Randomness and Computability

by

Adam Richard Day

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the requirements for the degree of
Doctor of Philosophy

Victoria University of Wellington
2011

This thesis is dedicated to Heather.

Abstract

This thesis establishes significant new results in the area of algorithmic randomness. These results elucidate the deep relationship between randomness and computability.

A number of results focus on randomness for finite strings. Levin introduced two functions which measure the randomness of finite strings. One function is derived from a universal monotone machine and the other function is derived from an optimal computably enumerable semimeasure. Gács proved that infinitely often, the gap between these two functions exceeds the inverse Ackermann function (applied to string length). This thesis improves this result to show that infinitely often the difference between these two functions exceeds the double logarithm. Another separation result is proved for two different kinds of process machine.

Information about the randomness of finite strings can be used as a computational resource. This information is contained in the overgraph. Muchnik and Positselsky asked whether there exists an optimal monotone machine whose overgraph is not truth-table complete. This question is answered in the negative. Related results are also established.

This thesis makes advances in the theory of randomness for infinite binary sequences. A variant of process machines is used to characterise computable randomness, Schnorr randomness and weak randomness. This result is extended to give characterisations of these types of randomness using truth-table reducibility. The computable Lipschitz reducibility measures both the relative randomness and the relative computational power of real numbers. It is proved that the computable Lipschitz degrees of computably enumerable sets are not dense.

Infinite binary sequences can be regarded as elements of Cantor space. Most research in randomness for Cantor space has been conducted using the uniform measure. However, the study of non-computable measures has led to interesting results. This thesis shows that the two approaches that have been used to define randomness on Cantor space for non-computable measures: that of Reimann and Slaman, along with the uniform test approach first introduced by Levin and also used by Gács, Hoyrup and Rojas, are equivalent. Levin established the existence of probability measures for which all infinite

sequences are random. These measures are termed neutral measures. It is shown that every PA degree computes a neutral measure. Work of Miller is used to show that the set of atoms of a neutral measure is a countable Scott set and in fact any countable Scott set is the set of atoms of some neutral measure. Neutral measures are used to prove new results in computability theory. For example, it is shown that the low computable enumerable sets are precisely the computably enumerable sets bounded by PA degrees strictly below the halting problem.

This thesis applies ideas developed in the study of randomness to computability theory by examining indifferent sets for comeager classes in Cantor space. A number of results are proved. For example, it is shown that there exist 1-generic sets that can compute their own indifferent sets.

Acknowledgments

Foremost thanks go to my wife Heather Day and my supervisor Rod Downey. Heather has provided me with amazing love and support throughout this long and frugal undertaking. Rod has been immensely generous. He has not just supervised this thesis but he has tried to teach me how to be a mathematician and I could not wish for better tuition.

Noam Greenberg has assisted the supervision of this thesis. Not only has Noam improved the quality of this thesis, he has made the production of it much more fun. The same can be said for my co-authors: Joseph Miller and Jan Reimann.

I would like to thank Péter Gács for his interest in, and comments on, results in this thesis. I also appreciate the work of the anonymous reviewers of the published material in this thesis who have made many useful comments. I thank Ian Haken for alerting me to an error in an early draft.

The postdocs and students at Victoria have provided a simulating environment in which to conduct research and I would like to thank: George Barmpalias, Laurent Bienvenu, Paul Brodhead, Andrew Fitzgerald, Asher Kach, Keng Meng Ng and Dan Turetsky.

I greatly benefited from traveling to France, Germany, Singapore and the U.S.A. during the course of this thesis. This was assisted by generous support from various institutions and individuals for which I would like to thank: Laurent Bienvenu, Mingzhong Cai, Peter Cholak, Chi Tat Chong, Denis Hirschfeldt, Steffen Lempp, Wolfgang Merkle, Joseph Miller, André Nies, Jan Reimann, Alexander Shen, Richard Shore, Steve Simpson, Bob Soare, Klaus Ambos-Spies, and Yue Yang.

One pleasure of this thesis has been meeting and discussing mathematics with a wide range of people working in algorithmic randomness and related fields. In addition to those mentioned above, I would like to thank Eric Allender, David Diamondstone, Johanna Franklin, Rupert Hölzl, Greg Igusa, Ted Slaman, Nikolai Vereshchagin, Paul Vitányi, Guohua Wu, Liang Yu, and the students and teachers at the 2010 Singapore Summer School.

The School of Mathematics, Statistics and Operations Research has been

very helpful for which I particularly thank the Head of School Megan Clark. Thanks also go to the School Manager, Ginny Whatarau, and the school administrative staff for minimising the administrative burden of undertaking this thesis. The Tertiary Education Commission has been the primary funder of this thesis through one of their last Top Achiever Doctoral Scholarships. The Victoria University Scholarships Office has ably administered this scholarship and special thanks go to the always helpful and proficient Barry Lewis. Additional funding was provided by Victoria University through a Faculty Strategic Grant and a Postgraduate Research Excellence Award.

Finally I would like to thank my parents, for everything.

