (iii): This is trivial.

(iii) implies (iii'): This is trivial.

(iii') implies (i): Let us assume that we have a shortest KDt-description d, finite conditioning information y and a described word x such that the universal machine U accepts the triple (d, x, y) in t_{KDt} steps.

Consider a variant U_{UP} of the universal machine U, where we assume that U_{UP} can simulate steps of U without time overhead, that is, every step of U can also be simulated by U_{UP} in a single step. The machine U_{UP} will be given inputs of the form $(\hat{d}, 1^{\hat{t}}, \hat{i}, \hat{b}, \hat{n}, \hat{y})$. On any such input, if $\hat{n} > \hat{t}$, then reject. Otherwise guess a word $\hat{x} \in \{0,1\}^{\hat{n}}$ and check whether $\hat{x}(\hat{i}) = \hat{b}$. If yes, U_{UP} behaves for $\hat{t}$ steps like U on input $(\hat{d}, \hat{x}, \hat{y})$, that is U_{UP} accepts iff $U(\hat{d}, \hat{x}, \hat{y})$ accepts in these $\hat{t}$ steps.

Since d is a *distinguishing* description for the word $x \in \{0,1\}^n$, for all i on input $(d, 1^{t_{\mathrm{KDt}}}, i, \underline{x(i)}, |x|, y)$ there is a unique accepting path of U_{UP} and none on input $(d, 1^{t_{\mathrm{KDt}}}, i, \overline{x(i)}, |x|, y)$. By assumption (iii') there is a deterministic machine M that for all such inputs has the same acceptance and rejection behaviour as N and works in some fixed polynomial time bound.

The input for M together with an encoding of M is a generating program for x. It only remains to prove that this program is small enough and computes fast enough, compared to the KDt-program.

1. The program consists of t_{KDt}, $|x|$ (both encoded in binary), M, d. Since t_{KDt} counted logarithmically for KDt we have $\log t_{\mathrm{KDt}} \leq \mathrm{KDt}(x|y)$. $\mathrm{KDt}(x|y)$ is always greater than $\log|x|$, because to be able to distinguish x from other words we need at least the time to read x completely. One fixed M works for all appropriate inputs and its encoding therefore has constant length. Obviously, $d \leq \mathrm{KDt}(x|y)$. Furthermore, y is given as part of the input to M, but does not add to $\mathrm{Kt}(x|y)$. In total, all the components together have a length bounded by $\mathrm{O}(\mathrm{KDt}(x|y))$.

2. By the above construction the running time t_{UP} of the non-deterministic machine U_{UP} on input $T := (d, 1^{t_{\mathrm{KDt}}}, i, x(i), |x|, y)$ will be $|T| + |x| + t_{\mathrm{KDt}}$.

   It holds that $|T| = \Theta(t_{\mathrm{KDt}})$:

   - Since we can in t_{KDt} steps only access the first t_{KDt} bits of y we can w.l.o.g. assume $|y| < t_{\mathrm{KDt}}$.
   - For the same reason as above, the execution of a distinguishing description for x on U takes at least $|x|$ steps, so a binary encoding of $|x|$ takes less than t_{KDt} bits.

– By a similar argument we can assume $|d| < t_{\mathrm{KDt}}$.

So the length of T must always be between t_{KDt} and $6 \cdot t_{\mathrm{KDt}}$. As before, $|x| \leq t_{\mathrm{KDt}}$. In summary, $t_{\mathsf{UP}} = |T| + |x| + t_{\mathrm{KDt}} \in \Theta(t_{\mathrm{KDt}}) = \Theta(|T|)$.

If we now replace the non-deterministic machine U_{UP} with the deterministic machine M, a polynomial overhead might be introduced. Therefore we have

$$t_M \in (\mathrm{O}(1) \cdot |T|)^{\mathrm{O}(1)} = (\mathrm{O}(1) \cdot t_{\mathrm{KDt}})^{\mathrm{O}(1)} = t_{\mathrm{KDt}}^{\mathrm{O}(1)},$$

hence $\log t_M \in \mathrm{O}(\log t_{\mathrm{KDt}})$.

All this, together with the fact that running time counts logarithmically, results in the required inequality $\mathrm{Kt}(x|y) \leq \mathrm{O}(\mathrm{KDt}(x|y))$. □

Remark 5.17. *For the same reason as in Remark 5.13, this proof would also work if we considered Levin's original definition of* Kt.

Corollary 5.18. *If for all x and y, $\mathrm{Kt}(x|y) \in \mathrm{O}(\mathrm{KDt}(x|y))$, then $\mathsf{UP} = \mathsf{P}$.*

Proof. According to the theorem, the assumption implies that UP decision instances are P-solvable. Since a language in UP contains only such instances, the claim follows. □

Fortnow and Kummer [FK96, Theorem 24] proved an equivalence related to Theorem 5.16 in the setting of the "traditional" polynomially time-bounded Kolmogorov complexities C^t and CD^t [LV08, Chapter 7] where for example

$$\mathrm{C}^t(x|y) := \min\{|\sigma| : \mathbb{V}((\sigma, y)) \downarrow = x \text{ in at most } t(|x|) \text{ steps}\}.$$

Theorem 5.19 (Fortnow, Kummer)**.** *The following two statements are equivalent:*

(i) UP *decision instances are* P*-solvable.*

(ii) For any polynomial t there are a polynomial t' and a constant $c \in \mathbb{N}$ such that for all x and y it holds that $\mathrm{C}^{t'}(x|y) \leq \mathrm{CD}^t(x|y) + c$.

Remark 5.20. *Even, Selman and Yacobi [ESY84] defined promise problems. The idea is that a computational problem can be solved if a certain promise is held. In our case, the promise is that the non-deterministic computation has at most few (or one) accepting path. $\mathcal{UP}$ is the class of promise problems (Q, R) such that there exists a non-deterministic polynomial time Turing machine N such that N accepts all words in R and such that N has at most one accepting path on the words in Q (this is the promise).*

Fortnow and Kummer formulated their above equivalence in terms of promise problems. Instead of the first statement in the theorem they used the assertion that the promise problem $(1\mathrm{SAT}, \mathrm{SAT})$ *is in* P*. Here* 1SAT *contains exactly those instances*

of SAT *that are satisfiable by exactly one assignment; and for the promise problem* (1SAT, SAT) *to be in* P *means that there is a* P*-machine that accepts all* $x \in 1\mathrm{SAT} \cap \mathrm{SAT}$ *and rejects all* $x \in \{0,1\}^{<\infty} - \mathrm{SAT}$.

Their formulation of the first statement is indeed equivalent to the one used above, because (1SAT, SAT) *is complete for* $\mathcal{UP}$*, as witnessed by a parsimonious version of Cook's Theorem due to Simon [Sim75, Theorem 4.1]. Here parsimonious means that the number of witnesses is preserved during the polynomial time Turing reduction, or, in other words, the number of accepting paths of the original instance is the same as the number of satisfying assignments for the Boolean formula to which we reduce.*

The following corollary is immediate from Theorems 5.16 and 5.19.

Corollary 5.21. *The following two statements are equivalent.*

(i) *For all* x *and* y, $\mathrm{Kt}(x|y) \in \mathrm{O}(\mathrm{KDt}(x|y))$.

(ii) *For any polynomial* t *there is a polynomial* t' *and a constant* $c \in \mathbb{N}$ *such that for all* x *and* y *it holds that* $\mathrm{C}^{t'}(x|y) \leq \mathrm{CD}^{t}(x|y) + c$.

5.5 Space bounds

In analogy to the time-bounded case one can define the following two notions of space-bounded Kolmogorov complexity.