Contents

Chapter 1

Introduction

1.1 Summary

This thesis continues an area of research, termed algorithmic randomness, that began in the 1960s and has seen significant interest in the last decade. The central tenet of this research area is that a precise mathematical definition of randomness can be given using computability theory. The main theme of this thesis is the deep interplay between algorithmic randomness and classical computability theory. While the field of algorithmic randomness was established by using computability theory to define randomness, recent research has used ideas and results from the study of algorithmic randomness to further develop computability theory.[1] This thesis makes a number of advances. It uncovers new facts about the relationship between different Kolmogorov complexities and how they can be used to characterise random sequences. Random strings are regularly used in algorithm design and this thesis solves a question related to the computational power of these strings. It also provides a solid basis for studying randomness for non-computable probability measures by unifying and extending previous work on this topic. Additionally, this thesis uses results and ideas from algorithmic randomness to prove new theorems in classical computability theory, particularly in the study of 1-generic and weakly 1-generic sequences.

To understand how algorithmic randomness differs from probability theory, consider the set of binary strings of length 20. If an element of this set is selected by flipping a fair coin 20 times, then probability theory tells us that each string is equally likely to occur. However, it does not tell us anything about the intrinsic randomness of the strings themselves. The string

$$00000000000000000000,$$

[1] For example, Downey and Greenberg have used ideas in algorithmic randomness to prove that there is a CEA operator that does not avoid upper cones.

certainly seems less random than the following string, which was obtained by coin flips,

$$11111010110101110010.$$

Research in algorithmic randomness is motivated by a desire to understand when an object, e.g. a finite binary string or an infinite binary sequence, can be called random. In the case of infinite binary sequences, one approach is to consider a sequence to be random if it obeys standard statistical laws. For example, consider the law of large numbers. If we flip a fair coin repeatedly then as a consequence of this law, the ratio of heads to tails should tend to 1 as the number of coin flips increases. Hence it is reasonable to expect a random infinite binary sequence to have this property. In 1919, von Mises suggested calling a sequence random if it obeys the law of large numbers, and additionally, the law of large numbers holds for certain subsequences as well [90]. However, von Mises was not specific about which subsequences should be considered and it would take the development of the field of computability theory to provide a sensible answer to this question.

In the late 1930s, Church and Turing independently solved the *Entscheidungsproblem* posed by Hilbert [18, 19, 85, 86].[2] They showed that there was no algorithm that could be used to determine the truth or falsity of any statement in arithmetic. An essential component of resolving Hilbert's problem was to provide a mathematically robust definition of an algorithm. Turing resolved this problem by defining an abstract computing device, the Turing machine, and defining an algorithm to be any function computable on such a machine. Church's approach used a mathematically equivalent definition of an algorithm. One outcome of this work was the proof of the existence of a universal partial computable function.

Church suggested adapting von Mises's approach and calling an infinite binary sequence random if all its partial computable subsequences obeyed the law of large numbers [20]. These sequences are now termed Church stochastic. Ville showed that Church stochastic sequences exist which do not obey other standard statistical tests such as the law of the iterated logarithm [89].

The question of which statistical tests to use in the definition of a random sequence was resolved by Martin-Löf. Martin-Löf used partial computable functions to generalise the notion of a statistical test [63]. A Martin-Löf test is a type of computable null set. For example, those sequences that *fail* to obey the law of large numbers can be captured in a Martin-Löf test. He defined a random sequence as one that was not contained in any such test. These

[2] Entscheidungsproblem is German for 'decision problem.'

sequences are called Martin-Löf random sequences.[3]

Contemporary with Martin-Löf, and within quick succession of one another, Solomonoff, Kolmogorov and Chaitin realised that a universal partial computable function could be used to quantify the information content of a finite binary string [14, 15, 46, 84]. This led to the identification of the random strings as those strings with maximal information content. The intuition behind this idea is that random strings are incompressible. We will term the functions that quantify the information content of binary strings Kolmogorov complexities.

Early researchers in this area saw the connection between this approach and that of Martin-Löf. An early question was whether Kolmogorov complexity could be used to characterise Martin-Löf random sequences. Levin and Schnorr independently gave a positive answer to this question [55, 80].

The last decade has seen a significant upsurge in work in algorithmic randomness initiated by papers of Calude, Coles, Hertling, Khoussainov and Wang [12, 13]. This work has led to major developments both in the underlying theory of randomness and also in applications to other areas of mathematics. The success of recent research in this area can be seen by the fact that two extensive treatises on this subject have recently been published: *Randomness and Computability* by Nies in 2009, and *Algorithmic Randomness and Complexity* by Downey and Hirschfeldt in 2010.

In Part I of this thesis we investigate the relationship between randomness and computability for finite strings. In Chapter 2 we will see that there are many different types of Kolmogorov complexities. Two of these complexities, plain complexity and prefix-free complexity, have been extensively studied. However, we pay particular attention to some of the less well-known varieties. What these other varieties have in common is that they are all based on continuous transformations. Hence we will term these continuous Kolmogorov complexities. These continuous Kolmogorov complexities have been less well studied, partly because they are technically more difficult to make use of. Nevertheless, the close link between continuous Kolmogorov complexities and measures means that these complexities are well suited for studying randomness.