Definition 5.22. *The* space-bounded distinction complexity Ks *and* generation complexity KDs *are defined by*

$$\mathrm{Ks}(x) = \min\left\{ |d| + \log s \;\middle|\; \begin{array}{c} \forall b \in \{0,1,*\}\colon \forall i \leq |x|\colon U(d,i,b) \\ \text{runs in space } s \text{ and accepts} \\ \text{iff the } i\text{-th bit of } (x*) \text{ is } b \end{array} \right\},$$

$$\mathrm{KDs}(x) = \min\left\{ |d| + \log\max(s,|x|) \;\middle|\; \begin{array}{c} \forall y \in \{0,1\}^{|x|}\colon U(d,y) \text{ runs in} \\ \text{space } s \text{ and accepts iff } x = y \end{array} \right\}.$$

Here U *is a machine with a two-way read-only input tape, where only the space on the work tapes is counted.*

Remark 5.23. *For the definition of* KDs *it is relevant how the candidate* y *is provided to* U *and if the space for* y *is counted. Here we chose to do count the space for* y *which accounts for the term* $\max|x|$ *in the definition of* KDs*. This then implies the inequality* $\log|x| \leq \mathrm{KDs}(x)$*, which is analogous to the corresponding statement for* KDt *and will be used in the proof of Theorem 5.24.*

Theorem 5.24. *For almost all* x*, it holds that* $\mathrm{Ks}(x) \leq 5 \cdot \mathrm{KDs}(x)$.

Proof. Let N be a non-deterministic machine which on input (d, s, i, b, n) guesses a word $y \in \{0,1\}^n$, simulates the computation of $U(d, y)$ while limiting the used space to s, and then accepts iff $y_i = b$ and $U(d, y)$ accepts. In particular, if d is a *distinguishing* description for a word $x \in \{0,1\}^n$, then for all sufficiently large s and for all $i \leq n$ there is an accepting path of N on input $(d, s, i, x(i), |x|)$ but none on $(d, s, i, \bar{x}(i), |x|)$.

By the Theorem of Savitch there is a deterministic machine M that has the same acceptance behavior as N and uses space at most s^2; observe in this connection that s is specified in the input of N and M, hence doesn't have to be computed by M.

Given a word x, fix a pair d and s such that d is a distinguishing program for x, it holds that $|d| + \log s \leq \mathrm{KDs}(x)$, and U uses space at most s on input (d, x). The specification of d, s, $|x|$ and M therefore constitutes a Ks-program for x which runs in space s^2. By choice of d and s we have

$$|d| + \log s + \log |x| \leq 2\mathrm{KDs}(x).$$

Furthermore, the space s^2 used in the computation of M counts only logarithmically, where $2 \cdot \log s \leq 2 \cdot \mathrm{KDs}(x)$. Taking into account that M has to be specified and that some additional information is needed to separate the components of the Ks-program for x, we obtain $\mathrm{Ks}(x) \leq 5 \cdot \mathrm{KDs}(x)$ for all sufficiently large x. □

Remark 5.25. *The exact multiplicative constant in the theorem also depends on how the used space is counted. Here we count all used tape cells. If, e.g., we would instead supply to the Turing machine every one of its arguments on its own tape and would only count the maximum number of used tape cells on any tape, a smaller constant than* 5 *would result.*

For the same reason as in Remark 5.13, the proof of Theorem 5.24 would also work if — in analogy to Levin's original definition of Kt *— we would demand generating the whole word instead of just one bit, even when counting the space used on the output tape. Due to the additional additive term* $\log |x|$ *we would get the slightly weaker conclusion* $\mathrm{Ks}(x) \leq 6 \cdot \mathrm{KDs}(x)$.

CHAPTER 6

Kolmogorov complexity and computational depth

The original notion of depth was introduced by Bennett [Ben95], offering the following argument to explain the necessity of such a notion: Imagine we make a measurement in a scientific experiment and wonder about the cause for the specific pattern we observe. There are some patterns, such as "111…", that point to a systematic cause, but to one that is rather simple. There are other patterns, like a random sequence, that make it plausible to assume that the measurement is just the result of background noise.

But there are also objects that display a large degree of organization or structure, e.g., a life-form or a literary work. Bennett argues that this large degree of organization is caused by a "nontrivial causal history" that was involved in creating the object. We want to be able to characterize formally such objects in the real world; that is, objects whose internal complexity evidences a complex generation process. Random sequences do not evidence this kind of causal history, nor do easy sequences like "111…".

In this chapter we first review the definitions and then state the most important known theorems about depth. We will then proof a dichotomy about time bounded Kolmogorov complexity that is closely related to the notion of depth. To do this we will consider the time-bounded and unbounded Kolmogorov complexity of the initial segments of sets that are computably enumerable.

The initial segments of a c.e. set A have small Kolmogorov complexity, more precisely, by Barzdins' lemma it holds that $C(A \upharpoonright m) \leq^+ 2\log m$. Kummer's celebrated gap theorem [DH10, Kum96] states that any array non-computable c.e. Turing degree contains a c.e. set B such that there are infinitely many m such that $C(B \upharpoonright m) \geq 2\log m$, whereas all c.e. sets in an array computable Turing degree satisfy $C(A \upharpoonright m) \leq (1+\varepsilon)\log m$ for all $\varepsilon > 0$ and almost all m.

Theorem 6.13, our main result in this chapter, has a structure similar to Kummer's gap theorem in so far as it asserts a dichotomy in the complexity of initial segments between high and non-high c.e. sets. More precisely, every high c.e. Turing degree contains a c.e. set B such that for any computable function t there is a constant $c_t > 0$ such that for all m it holds that $C^t(B \upharpoonright m) \geq c_t \cdot m$, whereas for any non-high c.e. set A there is a computable time bound t and a constant c such that for infinitely many m it holds that $C^t(A \upharpoonright m) \leq \log m + c$. By similar methods it can be shown that any high degree contains a set B such that $C^t(B \upharpoonright m) \geq^+ m/4$ for all computable t. The constructed sets B have low unbounded but high time-bounded Kolmogorov complexity, and accordingly we obtain an alternative proof of the result due to Juedes, Lathrop, and Lutz [JLL94] that every high degree contains a strongly deep set.

6.1 Introduction

We first give Bennett's original definition of depth.

Definition 6.1 (Bennett [Ben95])**.** *Let x and w be strings and s a significance parameter. A string's* depth at significance level s*, denoted by $D_s(x)$, will be defined as* $\min\{T(p): (|p| - |p^*| < s) \wedge (\mathbb{U}(p) = x)\}$*, where p^* is the shortest possible prefix-free program for p and $T(p)$ is the running time of $\mathbb{U}$ on input p. At any given level s, a string is called* t-deep *if its depth exceeds t, and* t-shallow *otherwise.*

To define strong depth for infinite sequences, Bennett's original definition requires that for all t and all computable functions f, all but finitely many initial segments of the sequence have t-depth exceeding $f(n)$. As an example that illustrates the difference between incompressibility and depth look at the sets Ω, the halting probability for a random program, and H, the halting problem. While they are Turing-reducible to each other, the information in H is arranged in a compression-wise exponentially less efficient way than in Ω. On the other hand, the information in H is accessible in linear time, while there is no computable time bound that allows to extract the same information from Ω.

An important result about depth is the Slow Growth Law, that expresses that high depth can only be generated by the investment of correspondingly large running times.