Process machines are computable functions which preserve the natural partial ordering on finite strings. Chapter 2 draws a distinction between two definitions of process machine that exist in the literature. In Chapter 3 we take a closer look at the complexity measures generated by the different types of

[3]We will provide precise definitions of all terms used in this section in Section 1.2 and Chapter 2.

process machine. We prove that these two definitions of process machine give rise to two different continuous Kolmogorov complexities.

A fundamental question to ask about any Kolmogorov complexity measure is whether or not it is subadditive. A Kolmogorov complexity is subadditive if for any pair of finite binary strings σ and τ, the complexity of the string σ concatenated with τ is less than the sum of the complexities of σ and τ plus some fixed constant. In Chapter 3 we also show that neither of these process complexities is subadditive. The work in Chapter 3 has been published in the *Chicago Journal of Theoretical Computer Science* [24].

Levin introduced two types of continuous Kolmogorov complexities [55]. One of these complexities, K_m, is an application of the idea that the complexity of a string can be measured by its shortest description. The other complexity, KM, has a measure-theoretic definition. These complexities are interesting because K_m is an application of the principle of Occam's razor, and Levin observed that KM is a natural candidate for any *a priori* probability used in Bayesian statistics. Results about the relationship between these two complexities can also be seen as relating these two philosophical principles.

Levin conjectured that the K_m and KM complexities measures might agree within some additive constant. Gács proved, 25 years ago, that infinitely often, the gap between these two functions exceeds the inverse Ackermann function (applied to string length), an *extremely* slow growing function [36]. In Chapter 4 we adapt and enhance Gács's techniques to show that infinitely often the difference exceeds the double logarithm. The work in Chapter 4 has been accepted for publication in the *Transactions of the American Mathematical Society*.

There are two fundamental computably enumerable sets associated with any Kolmogorov complexity. These are the set of non-random strings and the overgraph. The overgraph of a Kolmogorov complexity is the set of pairs $\langle \sigma, n \rangle$ such that the complexity of the finite string σ does not exceed the natural number n. Chapter 5 investigates the computational power of these sets. It follows work of Kummer, Muchnik and Positselsky, and Allender and co-authors. It is easy to show that both the overgraph and the set of non-random strings have the same Turing complexity as the halting problem. The main question is whether this is still true under more refined reductions. Kummer showed that the set of non-random strings defined by plain Kolmogorov complexity is truth-table complete [50]. Muchnik proved that for prefix-free Kolmogorov complexity the set of non-random strings is truth-table complete for some universal machines and not for others [66]. Allender, Buhrman, Koucký, van Melkebeek and Ronneburger established a link between sets of non-random strings and complexity classes. They showed that sets of strings of high Kol-

mogorov complexity are complete for several complexity classes under probabilistic and non-uniform reductions [2].

Muchnik and Positselsky asked whether there exists an optimal monotone machine whose overgraph is not truth-table complete [66]. This chapter answers this question in the negative by proving that the overgraph of any continuous Kolmogorov complexity is truth-table complete. For strict process machines, it is shown that there is a universal machine whose set of non-random strings is not truth-table complete. The work in Chapter 5 has been published in the *Annals of Pure and Applied Logic* [22].

In Part II we examine randomness and computability in Cantor space. In Chapter 6 we investigate a type of process machine considered by Levin in his doctoral dissertation [58]. We call this a quick process machine. We use quick process machines to provide new and simple Kolmogorov complexity-based characterisations of computable randomness, Schnorr randomness and weak randomness. A new technique for building process machines and quick process machines is presented. This technique is similar to the KC theorem for prefix-free machines. Using this technique, a method of translating computable martingales to quick process machines is given. This translation forms the basis for these new randomness characterisations. Quick process machines are closely linked to truth-table reductions. This relationship allows us to characterise computable randomness, Schnorr randomness, and weak randomness purely in terms of truth-table reducibility.

In Chapter 7, which is joint work with Joseph Miller, we examine randomness for non-computable probability measures. Different approaches have been taken to defining randomness for non-computable probability measures. We will explain the approach of Reimann and Slaman, along with the uniform test approach first introduced by Levin and also used by Gács, Hoyrup and Rojas [38, 40, 57, 74, 73]. We will show that these approaches are fundamentally equivalent.

Levin showed that there exist probability measures for which all sequences are random [57]. Gács termed these neutral measures. We show that every PA degree computes a neutral measure. We also show that a neutral measure has no least Turing degree representation and explain why the framework of the continuous degrees (a substructure of the enumeration degrees studied by Miller [64]) can be used to determine the computational complexity of neutral measures. This allows us to show that the Turing ideals below neutral measures are exactly the Scott ideals. This provides us with a complete understanding of the possible sets of atoms of a neutral measure. One simple consequence is that every neutral measure has a Martin-Löf random atom.

The work in Chapter 7 has been accepted for publication in the *Transactions of the American Mathematical Society*.

In Chapter 8, which is joint work with Jan Reimann, we continue our investigation of randomness for non-computable measures. We examine which pairs of infinite binary sequences are relatively random with respect to some probability measure μ such that neither sequence is an atom of μ. We show that if a computably enumerable set is random with respect to some probability measure, and the set is not an atom of the measure, then that computably enumerable set together with the measure can compute the halting problem. This has some important implications to classical computability theory. For example, it is shown that the low computable enumerable sets are precisely those c.e. sets which are bounded by a set of PA degree strictly below the halting problem.

A natural question to ask when studying randomness is when is one infinite binary sequence more random than another. One possible way of ordering sequences according to randomness is given by the computable Lipschitz reducibility introduced by Downey, Hirschfeldt and LaForte [31]. This reducibility links computability and randomness because it measures both relative randomness and relative computational power. In Chapter 9 we prove that the computable Lipschitz degrees of computably enumerable sets are not dense. An immediate corollary is that the Solovay degrees of strongly c.e. reals are not dense. This is also an important result in terms of classical computability theory. Barmpalias and Lewis established that the ordering induced on the c.e. sets by a stronger reducibility, the identity-bounded Turing reducibility, is not dense. The orderings induced on the c.e. sets by weaker reducibilities like weak truth-table reducibility and general Turing reducibility are dense as shown by Ladner and Sasso, and Sacks respectively [54, 77]. Hence this result lies at the boundary between where density and non-density hold. The work in Chapter 9 has been published in the *Annals of Pure and Applied Logic* [23].

Many results about randomness can be regarded as results about forcing with closed sets of positive measure. There are many other types of forcing and often analogous results hold. For example, van Lambalgen's theorem is a measure-theoretic version of a result which holds for Cohen generics. Figueira, Miller, and Nies showed that given any Martin-Löf random sequence X, there is an infinite set I such that no matter how X is changed on the positions specified by I, the resulting sequence remains Martin-Löf random [34]. Such a set I is called an indifferent set. In Chapter 10 we look at indifferent sets for comeager classes in Cantor space. The comeager classes were defined by Baire to capture the notion of a topologically large set. This is

an alternative notion of largeness to that provided by measure theory. In particular, we focus on the comeager classes of 1-generic and weakly 1-generics sequences.

Some previous results in this area have been established by Jockusch and Posner [43]. These can be improved using recent work of Cai and Shore [10]. We prove a number of results about indifferent sets for 1-generic sequences and weakly 1-generic sequences. Some of the results have counterparts in randomness. For example, indifferent sets for 1-generic sequences must be hyperimmune as is the case with indifferent sets for Martin-Löf random sequences. However, many results exhibit striking differences. For example, indifferent sets for Martin-Löf random sequences must compute the halting problem but there are indifferent sets for 1-generic sequences which are low. Further, we will show that there exist 1-generic sequences that can compute their own indifferent sets (the analogous problem for Martin-Löf random sequences remains open). Using a powerful new permitting technique of Downey and Greenberg [26], we establish that there is a broad class of computably enumerable sets that compute such a 1-generic sequence.

1.2 Conventions and background

In this section we will briefly review the basic concepts in computability theory, measure theory and algorithmic randomness that we will make use of.

Computable functions. We take $\omega = \{0, 1, 2, \ldots\}$ to be the set of natural numbers. Given $A \subseteq \omega$, we will regularly equate A with its characteristic function writing $A(x) = 1$ if $x \in A$ and $A(x) = 0$ if $x \notin A$. We call a partial function $f : \omega \to \omega$ computable if there is some Turing machine that on input n halts if and only if $f(n)$ is defined, and further if the Turing machine halts, it does so with output $f(n)$. There exists a universal Turing machine. This provides us with a computable enumeration of all partial computable functions:

$$\varphi_0, \varphi_1, \varphi_2, \ldots.$$

We will write $\varphi_e(x) \downarrow$ if the eth Turing machine halts on input x and $\varphi_e(x) \uparrow$ otherwise. We call $A \subseteq \omega$ computable enumerable, or c.e., if A is the range of a total computable function. The most natural c.e. set is $\{e \in \omega : \varphi_e(e) \downarrow\}$. This is known as the halting problem and it is denoted $\emptyset'$.

Relative computational power. Turing machines can also be used to determine the relative computational power of sets. An oracle Turing machine is

a Turing machine with an additional infinite read-only input tape. There is a universal oracle Turing machine which provides us with a computable enumeration $\Phi_0, \Phi_1, \ldots$ of all oracle Turing machines. If $X \subseteq \omega$, we will write $\Phi_e^X(z)$ to denote the result of the computation of the eth oracle Turing machine with input z and oracle X written on the input tape. If for all z, $\Phi_e^X(z)$ halts and $\Phi_e^X(z) \in \{0,1\}$ then we will equate Φ_e^X with the set $\{z \in \omega : \Phi_e^X(z) = 1\}$. At times we will write $\Phi_e(X; z)$ and $\Phi_e(X)$ for $\Phi_e^X(z)$ and Φ_e^X respectively. We will use the following types of computational reducibility.