Theorem 6.2 (Slow Growth Law [Ben95])**.** *Given any data string x and two significance parameters $s_2 > s_1$, a random program generated by coin tossing has probability less than $2^{-(s_2 - s_1) + O(1)}$ of transforming x into an excessively deep output, i.e. one whose s_2-significant depth exceeds the s_1-significant depth of x plus the run time of the transforming program plus* $O(1)$*. More precisely, there exist positive constants c_1, c_2 such that for all strings x, and all pairs of significance parameters $s_2 > s_1$, the prefix set $\{q: D_{s_2}(\mathbb{U}(q,x)) > D_{s_1}(x) + T(q,x) + c_1\}$ has measure less than $2^{-(s_2 - s_1) + c_2}$.*

Bennett also suggests another approach for modelling depth, namely through "usefulness" of a sequence for other computations, but conjectures that this is too anthropocentric a concept to formalize. In fact though, later authors have shown that his definition of depth models usefulness quite well (see below).

Definition 6.3 (Bennett [Ben95]; Juedes, Lathrop, Lutz [JLL94])**.** *For functions $t, g : \mathbb{N} \to \mathbb{N}$, $n \in \mathbb{N}$ and $x \in \{0,1\}^{<\infty}$ let*

$$\mathsf{PROG}^t(x) := \{p \mid \mathbb{U}(p) = x \text{ in time at most } t(|x|)\},$$

$$\mathsf{D}^t_g(n) := \{\alpha \in \{0,1\}^\infty \mid \forall p \in \mathsf{PROG}^t(\alpha \upharpoonright n) \colon \mathrm{K}(p) \leq |p| - g(n)\},$$

$$\mathsf{D}^t_g := \{\alpha \in \{0,1\}^\infty \mid \text{For nearly all } n \colon \alpha \in \mathsf{D}^t_g(n)\},$$

A sequence $\alpha \in \{0,1\}^\infty$ is called strongly deep *if for every computable time bound t and every constant $c \in \mathbb{N}$ we have $\alpha \in \mathsf{D}^t_c$.*

It may be surprising in the definitions of D^t_g and strong depth we are comparing $|p|$ and $\mathrm{K}(p)$ instead of $\mathrm{K}(\alpha \upharpoonright n)$ with $\mathrm{K}^t(\alpha \upharpoonright n)$. This is resolved by the following lemma.

Definition 6.4 (Bennett [Ben95]; Juedes, Lathrop, Lutz [JLL94])**.** *Define the following sets.*

$$\widehat{\mathsf{D}}^t_g(n) := \{\alpha \in \{0,1\}^\infty \mid \mathrm{K}(\alpha \upharpoonright n) \leq \mathrm{K}^t(\alpha \upharpoonright n) - g(n)\},$$

$$\widehat{\mathsf{D}}^t_g := \{\alpha \in \{0,1\}^\infty \mid \text{For almost all } n \colon \alpha \in \widehat{\mathsf{D}}^t_g(n)\}.$$

Lemma 6.5 (Bennett [Ben95], Juedes, Lathrop, Lutz [JLL94])**.** *For a computable time bound t there are constants c, d and a computable time bound t' such that for all g and n,*

- $\mathsf{D}^t_{g+c}(n) \subseteq \widehat{\mathsf{D}}^t_g(n)$,
- $\mathsf{D}^t_{g+c} \subseteq \widehat{\mathsf{D}}^t_g$,
- $\widehat{\mathsf{D}}^{t'}_{g+d}(n) \subseteq \mathsf{D}^t_g(n)$,
- $\widehat{\mathsf{D}}^{t'}_{g+d} \subseteq \mathsf{D}^t_g$.

Juedes et al. proved another version of the slow growth law that shows that strong depth is inherited upward under tt-reducibility.

Theorem 6.6 (Juedes, Lathrop, Lutz [JLL94])**.** *Let $\alpha, \beta \in \{0,1\}^\infty$. Then, if $\beta \leq_{\mathrm{tt}} \alpha$ and β is strongly deep, α is also strongly deep.*

The most interesting result in [JLL94] is probably the following relation between depth and usefulness.

Definition 6.7. *We say that a class $C \subseteq \{0,1\}^\infty$ has rec-measure* 0 *iff there is a computable martingale d that succeeds on all $\alpha \in C \cap \mathsf{REC}$, that is, if it holds that* $\limsup_n d(\alpha \upharpoonright n) = \infty$.

Definition 6.8. *A sequence $\alpha \in \{0,1\}^\infty$ is* weakly useful *if there is a computable time bound $t : \mathbb{N} \to \mathbb{N}$ such that $\mathsf{DTIME}^\alpha(t)$ does not have rec-measure* 0 *in* REC, *where $\mathsf{DTIME}^\alpha(t)$ designates the sets recognizable with time bound t and oracle access to α.*

In other words a weakly useful set is useful in the sense that it allows to compute a non-negligible part of REC within a fixed time bound. The following theorem shows that for a set to be able to do that, it has to be deep.

Theorem 6.9 (Juedes, Lathrop, Lutz [JLL94])**.** *Every weakly useful sequence is strongly deep.*

This proves in a sense that Bennett's idea of depth modelling the computational usefulness of a sequence was correct.

6.2 Time bounded Kolmogorov complexity and strong depth

The main result of this chapter is a dichotomy for the time-bounded Kolmogorov complexity of the prefixes of high and non-high c.e. sets. For a start, we discuss the Kolmogorov complexity of initial segments of c.e. sets in general.

The initial segments of a c.e. set A have small Kolmogorov complexity; by Barzdins' lemma [DH10] it holds for all m that

$$\mathrm{C}(A \upharpoonright m \mid m) \leq^+ \log m \quad \text{and} \quad \mathrm{C}(A \upharpoonright m) \leq^+ 2\log m.$$

Furthermore, there are infinitely many initial segments that have considerably smaller complexity. The corresponding observation in the following remark is extremely easy, but was only noted recently [HKM09] and, in particular, improves on corresponding statements in the literature [DH10, Lemma attributed to Solovay in Chapter 14].

Remark 6.10. *Let A be a c.e. set. Then there is a constant c such that for infinitely many m it holds that*

$$\mathrm{C}(A \upharpoonright m \mid m) \leq c, \ \mathrm{C}(A \upharpoonright m) \leq^+ \mathrm{C}(m) + c, \ \textit{and}\ \mathrm{C}(A \upharpoonright m) \leq \log m + c.$$

For a proof, it suffices to fix an effective enumeration of A and to observe that there are infinitely many $m \in A$ such that m is enumerated after all numbers $n \leq m$ that

are in A, i.e., when knowing m one can simulate the enumeration until m appears, at which point one then knows $A \upharpoonright m$.

Barzdins [Bar68] states that there are c.e. sets with high time-bounded Kolmogorov complexity, and the following lemma generalizes this in so far as such sets can be found in every high Turing degree.

Lemma 6.11. *For any high set A there is a set B where $A =_T B$ such that for every computable time bound t there is a constant $c_t > 0$ where*

$$C^t(B \upharpoonright m) \geq^+ c_t \cdot m \quad \text{and} \quad C(B \upharpoonright m) \leq^+ 2 \log m.$$

Moreover, if A is c.e., B can be chosen to be c.e. as well.

Proof. Let A be any high set. We will construct a Turing-equivalent set B as required. Since A is high there is a function g computable in A that dominates any computable function f, i.e., $f(n) \leq g(n)$ for almost all n. Fix such a function g, and observe that in case A is c.e., we can assume that g can be effectively approximated from below. This is because otherwise we may replace g with the function g' defined as follows. Let M_g be an oracle Turing machine that computes g if supplied with oracle A. For all n, let

$$\widetilde{g}(n,s) := \max(\{M_g^{A_i}(n) \mid i \leq s\} \cup \{0\}),$$

where A_i is the approximation to A after i steps of enumeration, and let

$$g'(n) := \lim_{s \to \infty} \widetilde{g}(n,s).$$

We have $g(n) \leq g'(n)$ for all n and by construction, g' can be effectively approximated from below.