(i). *Turing reducibility*: $A \leq_T B$ if there is an oracle Turing machine Φ such that $\Phi^B = A$.

(ii). *Weak truth-table reducibility*: $A \leq_{wtt} B$ if there is an oracle Turing machine Φ such that $\Phi^B = A$, and a computable function $\varphi(n)$ such that the computation of $\Phi^B(n)$ only makes queries of the oracle B for values less than or equal to $\varphi(n)$.

(iii). *Truth-table reducibility*: $A \leq_{tt} B$ if there is an oracle Turing machine Φ such that $\Phi^B = A$ and Φ^C is total for any oracle C.

We can define a degree structure on the subsets of ω using any of these reducibilities. For example, we say $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$. We define the Turing degree of A to be $\deg(A) = \{B : A \equiv_T B\}$. The class of all Turing degrees is a partially ordered set under $\leq_T$ where $\deg(A) \leq_T \deg(B)$ is defined to hold if $A \leq_T B$. If we only consider those degrees that have a computably enumerable member, then we can talk about the Turing degrees of c.e. sets. Similarly we can also talk about the weak truth-table degrees or the truth-table degrees. Given $A \subseteq \omega$, we define the Turing jump of A to be the set $A' = \{e \in \omega : \Phi_e^A(e) \downarrow\}$. We can also relativise the definition of computable enumerability. We call a set X c.e. in A if X is the range of a total A-computable function i.e. the range of Φ_e^A for some e. The Shoenfield Limit Lemma tells us that if $A \subseteq \omega$, then $A \leq_T \emptyset'$ if and only if then there is a total computable function $f : \omega \times \omega \to \{0,1\}$ such that for all x, $\lim_s f(x,s)$ exists and $A(x) = \lim_s f(x,s)$. The function f is called a computable approximation to A. If A is c.e. then A has a computable approximation f such that for all x and s, if $f(x,s) = 1$ then $f(x,s+1) = 1$. The c.e. sets are also known as Σ_1^0 sets, and the sets Turing below $\emptyset'$ are also known as Δ_2^0 sets due to the fact these are characterisations of these levels of the arithmetic hierarchy.

Sequences and strings. We will denote the set of binary strings of length n by $\{0,1\}^n$ and we will take $2^{<\omega}$ to be the set of all finite binary strings. The

relation $\preceq$ on $2^{<\omega}\times 2^{<\omega}$ is defined by $\sigma \preceq \tau$ if σ is an initial segment of τ. We say $\sigma \prec \tau$ if $\sigma \preceq \tau$ and $\sigma \neq \tau$. The relations $\succeq$ and $\succ$ are defined to be the inverse relations of $\preceq$ and $\prec$ respectively. If $\sigma \preceq \tau$ or $\tau \preceq \sigma$ then σ and τ are said to be comparable. This will be written $\sigma \approx \tau$. Otherwise, σ and τ are incomparable and this will be written $\sigma \mid \tau$. The empty string will be represented by λ. If $\sigma \in 2^{<\omega}$, let $|\sigma|$ be the length of σ and if $i \in \omega$ with $0 \leq i < |\sigma|$ let $\sigma(i)$ be the ith bit of σ. The operation of appending a string τ to the end of a finite string σ, will be represented by $\sigma\tau$. We will take $\langle . \rangle : 2^{<\omega} \to \omega$ to be the bijection that maps $\lambda, 0, 1, 00, 01, 10, 11, 000, \ldots$ to $0, 1, 2, 3, 4, 5, 6, 7, \ldots$ respectively. This allows us to apply the computability properties of ω to $2^{<\omega}$ e.g. we can talk about c.e. subsets of $2^{<\omega}$.

Cantor space. We will regard Cantor space as the set of all infinite binary sequences and denote this space by 2^{ω}. If $A \in 2^{\omega}$, and $A = a_0a_1a_2\ldots$, then we will use $A(x)$ to refer to a_x. We will also identify Cantor space with $\mathcal{P}(\omega)$ by the bijection that maps $A \in 2^{\omega}$ to $\{x \in \omega : A(x) = 1\}$. We will extend the relation $\prec$ to $2^{<\omega} \times 2^{\omega}$ by saying $\sigma \prec A$ if σ is an initial segment of A. If $A \in 2^{\omega}$ and $n \in \omega$ then $A \upharpoonright n$ is the finite string of length n that forms an initial segment of A. Further if $\sigma \in 2^{<\omega}$ and $A \in 2^{\omega}$ we will write σA to be the sequence formed by appending A to σ. We will place a topology on Cantor space by taking $\{[\sigma] : \sigma \in 2^{<\omega}\}$ (where $[\sigma] = \{\sigma A : A \in 2^{\omega}\}$) as a basis of open sets. If $X \subseteq 2^{<\omega}$, then $[X] = \bigcup_{\sigma \in X}[\sigma]$.