Partition $\mathbb{N}$ into consecutive intervals $I_0, I_1, \ldots$ where interval I_j has length 2^j and let $m_j = \max I_j$. By abuse of notation, let $t_0, t_1, \ldots$ be an effective enumeration of all partial computable functions. Observe that it is sufficient to ensure that the assertion in the theorem is true for all $t = t_i$ such that t_i is computable, non-decreasing and unbounded. Assign the (potential) time bounds to the intervals $I_0, I_1, \ldots$ such that t_0 will be assigned to every second interval including the first one, t_1 to every second interval including the first one of the *remaining* intervals, and so on for $t_2, t_3, \ldots$, and note that this way t_i will be assigned to every 2^{i+1}-th interval.

We construct a set B as required. To code A into B, for all j let $B(m_j) = A(j)$, while the remaining bits of B are specified as follows. Fix any interval I_j and assume that this interval is assigned to $t = t_i$. Let B have empty intersection with $I_j \setminus \{m_j\}$ in case the computation of $t(m_j)$ requires more than $g(m_j)$ steps. Otherwise, run all codes of length at most $|I_j| - 2$ on the universal machine $\mathbb{V}$ for $2t(m_j)$ steps each,

and let w_j be the least word of length $|I_j| - 1$ that is not output by any of these computations, hence $C^{2t}(w_j) \geq |w_j| - 1$. Let the restriction of B to the first $|w_j|$ places in I_j be equal to w_j.

Now let v_j be the initial segment of B of length $m_j + 1$, i.e., up to and including I_j. In case $t = t_i$ is computable, non-decreasing and unbounded, for almost all intervals I_j assigned to t, we have $C^t(v_j) > |v_j|/3$, because otherwise, for some appropriate constant c, the corresponding codes would yield for almost all j that $C^{2t}(w_j) \leq |v_j|/3 + c \leq |I_j| - 2$. Furthermore, by construction for every such t there is a constant $c_t > 0$ such that for almost all m, there is some interval I_j assigned to t such that $m_j \leq m$ and $c_t m \leq m_j/4$. To see this, fix a t that is computable, non-decreasing and unbounded and an I_j assigned to t. Assume m could be arbitrarily large compared to m_j. If m became *too* large, due to the regular appearance of intervals assigned to t, this would imply that m is larger than the next m_k assigned to t — so replace m_j by m_k.

Hence for almost all m the initial segment of B up to m cannot have Kolmogorov complexity of less than $c_t m$.

By construction it is clear that if A was c.e., B is c.e. as well.

To see that $B \leq_T A$, let's compute any fixed interval I_j using A: We compute $g(m_j)$ using A and try to compute the assigned time bound $t_i(m_j)$. If this computation does not halt in at most $g(m_j)$ steps, then we know that during the construction of B no diagonalization has occurred on this interval, so we output all 0's on the positions in $(m_{j-1}, m_j - 1]$. If on the other hand the computation of $t_i(m_j)$ halts within the given number of steps, we can retrace the diagonalization done in the construction of B. In both cases we let $B(m_j) = A(j)$ in addition.

Finally, to see that $C(B \upharpoonright m) \leq^+ 2 \log m$, notice that, in order to determine $B \upharpoonright m$ without time bounds, it is enough to know for all intervals I_i up to the interval that contains m whether the time bound t_i assigned to I_i terminates before its computation is canceled by g. Encoding this information requires one bit per interval, plus another one describing the bit of A coded into B at the end of each interval. □

Lemma 6.12. *Every high degree contains for every computable, non-decreasing and unbounded function h a set B such that for every computable time bound t and almost all m,*

$$C^t(B \upharpoonright m) \geq^+ \frac{1}{4} m \quad \text{and} \quad C(B \upharpoonright m) \leq h(m) \cdot \log m.$$

Proof. The argument is similar to the proof of Lemma 6.11, but now, when considering interval I_j, we diagonalize against the largest running time among

$$t_0(m_j), \ldots, t_{h(j)-2}(m_j)$$

such that the computation of this value requires not more than $g(j)$ steps. This way we ensure — for any computable time bound t — that *at the end* of almost all intervals I_j compression by a factor of at most $1/2$ is possible, and that *within* interval I_j, we have compressibility to a factor of at most $1/4$, up to a constant additive term, because $B \upharpoonright m_{j-1}$ was compressible by a factor of at most $1/2$ and $m_{j-1} = |I_j|$. □

Kummer's gap theorem [DH10, Kum96] asserts that any array non-computable c.e. Turing degree contains a c.e. set A such that there are infinitely many m such that $\mathrm{C}(A \upharpoonright m) \geq 2 \log m$, whereas all c.e. sets in an array computable Turing degree satisfy $\mathrm{C}(A \upharpoonright m) \leq (1+\varepsilon) \log m$ for all $\varepsilon > 0$ and almost all m. Similarly, Theorem 6.13, the main result of this section, asserts a dichotomy for the time-bounded complexity of initial segments between high and non-high sets.

Theorem 6.13. *Let A be any c.e. set.*

(i) If A is high, then there exists a c.e. set B with $B =_{\mathrm{T}} A$ such that for every computable time bound t there is a constant $c_t > 0$ such that for all m, it holds that $\mathrm{C}^t(B \upharpoonright m) \geq c_t \cdot m$.

(ii) If A is not high, then there is a computable time bound t such that for infinitely many n, $\mathrm{C}^t(A \upharpoonright m) \leq^+ \log m$.

Proof. The first assertion is immediate from Lemma 6.11. In order to demonstrate the second assertion, let m_A be a modulus of convergence of the set A, i.e.,

$$\mathrm{m}_A(n) = \min\{s \geq m \mid A_s \upharpoonright m = A \upharpoonright m\},$$

where A_s is the finite set of numbers that have been enumerated by a fixed enumeration of A after s steps. The modulus m_A is obviously computable in A, and since A is not high there is a computable function f such that for infinitely many m it holds that $\mathrm{m}_A(m) \leq f(m)$. That means that there are infinitely many lengths m where $A \upharpoonright m$ can be computed by enumerating A for $f(m)$ steps. For these lengths, $A \upharpoonright m$ can be coded by providing the number m (code length $\log m$) and a constant-size code for f. Because f is computable, the time needed for the computation of $f(m)$ and for simulating the enumeration of A is computable itself, hence there is a computable time bound t as required. □

Remark 6.14. *The second assertion in Theorem 6.13 does not extend in general to sets that are not c.e., since for example there are low ML-random sets. This can be seen by using the characterization of ML-random sets from Theorem 2.1, fixing one value for c to get a Π^0_1 class, and then applying the Low Basis Theorem [DH10].*

As another easy consequence of Lemma 6.11, we get an alternative proof of the result due to Juedes, Lathrop and Lutz [JLL94] that every high degree contains a strongly deep set.

Corollary 6.15. *Every high degree contains a strongly deep set.*

Proof. We use the set B constructed in Lemma 6.11. We know that for every computable time bound t there is a constant c_t such that for all n:

$$\mathrm{C}^t(B \upharpoonright n) \geq c_t \cdot n \quad \text{and} \quad \mathrm{C}(B \upharpoonright n) \leq 2 \log n.$$

Since $\mathrm{K}^t(x) \geq \mathrm{C}^t(x)$ and $2\mathrm{C}(x) \geq \mathrm{K}(x)$ for all words x, it follows that B is in $\widehat{\mathrm{D}}_c^t$ for all c and t. Using Lemma 6.5 it follows that B is strongly deep. □

CHAPTER 7

Time bounded complexity and Solovay functions

Prefix-free Kolmogorov complexity K is not computable and in fact does not even allow for unbounded computable lower bounds. The argument may be considered a version of the Berry Paradox: Assume the function f is unbounded, computable and a lower bound for Kolmogorov complexity. Look for the smallest $n \in \mathbb{N}$, such that $f(n) > k$ for some fixed k. By definition, n's complexity is at least k. But on the other hand it can be described as "the smallest natural number n such that $f(n) > k$"; and since f is computable this description is of size $\log k + \mathrm{O}(1)$, which is a contradiction for large enough k.