Measures and premeasures. Let X be a set and $\mathcal{X} \subseteq \mathcal{P}(X)$. We call $\mathcal{X}$ a σ-algebra if:

(i). $\emptyset \in \mathcal{X}$.

(ii). $A \in \mathcal{X}$ implies $X \setminus A \in \mathcal{X}$.

(iii). $\mathcal{X}$ is closed under countable unions.

A measure space is a triple $(X, \mathcal{X}, \mu)$ such that $\mathcal{X}$ is a σ-algebra on X and μ is a function $\mu : \mathcal{X} \to \overline{\mathbb{R}}^{\geq 0}$ (where $\overline{\mathbb{R}}^{\geq 0}$ is the extended non-negative real numbers). The function μ is called a measure and we require it to have the properties that:

(i). $\mu(\emptyset) = 0$.

(ii). If $\{X_i\}_{i\in\omega}$ is a countable collection of pairwise disjoint subsets of $\mathcal{X}$ then $\mu \bigcup_i X_i = \sum_i \mu X_i$.

If $\mu X = 1$ then we call μ a probability measure. In this thesis we will restrict our attention to measures on Cantor space. The σ-algebra will be the Borel algebra, which is the smallest σ-algebra including $\{[\sigma] : \sigma \in 2^{<\omega}\}$. We will also require our measures to be finite, i.e. that the measure of the whole space is not infinite. We will define a premeasure as a function $f : \{[\sigma] : \sigma \in 2^{<\omega}\} \to \mathbb{R}^{\geq 0}$ such that for all $\sigma \in 2^{<\omega}$,

$$f([\sigma]) = f([\sigma 0]) + f([\sigma 1]). \tag{1.2.1}$$

We will often abuse notation and write $f(\sigma)$ for $f([\sigma])$. By the Carathéodory extension theorem we can extend any premeasure to a measure. By the Hahn extension theorem this extension is unique because for any premeasure $f(\lambda) < \infty$. Hence there is a natural bijective correspondence between premeasures and measures. By the uniform or Lebesgue measure on Cantor space, we mean the unique measure μ with the property that for all $\sigma \in 2^{<\omega}$, $\mu[\sigma] = 2^{-|\sigma|}$.

Σ^0_1 and Π^0_1 classes. Let $X \subseteq 2^\omega$. We call X a Σ^0_1 class if $X = [W]$ for some c.e. set $W \subseteq 2^{<\omega}$. We call X a Π^0_1 class if the complement of X is a Σ^0_1 class. The Σ^0_1 classes can be regarded as effectively open sets in Cantor space and the Π^0_1 classes as effectively closed sets. We relativise this definition and call X a $\Sigma^0_1(A)$ class if there is some c.e. in A set of strings W with $X = [W]$ and similarly X a $\Pi^0_1(A)$ class if its complement is $\Sigma^0_1(A)$.

Martin-Löf randomness. Let μ be the uniform measure on Cantor space. A Martin-Löf test is uniform sequence $\{U_n\}_{n\in\omega}$ of Σ^0_1 classes such that for all n, $\mu U_n \leq 2^{-n}$. Given a Martin-Löf test $\{U_n\}_{n\in\omega}$, we say that $X \in 2^\omega$ passes the test if $X \notin \bigcap_n U_n$. We say that $X \in 2^\omega$ is Martin-Löf random if it passes all Martin-Löf tests. We denote the class of all Martin-Löf random sequences by MLR. There exists a single Martin-Löf test $\{U_n\}_{n\in\omega}$ such that $\mathrm{MLR} = 2^\omega \setminus \bigcap_n U_n$ and such a test is termed a universal Martin-Löf test.

Effective real numbers. We will use the following notion of effectiveness for real numbers. The dyadic rationals are $\mathbb{Q}_2 = \{z2^{-n} : z \in \mathbb{Z}, n \in \omega\}$. We say that $x \in \mathbb{R}$ is left-c.e. if $\{q \in \mathbb{Q}_2 : q < x\}$ is a computably enumerable set. Note that it can be shown that x is left-c.e. if and only if $x = \lim_{n\to\infty} q_n$, for some nondecreasing computable sequence of $\{q_n\}_{n\in\omega}$ of dyadic rationals. A function $f : \omega \to \mathbb{R}$ is uniformly left-c.e., if there is a computable function $\hat{f} : \omega \times \omega \to \mathbb{Q}_2$, nondecreasing in the second variable such that for all $a \in \omega$, $f(a) = \lim_{n\to\infty} \hat{f}(a, n)$. We call $\hat{f} : \omega \times \omega \to \mathbb{Q}_2$ an enumeration of $f : \omega \to \mathbb{R}$.

Other. We define $\log x$ to be the least $y \in \omega$ such that $2^y \geq x$. We take $\langle \cdot, \cdot \rangle$ to be a computable bijection $\langle \cdot, \cdot \rangle : \omega \times \omega \to \omega$ such as the Cantor pairing function $\langle x, y \rangle = \frac{1}{2}(x + y)(x + y + 1) + y$. Given two real valued functions f, g on a set X we say that f multiplicatively majorizes g if for some positive $c \in \mathbb{R}$, for all $x \in X$, $f(x) > c \cdot g(x)$. For functions $f, g : \omega \to \omega$, we write: $f(n) \leq g(n) + O(1)$, or $g(n) \geq f(n) - O(1)$ if there is some $c \in \omega$ such that for all $n \in \omega$, $f(n) \leq g(n) + c$.