However, there *are* computable upper bounds for K and, by a construction that goes back to Solovay [BD09, Sol75], there are even computable upper bounds that are non-trivial in the sense that g agrees with K, up to some additive constant, on infinitely many places n. Such upper bounds are called Solovay functions.

For the considerations in this chapter, it makes sense to identify words with natural numbers as described in the introduction and to look at Kolmogorov complexity as a function from $\mathbb{N}$ to $\mathbb{N}$ instead of from $\{0,1\}^{<\infty}$ to $\mathbb{N}$.

For any computable time-bound t, the time-bounded version K^t of K is obviously a computable upper bound for K, and we show that K^t is indeed a computable Solovay function in case $c_0 n \leq t(n)$ for some appropriate constant c_0. As a corollary, we obtain that the Martin-Löf randomness of the various variants of Chaitin's Ω extends to the time-bounded case in so far as for any t as above, the real number

$$\Omega_{\mathrm{K}^t} = \sum_{n \in \mathbb{N}} \frac{1}{2^{\mathrm{K}^t(n)}}$$

is Martin-Löf random. The corresponding proof exploits the result by Bienvenu and Downey [BD09] that a computable function g such that $\Omega_g = \sum 2^{-g(n)}$ converges is

a Solovay function if and only if Ω_g is Martin-Löf random. In fact, this equivalence extends by an even simpler proof to the case of functions g that are just right-computable, i.e., effectively approximable from above, and one then obtains as special cases the result of Bienvenu and Downey and a related result of Miller where the role of g is played by the fixed right-computable but non-computable function K.

An open problem that received some attention recently [GBG09, DH10, Nie09] is whether the class of K-trivial sets coincides with the class of sets that are $g(n)$-jump-traceable for all computable functions g such that $\sum 2^{-g(n)}$ converges. As a step in the direction of a characterization of K-triviality in terms of jump-traceability, we demonstrate that a set A is K-trivial if and only if A is $\mathrm{O}(g(n)-\mathrm{K}(n))$-jump traceable for all computable Solovay functions g, where the equivalence remains true when we restrict attention to functions g of the form K^t, either for a single or all functions t as above.

7.1 Solovay functions and Martin-Löf randomness

Definition 7.1 (Li, Vitányi [LV08]). *A function $f:\mathbb{N}\to\mathbb{N}$ is called an A-*Solovay function *if $\mathrm{K}^A(n)\leq^+ f(n)$ for all n and $\mathrm{K}^A(n)=^+ f(n)$ for infinitely many n. If $A=\emptyset$, we say that f is a* Solovay function.

Solovay [Sol75, BD09] had already constructed computable Solovay functions and by slightly varying the standard construction, next we observe that time-bounded prefix-free Kolmogorov complexity indeed provides natural examples of computable Solovay functions.

Theorem 7.2. *There is a constant c_0 such that time-bounded prefix-free Kolmogorov complexity K^t is a computable Solovay function for any computable function $t:\mathbb{N}\to\mathbb{N}$ such that $c_0 n\leq t(n)$ holds for almost all n.*

Proof. Fix a standard effective and effectively invertible pairing function $\langle .,.\rangle$ from $\mathbb{N}^2$ to $\mathbb{N}$ and define a tripling function $[.,.,.]$ from $\mathbb{N}^3$ to $\mathbb{N}$ by letting

$$[s,\sigma,n]=1^s0\langle\sigma,n\rangle.$$

Let M be a Turing machine with two tapes that on input σ uses its first tape to simulate the universal machine $\mathbb{U}$ on input σ and, in case $\mathbb{U}(\sigma)=n$, to compute $\langle\sigma,n\rangle$, while maintaining on the second tape a unary counter for the number of steps of M required for these computations. In case eventually $\langle\sigma,n\rangle$ has been computed with final counter value s, the output of M is $z=[s,\sigma,n]$, where by construction in this case the total running time of M is in $\mathrm{O}(s)$.

Call z of the form $[s,\sigma,n]$ a Solovay triple in case $M(\sigma)=z$, σ is an optimal code for n, i.e., $\mathrm{K}(n)=|\sigma|$ and s is the number of steps it takes until the computation of M on input σ stops. For some appropriate constant c_0 and any computable

function t that eventually is at least $c_0 n$, for almost all such triples z it then holds that

$$\mathrm{K}(z) =^{+} \mathrm{K}^t(z),$$

because given a code for M and σ, by assumption the universal machine $\mathbb{U}$ can simulate the computation of the two-tape machine M with input σ with linear overhead, hence $\mathbb{U}$ uses time $\mathrm{O}(s)$ plus the constant time required for decoding M, i.e., time at most $c_0|z|$. □

Next we derive a unified form of a characterization of Solovay functions in terms of Martin-Löf randomness of the corresponding Ω-number due to Bienvenu and Downey [BD09] and a result of Miller [Mil10] that asserts that the notions of weakly low and low for Ω coincide. Before, we review some standard notation and facts relating to Ω-numbers.

Definition 7.3. *For a function $f : \mathbb{N} \to \mathbb{N}$, the* Ω-number *of f is*

$$\Omega_f := \sum_{n \in \mathbb{N}} 2^{-f(n)}.$$

We write Ω^A_{K} for $\sum_{n \in \mathbb{N}} 2^{-\mathrm{K}^A(n)}$.

Definition 7.4. *A function $f : \mathbb{N} \to \mathbb{N}$ is an* information content measure relative to a set *A in case f is right-computable with access to the oracle A and Ω_f converges; furthermore, the function f is an* information content measure *if it is an information content measure relative to the empty set.*

The following remark describes for a given information content measure f an approximation from below to Ω_f that has certain special properties. For the sake of simplicity, in the remark only the oracle-free case is considered and the virtually identical considerations for the general case are omitted.

Remark 7.5. *For a given information content measure f, we fix as follows a non-decreasing computable sequence $a_0, a_1, \ldots$ that converges to Ω_f and call this sequence the* canonical approximation *of Ω_f.*

First, we fix some standard approximation to the given information content measure f from above, i.e., a computable function $(n, s) \mapsto f_s(n)$ such that for all n the sequence $f_0(n), f_1(n), \ldots$ is a non-ascending sequence of natural numbers that converges to $f(n)$, where we assume in addition that $f_s(n) - f_{s+1}(n) \in \{0, 1\}$. Then in order to obtain the a_i, let $a_0 = 0$ and given a_i, define a_{i+1} by searching for the next pair of the form $(n, 0)$ or the form $(n, s+1)$ where in addition it holds that $f_s(n) - f_{s+1}(n) = 1$ (with some ordering of pairs understood), let

$$d_i = 2^{-f_0(n)} \quad \text{or} \quad d_i = 2^{-f_{s+1}(n)} - 2^{-f_s(n)} = 2^{-f_s(n)},$$

respectively, and let $a_{i+1} = a_i + d_i$. *Furthermore, in this situation, say that the increase by* d_i *from* a_i *to* a_{i+1} *occurs* due to n.

It is well-known [DH10] that among all right-computable functions exactly the information content measures are, up to an additive constant, upper bounds for the prefix-free Kolmogorov complexity K. The same applies relative to a set A.