Further information. For further information on: computability theory see Odifreddi [71], Rogers [75], or Soare [83]; algorithmic randomness see Downey and Hirschfeldt [30], Li and Vitányi [60], or Nies [70]; and measure theory see Bartle [8] or Munroe [67].

Part I

Randomness and Computability for Finite Strings

Chapter 2

Varieties of Kolmogorov Complexity

The main results in Part I are based on continuous Kolmogorov complexities. In this chapter, we will introduce the continuous Kolmogorov complexities used, as well as plain and prefix-free Kolmogorov complexity. We will examine the relationship between these complexities and see how they can be used to characterise the Martin-Löf random sequences.

2.1 Descriptive complexities

A Kolmogorov complexity is a means of quantifying the information content of a string. One approach to defining a Kolmogorov complexity is to quantify the information content of a string in terms of the length of its shortest description. Take $\sigma, \tau \in 2^{<\omega}$, and let F be a mapping from $2^{<\omega}$ to itself. We say τ is an F-description of σ if $F(\tau) = \sigma$. The length of an F-description of σ quantifies the information content of σ, because from the description and the function F we can determine σ. Hence we define the complexity of the string σ with respect to F as:

$$C^F(\sigma) = \begin{cases} \min\{|\tau| : F(\tau) = \sigma\} & \text{if } (\exists \tau \in 2^{<\omega})(F(\tau) = \sigma) \\ \infty & \text{otherwise.} \end{cases}$$

We call a string $\sigma \in 2^{<\omega}$ *random with respect to* F if $C^F(\sigma) \geq |\sigma|$. The intuition behind this definition is that random strings are incompressible. For all n, a simple counting argument establishes that there is a random string of length n. There are 2^n binary strings of length n and $2^n - 1$ descriptions of length less than n.

This approach does not give an absolute notion of randomness for strings. For any string σ, a function could be defined that has a very short description of σ. It is possible, however, to define a notion of randomness *up to a*

constant. First we will restrict ourselves to effective descriptions by only considering partial computable functions. We call a partial computable function $U : 2^{<\omega} \to 2^{<\omega}$ *optimal* if for any other partial computable function $F : 2^{<\omega} \to 2^{<\omega}$, there is some constant d such that $C^U(\sigma) \leq C^F(\sigma) + d$ for all σ. The existence of an optimal partial computable function U can be established by taking an enumeration $F_1, F_2, \ldots$ of all partial computable functions from $2^{<\omega}$ to $2^{<\omega}$. U is then defined by $U(1^e 0\tau) = F_e(\tau)$. This ensures that $C^U(\sigma) \leq C^{F_e}(\sigma) + e + 1$ for all σ. This approach was first suggested by Solomonoff [84], Kolmogorov [46] and Chaitin [14, 15]. As well as being optimal, U is a *universal* partial computable function because for all partial computable functions V, there is some string σ such that for all $\tau \in 2^{<\omega}$, we have that $U(\sigma\tau) = V(\tau)$.

If we take a different optimal partial computable function V, then we get a different complexity C^V. However, as both U and V are optimal, the complexities C^U and C^V can differ only by a constant.

We will often refer to partial computable functions from $2^{<\omega}$ to $2^{<\omega}$ as machines. This terminology comes from the fact that a partial computable function can be computed by a Turing machine.

Definition 2.1.1. The *plain Kolmogorov complexity* is $C(\sigma) = C^U(\sigma)$ where U is a universal machine.

We call $C(\sigma)$ the plain Kolmogorov complexity of σ because there are many other varieties of Kolmogorov complexity. These varieties arose early in the study of algorithmic randomness. A significant impetus towards their development was to characterise the Martin-Löf random sequences in terms of initial segment complexity. On first thought, it might seem $X \in 2^\omega$ should be a Martin-Löf random sequence if all of its initial segments are random strings. Using plain complexity this would mean that $C(X \upharpoonright n) \geq n - O(1)$. However, Martin-Löf showed that no $X \in 2^\omega$ has this property!

Theorem 2.1.2 (Martin-Löf, see Li and Vitányi [60])**.** *For all $X \in 2^\omega$, for all $c \in \omega$, there exists some $n \in \omega$ such that $C(X \upharpoonright n) \leq n - c$.*

This theorem can be proved by building a machine that on input τ uses both the bits of τ and *additionally the length of* τ to determine its output. The proof of this theorem exposes a limitation of plain Kolmogorov complexity. The intention behind Kolmogorov complexity is that $C(\sigma)$ quantifies the information content of σ. This would imply that if $U(\tau) = \sigma$, then solely the information in the *bits* of τ is used by U to produce σ. The fact that the machine can use the length of τ as well, indicates that our definition may need modification.