Theorem 7.6 unifies two results by Bienvenu and Downey [BD09] and by Miller [Mil10], which are stated below as Corollaries 7.7 and 7.10. The proof of the backward direction of the equivalence stated in Theorem 7.6 is somewhat more direct and uses different methods when compared to the proof of Bienvenu and Downey, and is quite a bit shorter than Miller's proof, though the main trick of delaying the enumeration via the notion of a matched increase is already implicit there [DH10, Mil10]. Note in this connection that Bienvenu has independently shown that Miller's result can be obtained as a corollary to the result of Bienvenu and Downey [DH10].

Theorem 7.6. *Let* f *be an information content measure relative to a set* A. *Then* f *is an* A*-Solovay function if and only if* Ω_f *is Martin-Löf random relative to* A.

In the proof we will use the classic Kraft-Chaitin Theorem (see Theorem 2.2.17 in Nies [Nie09]). A bounded request set is a computably enumerable list of pairs (l_i, w_i) for $i \in \mathbb{N}$ such that $\sum_i 2^{-l_i} \leq 1$. A bounded request set is a "wish list" on which we can place requests such as "Ensure that word w has a short description of length l." The Kraft-Chaitin Theorem now states that this list can be effectively converted into a description of a prefix-free machine M such that for every word w_i there is a description d_i with $|d_i| = l_i$ such that $M(d_i) = w_i$. In other words, if our wish list is not too demanding, the theorem guarantees that there exists a machine that meets our demands.

Moreover, the theorem guarantees that if we even have $\sum_i 2^{-l_i} \leq 2^{-k}$ for some k, then M only terminates on a set of inputs of measure 2^{-k}. We can use this to economize on description lengths: Simply modify the bounded request set by replacing each l_i by $l_i - k$. We then still have $\sum_i 2^{-(l_i-k)} \leq 1$, which means we can apply the theorem also to the new bounded requested set. That way, we will get a new valid prefix-free machine M' that still generates the same words w_i, but from descriptions that are even k bits shorter than before. We will use this trick in the following proof.

Proof of Theorem 7.6. We first show the backwards direction of the equivalence asserted in the theorem, where the construction and its verification bear some similarities to Kučera and Slaman's [KS01] proof that left-computable sets that are not Solovay complete cannot be Martin-Löf random. We assume that f is not an A-Solovay function and construct a sequence $U_0, U_1, \ldots$ of sets that is a Martin-Löf test relative to A and covers Ω_f. In order to obtain the component U_c, let $a_0, a_1, \ldots$ be the canonical approximation to Ω_f where in particular $a_{i+1} = a_i + d_i$ for increases d_i

that occur due to some n. Let $b_{i,n}$ be the sum of the first i increases d_j that are due to n. Say that $b_{i,n}$ is c-matched if it holds that

$$b_{i,n} \leq \frac{2^{-\mathrm{K}^A(n)}}{2^{c+1}}. \tag{7.1}$$

For every $b_{i,n}$ for which it could be verified that it is c-matched, let j be largest index such that d_j contributes to $b_{i,n}$. Now add an interval of size $2d_j$ to U_c, where this interval either starts at the maximum place that is already covered by U_c or at a_ℓ, whichever is larger. Here ℓ is the number of steps in the approximation of Ω_f that we have made, where we need to make sure that ℓ is at least j (if it is not we can just approximate Ω_f further until it is).

By construction, for all $b_{i,n}$ that are c-matched, (7.1) together with the trivial bound $\sum_{n\in\mathbb{N}} 2^{-\mathrm{K}^A(n)} \leq 1$ implies the measure bound 2^{-c} for U_c. Also, the sets U_c are uniformly c.e. relative to A, hence $U_0, U_1, \ldots$ is a Martin-Löf test relative to A. Furthermore, this test covers Ω_f because by the assumption that f is not an A-Solovay function, and by the fact stated above that information content measures relative to A are upper bounds for K^A (up to a constant), it holds that

$$\lim_{n\to\infty} (f(n) - \mathrm{K}^A(n)) = \infty.$$

Hence for any fixed c, for any j large enough the $b_{i,n}$'s to which d_j contributes will eventually become c-matched, resulting in the addition of an interval to U_c. This makes sure that almost every a_ℓ is contained in one of the intervals from which U_c is built. At any moment the sum of the still missing increases is obviously equal to the difference between the current value a_ℓ of the approximation to Ω_f and Ω_f itself. Since we always add intervals twice as long as the increase, this ensures that Ω_f is contained in U_c.

For ease of reference, we review the proof of the forward direction of the equivalence asserted in the theorem, which follows by the same line of standard argument that has already been used by Bienvenu and Downey and by Miller. For a proof by contraposition, assume that Ω_f is not Martin-Löf random relative to A, that is, for every constant c there is a prefix σ_c of Ω_f such that $\mathrm{K}^A(\sigma_c) \leq |\sigma_c| - 2c$. Again consider the canonical approximation $a_0, a_1, \ldots$ to Ω_f where $f(n,0), f(n,1), \ldots$ is the corresponding effective approximation from above to $f(n)$ as in Remark 7.5. Moreover, for σ_c as above we let s_c be the least index s such that a_s exceeds σ_c. Since $\sigma_c \sqsubseteq \Omega_f$ this implies $\sigma_c \leq a_{s_c} \leq \Omega_f$; so the sum over all values $2^{-f(n)}$, where the n are such that none of the increases d_0 through d_{s_c} was due to n, is at most $2^{-|\sigma_c|}$. Hence all pairs of the form $(f(n,s) - |\sigma_c| + 1, n)$ for such n and s where either $s = 0$ or $f(n,s)$ differs from $f(n,s-1)$ form a sequence of Kraft-Chaitin axioms, which is uniformly effective in c and σ_c relative to oracle A. Observe that the approximation

$f(n,s)$ will eventually reach the correct value $f(n)$; so for each n, there is an axiom of the form $(f(n)-|\sigma_c|+1,n)$. Also, since we only add axioms to the KC set when the approximation of $f(n)$ has actually changed, the sum of all terms 2^{-k} over all axioms of the form (k,n) is less than $2^{-f(n)-|\sigma_c|}$.

Now consider a prefix-free Turing machine M with oracle A that given codes for c and σ_c and some other word p as input, first computes c and σ_c, then searches for s_c, and finally outputs the word that is coded by p according to the Kraft-Chaitin axioms for c, if such a word exists. If we let d be the coding constant for M, we have for all sufficiently large c and n that

$$\mathrm{K}^A(n) \leq 2\log c + \mathrm{K}^A(\sigma_c) + f(n) - |\sigma_c| + 1 + d \leq f(n) - c. \qquad \square$$

As special cases of Theorem 7.6 we obtain the following results by Bienvenu and Downey [BD09] and by Miller [Mil10], where the former one is immediate and for the latter one it suffices to observe that the definition of the notion low for Ω in terms of Chaitin's Ω number

$$\Omega := \sum_{\{x\colon \mathbb{U}(x)\downarrow\}} 2^{-|x|}.$$

is equivalent to a definition in terms of Ω_{K}.

Corollary 7.7 (Bienvenu and Downey)**.** *A computable information content measure f is a Solovay function if and only if Ω_f is Martin-Löf random.*

Definition 7.8. *A set A is called* low for Ω *if Ω is Martin-Löf random relative to A. A is low for Ω_f if Ω_f is Martin-Löf random relative to A.*

Definition 7.9. *A set A is called* weakly low for K *iff there are infinitely many n such that $\mathrm{K}(n) \leq^+ \mathrm{K}^A(n)$.*

Corollary 7.10 (Miller)**.** *A set A is weakly low for* K *if and only if A is low for Ω.*

Proof. In order to see the latter result, it suffices to let $f = \mathrm{K}$ and to recall that for this choice of f the properties of A that occur in the two equivalent assertions in the conclusion of Theorem 7.6 coincide with the concepts weakly low and low for Ω_{K}. But the latter property is equivalent to being low for Ω, because of Remark 7.11 below. $\square$

Remark 7.11. *Because all left-computable Martin-Löf random sets are mutually Solovay equivalent, it follows that a set A is low for Ω if and only if any left-computable random set is Martin-Löf random relative to A if and only if all left-computable random sets are Martin-Löf random relative to A [Nie09, Theorem 3.2.29].*

By Corollary 7.7 and Theorem 7.2 it is immediate that the known Martin-Löf randomness of Ω_{K} extends to the time-bounded case.

Corollary 7.12. *There is a constant c_0 such that $\Omega_{\mathrm{K}^t} := \sum_{x\in\mathbb{N}} 2^{-\mathrm{K}^t(x)}$ is Martin-Löf random for any computable function t where $c_0 n \leq t(n)$ for almost all n.*

7.2 Solovay functions and jump-traceability

In an attempt to define K-triviality without resorting to effective randomness or measure, Barmpalias, Downey and Greenberg [GBG09] searched for characterizations of K-triviality via jump-traceability. They demonstrated that K-triviality is not implied by being h-jump-traceable for all computable functions h such that $\sum_n 1/h(n)$ converges. Subsequently, the following question received some attention: Can K-triviality be characterized by being g-jump traceable for all computable functions g such that $\sum 2^{-g(n)}$ converges, that is, for all computable functions g that, up to an additive constant term, are upper bounds for K?

We will now argue that Solovay functions can be used for a characterization of K-triviality in terms of jump traceability. However, we will not be able to completely avoid the notion of Kolmogorov complexity.

Definition 7.13. *A set A is* K-trivial *if $\mathrm{K}(A\upharpoonright n) \leq^+ \mathrm{K}(n)$ for all n.*

Recall the definition of a trace from section 4.1.

Definition 7.14. *Let $h\colon \mathbb{N}\to\mathbb{N}$ be a computable function. A set A is* $\mathrm{O}(h(n))$-jump-traceable *if there is a function $h'\in \mathrm{O}(h(n))$ such that for every Φ partially computable in A there is an h'-bounded c.e. trace for Φ.*

Theorem 7.15. *There is a constant c_0 such that the following assertions are equivalent for any set A.*

(i) *A is* K*-trivial.*

(ii) *A is $\mathrm{O}(g(n)-\mathrm{K}(n))$-jump-traceable for every computable Solovay function g.*

(iii) *A is $\mathrm{O}(\mathrm{K}^t(n)-\mathrm{K}(n))$-jump-traceable for all computable functions t where for almost all n, $c_0 n \leq t(n)$.*

(iv) *A is $\mathrm{O}(\mathrm{K}^t(n)-\mathrm{K}(n))$-jump-traceable for some computable function t where for almost all n, $c_0 n \leq t(n)$.*

Proof. That (ii) implies (iii) is immediate by Theorem 7.2, and the implication from (iii) to (iv) is trivially true.

(i) implies (ii): First, let A be K-trivial and let Φ^A be any partially A-computable function. Let $\langle .,. \rangle$ be some standard effective pairing function. Since A is K-trivial and hence low for K, we have

$$\mathrm{K}(\langle n, \Phi^A(n)\rangle) =^+ \mathrm{K}^A(\langle n, \Phi^A(n)\rangle) =^+ \mathrm{K}^A(n) =^+ \mathrm{K}(n), \tag{7.2}$$

whenever $\Phi^A(n)$ is defined. Observe that the constant that is implicit in the relation $=^+$ depends only on A in the case of the first and last relation symbol, but depends also on Φ in case of the middle one. Let d be a constant such that $\mathrm{K}(\langle n, \Phi^A(n)\rangle) \leq \mathrm{K}(n) + d$ for all n as above.

By the coding theorem there can be at most constantly many pairs of the form (n, y) such that $\mathrm{K}(n, y)$ and $\mathrm{K}(n)$ differ at most by a constant, and given n, $\mathrm{K}(n)$ and the constant, we can enumerate all such pairs.

So let c be a constant such that for all n,

$$\#\{\sigma : |\sigma| \leq \mathrm{K}(n) + d \text{ and } \mathbb{U}(\sigma) = \langle n, y\rangle \text{ for some } y\} \leq c. \tag{7.3}$$

If we knew $\mathrm{K}(n)$, we could build a trace T_n for $\Phi^A(n)$ by simply trying to compute $\mathbb{U}(\sigma)$ for all strings σ of length at most $\mathrm{K}(n) + d$ and, whenever one such computation converges and outputs some string of the form $\langle n, y\rangle$, putting y in T_n. Then all T_n would have size at most c by (7.3) and $\Phi^A(n) \in T_n$ would follow by (7.2).

Since $\mathrm{K}(n)$ is not computable, this strategy will not work. Instead, we computably approximate $\mathrm{K}(n)$ from above by a decreasing sequence $\mathrm{K}_s(n)$. Here, as we know that $\mathrm{K}(n) \leq g(n) + \mathrm{O}(1)$, and since this upper bound is computable, we may as well assume that $\mathrm{K}_0(n) \leq g(n) + \mathrm{O}(1)$. As soon as a value $\mathrm{K}_s(n)$ is reached in the approximation, we apply the strategy for enumerating T_n as described, but with $\mathrm{K}_s(n)$ in place of $\mathrm{K}(n)$. We can stop the enumeration for the current K_s as soon as c elements have been enumerated into T_n. As soon as a new value $\mathrm{K}_{s+1}(n) < \mathrm{K}_s(n)$ is reached, we start the strategy anew, again enumerating up to c elements etc. Eventually, $\mathrm{K}_s(n)$ will drop to the true value $\mathrm{K}(n)$ and by (7.2) we can be sure that $\Phi^A(n)$ will be enumerated if it is defined. Since the value of K_s drops at most $g(n) - \mathrm{K}(n) + \mathrm{O}(1)$ times and for each change we enumerate at most c elements, the size of the trace thus enumerated can be at most $c \cdot (g(n) - \mathrm{K}(n) + \mathrm{O}(1))$.

(iv) implies (i): Let c_0 be the constant from Theorem 7.2 and let t be a computable time bound such that (iv) is true for this value of c_0. Then K^t is a computable Solovay function by choice of c_0.

Recall the tripling function $[.,.,.]$ and the concept of a Solovay triple $[s, \sigma, n]$ from the proof of Theorem 7.2, and define a partial A-computable function Φ that maps any Solovay triple $[s, \sigma, n]$ to $A \upharpoonright n$. Then given an optimal code σ for n, one can compute the corresponding Solovay triple $z = [s, \sigma, n]$, where then $\mathrm{K}^t(z)$ and $\mathrm{K}(z)$ differ only by a constant, hence, by $\mathrm{O}(\mathrm{K}^t(n) - \mathrm{K}(n))$-traceability, the trace

of Φ^A at z has constant size and contains the value $A \upharpoonright n$, that is, we have

$$\mathrm{K}(A \upharpoonright n) \leq^+ |\sigma| = \mathrm{K}(n),$$

hence A is K-trivial. □

Bibliography

[AKRR03] Eric Allender, Michal Koucky, Detlef Ronneburger, and Sambuddha Roy. Derandomization and distinguishing complexity. In *Annual IEEE Conference on Computational Complexity*, pages 209–220, Los Alamitos, CA, USA, 2003.

[AS98] Klaus Ambos-Spies. Algorithmic randomness revisited. In *Language, Logic and Formalization of Knowledge. Coimbra Lecture and Proceedings of a Symposium held in Siena in September 1997*, pages 33–52, 1998.

[Bar68] Janis Barzdin. Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set. *Soviet Math. Dokl.*, 9:1251–1254, 1968.

[BD09] Laurent Bienvenu and Rod Downey. Kolmogorov complexity and solovay functions. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 147–158, 2009.

[BDG09] George Barmpalias, Rod Downey, and Noam Greenberg. K-trivial degrees and the jump-traceability hierarchy. *Proc. Amer. Math. Soc.*, 137(6):2099–2109, 2009.

[Ben95] Charles H. Bennett. Logical depth and physical complexity. In *The universal Turing machine (2nd ed.): a half-century survey*, pages 207–235. Springer, 1995.

[BFL01] Harry Buhrman, Lance Fortnow, and Sophie Laplante. Resource-bounded Kolmogorov complexity revisited. *SIAM J. Comput.*, 31(3):887–905, 2001.

[BHKM] Laurent Bienvenu, Rupert Hölzl, Thorsten Kräling, and Wolfgang Merkle. Separations of non-monotonic randomness notions. *Journal of Logic and Computation*. 22 pages. In print.

[BHKM09] Laurent Bienvenu, Rupert Hölzl, Thorsten Kräling, and Wolfgang Merkle. Separations of non-monotonic randomness notions. In *Proceedings of the Sixth International Conference on Computability and Complexity in Analysis*, Dagstuhl Seminar Proceedings, 2009. 12 pages (electronic).

[BM07] Laurent Bienvenu and Wolfgang Merkle. Reconciling data compression and Kolmogorov complexity. In *Proceedings of the 34th International Conference on Automata, Languages and Programming (ICALP)*, pages 643–654, Springer, Berlin, 2007.

[BvMR+00] Harry Buhrman, Dieter van Melkebeek, Kenneth Regan, D. Sivakumar, and Martin Strauss. A generalization of resource-bounded measure, with application to the BPP vs. EXP problem. *SIAM Journal on Computing*, 30(2):576–601, 2000.

[CDG08] Peter Cholak, Rod Downey, and Noam Greenberg. Strong jump-traceability I: The computably enumerable case. *Advances in Mathematics*, 217(5):2045–2074, 2008.

[CHV93] Jin-Yi Cai, Lane A. Hemachandra, and Jozef Vyskoč. Promises and fault-tolerant database access. In *Complexity Theory: Current Research*, pages 101–146. Cambridge University Press, 1993.

[DH10] Rod Downey and Denis Hirschfeldt. *Algorithmic Randomness and Complexity*. Springer, 2010.

[ESY84] Shimon Even, Alan L. Selman, and Yacov Yacobi. The complexity of promise problems with applications to public-key cryptography. *Inform. and Control*, 61(2):159–173, 1984.

[FGSW] Johanna Franklin, Noam Greenberg, Frank Stephan, and Guohua Wu. Anti-complexity, lowness and highness notions, and reducibilities with tiny use. Manuscript, 2009.

[FK96] Lance Fortnow and Martin Kummer. On resource-bounded instance complexity. *Theor. Comput. Sci.*, 161(1-2):123–140, 1996.

[FS10] Johanna Franklin and Frank Stephan. Schnorr trivial sets and truth-table reducibility. *Journal of Symbolic Logic*, 75:501–521, 2010.

[GBG09] R. Downey G. Barmpalias and N. Greenberg. K-trivial degrees and the jump-traceability hierarchy. *Proc. Amer. Math. Soc.*, 2009.

[HKM09] Rupert Hölzl, Thorsten Kräling, and Wolfgang Merkle. Time-bounded Kolmogorov complexity and Solovay functions. In *Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science*, pages 392–402, August 2009.

[HM08] Rupert Hölzl and Wolfgang Merkle. Generation complexity versus distinction complexity. In *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation*, pages 457–466, April 2008.

[HM10] Rupert Hölzl and Wolfgang Merkle. Traceable sets. In *Proceedings of the World Computer Congress: Theoretical Computer Science*, September 2010. 15 pages. Accepted for publication.

[JLL94] David W. Juedes, James I. Lathrop, and Jack H. Lutz. Computational depth and reducibility. *Theor. Comput. Sci.*, 132(1-2):37–70, 1994.

[Joc89] Carl G. Jockusch, Jr. Degrees of functions with no fixed points. In *Proceedings of the Eighth International Congress of Logic, Methodology and Philosophy of Science (Moscow 1987)*, pages 191–201. 1989.

[Kan70] Max I. Kanovich. On the complexity of enumeration and decision of predicates. *Soviet Math. Dokl.*, 11:17–20, 1970.

[KHMS] Bjørn Kjos-Hanssen, Wolfgang Merkle, and Frank Stephan. Kolmogorov complexity and the recursion theorem. *Trans. Amer. Math. Soc.* In print.

[KL10] Bart Kastermans and Steffen Lempp. Comparing notions of randomness. *Theor. Comput. Sci.*, 411:602–616, 2010.

[KS01] Antonín Kucera and T. Slaman. Randomness and recursive enumerability. *SIAM J. Comput.*, 31(1):199–211, 2001.

[Kum96] Martin Kummer. Kolmogorov complexity and instance complexity of recursively enumerable sets. *SIAM J. Comput.*, 25(6):1123–1143, 1996.

[Lev84] Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Inf. Control*, 61(1):15–37, 1984.

[LL99] James I. Lathrop and Jack H. Lutz. Recursive computational depth. *Inf. Comput.*, 153(2):139–172, 1999.

[LV08] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 2008.

[Mar66] Donald A. Martin. Classes of recursively enumerable sets and degrees of unsolvability. *Z. Math. Logik Grundlagen Math.*, 12:295–310, 1966.

[Mer03] Wolfgang Merkle. The Kolmogorov-Loveland stochastic sequences are not closed under selecting subsequences. *Journal of Symbolic Logic*, 68:1362–1376, 2003.

[Mer08] Wolfgang Merkle. The complexity of stochastic sequences. *Journal of Computer and System Sciences*, 74(3):350–357, 2008.

[Mil10] Joseph S. Miller. The K-degrees, low for K-degrees and weakly low for K-degrees. *Notre Dame Journal of Formal Logic*, 50(4)(4):381–391, 2010.

[MMN+06] Wolfgang Merkle, Joseph S. Miller, André Nies, Jan Reimann, and Frank Stephan. Kolmogorov-Loveland randomness and stochasticity. *Annals of Pure and Applied Logic*, 138(1-3):183–210, 2006.

[MN06] Joseph Miller and André Nies. Randomness and computability: open questions. *Bulletin of Symbolic Logic*, 12(3):390–410, 2006.

[MSU98] Andrei A. Muchnik, Alexei Semenov, and Vladimir Uspensky. Mathematical metaphysics of randomness. *Theoretical Computer Science*, 207(2):263–317, 1998.

[Nie09] André Nies. *Computability and Randomness*. Oxford University Press, 2009.

[Sch71] Claus Schnorr. *Zufälligkeit und Wahrscheinlichkeit*, volume 218 of *Lecture Notes in Mathematics*. Springer-Verlag, 1971.

[Sch73] Claus P. Schnorr. Process complexity and effective random tests. *J. Comput. System Sci.*, 7:376–388, 1973.

[She10] Alexander Shen. Private communication. February 2010.

[Sim75] Janos Simon. On some central problems in computational complexity. Technical report, Cornell University, Ithaca, NY, USA, 1975.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 330–335, New York, NY, USA, 1983. ACM.

[Sol75] Robert M. Solovay. Draft of paper (or series of papers) on Chaitin's work. 1975. Unpublished notes.

[Ste10] Frank Stephan. Private communication. April 2010.

[TZ01] Sebastiaan A. Terwijn and Domenico Zambella. Computational randomness and lowness. *Journal of Symbolic Logic*, 66(3):1199–1205, 2001.