The varieties of plain Kolmogorov complexity we will consider avoid this problem and arguably capture the original spirit behind Kolmogorov complexity. The principle behind our next definitions is to restrict our attention to subclasses of Turing machines which can only use the bits of the description to determine the output. The types of complexity we will look at derive from: prefix-free machines, process machines and monotone machines. We will see in Theorem 2.4.1 that all the complexities we introduce can be used to give simple characterisations of Martin-Löf randomness.

The first subclass we will look at is the subclass of all prefix-free machines. Prefix-free machines were developed by Levin [56], Gács [35] and Chaitin [16]. A subset $A \subseteq 2^{<\omega}$ is *prefix-free* if for all $\tau_1, \tau_2 \in A$, $\tau_1 \not\preceq \tau_2$. A natural example of a prefix-free set is the set of all telephone numbers. Given any two telephone numbers, it is not possible for one number to be an initial segment of another. The information about the length of the telephone number is included in the number itself. This is why prefix-free sets are sometimes called self-delimiting.

Definition 2.1.3. A *prefix-free machine* is a partial computable function $M : 2^{<\omega} \to 2^{<\omega}$ such that the domain of M is prefix-free.

We can enumerate all partial computable prefix-free machines $\{M_e\}_{e\in\omega}$ and create a universal prefix-free machine U by $U(1^e 0\tau) = M_e(\tau)$.

Definition 2.1.4. The *prefix-free complexity* is $K(\sigma) = C^U(\sigma)$ where U is a universal prefix-free machine.

One reason prefix-free complexity has provided significant insight into algorithmic randomness is the ease with which prefix-free machines can be created. This is a result of the following effective version of the Kraft inequality [47].

Theorem 2.1.5 (Kraft Computable Theorem)**.** *If $D = \{d_i\}_{i\in\omega}$ is a computable sequence in ω such that $\sum_{i=0}^{\infty} 2^{-d_i} \leq 1$, then there exists a function $f_D : \omega \to 2^{<\omega}$ such that:*

(i). The range of f_D is prefix-free.

(ii). For all i, $|f_D(i)| = d_i$.

(iii). For all i, $f_D(i)$ is computable from $d_0, \ldots, d_i$.

Given this theorem, it is possible to turn any c.e. set of pairs $\langle \sigma_i, n_i \rangle_{i\in\omega}$ such that $\sum_i 2^{-n_i} \leq 1$ into a prefix-free machine M with the property that for all i, there is some τ_i of length n_i and $M(\tau_i) = \sigma_i$ (thus $C^M(\sigma_i) \leq n_i$).

The next subclass of Turing machines that we will look at is the subclass of all process machines. The main idea behind a process machine is that it must preserve the ordering of $2^{<\omega}$.

Definition 2.1.6. A *process machine* is a partial computable function $M : 2^{<\omega} \to 2^{<\omega}$ such that if $\tau, \tau' \in \text{dom}(M)$, and $\tau' \preceq \tau$, then $M(\tau') \preceq M(\tau)$.

A natural example of a process is given by the recordings of moves in a game of chess, e.g. 1. e4 e5, 2. Nf3 Nc6 etc. Given the descriptions of the first n moves, it is possible to replay the game up until that point. If we extend the description, we can extend the replay of the game by adding new moves, but we cannot change any moves already defined.

This definition of a process machine is due to Schnorr [80], who compared a process to a book and noted that: "he who wants to understand a book will not read it backwards, since the comments or facts which are given in the first part will help him to understand the subsequent chapters (this means they help him to find regularities in the rest of the book)." Schnorr's definition of a process differs slightly from that given by Levin in his thesis [58]. We will use the term strict process machine for Levin's definition.

Definition 2.1.7. A *strict process machine* is a partial computable function $M : 2^{<\omega} \to 2^{<\omega}$ such that if $\tau \in \text{dom}(M)$ and $\tau' \preceq \tau$, then $\tau' \in \text{dom}(M)$ and $M(\tau') \preceq M(\tau)$.

Both of these definitions of process machines have merit. Schnorr's definition corresponds to a homomorphism of the domain of M. Levin's definition has the following very natural model. This model is almost identical to one described in the first paper on algorithmic randomness by Solomonoff [84]. Take a three-tape Turing machine M with a read-only one-way input tape, a one-way write-once output tape, and a work tape. The first square of the input tape is blank and the input head starts on that square. Let the machine run. If at any stage M wants to move the input head of the tape, first we define $M(\tau) = \sigma$, where τ is the input string read so far and σ is the current output on the output tape.

Schnorr established the existence of a universal process machine, then used this to introduce process complexity.

Definition 2.1.8. The *process complexity* is $K_{M_D}(\sigma) = C^U(\sigma)$ where U is a universal process machine.

The notation of K_{M_D} to denote process complexity follows Downey and Hirschfeldt [30].

Process machines have not been extensively used in the study of randomness. One of the main reasons for this is that they are combinatorially more difficult than prefix-free machines. Strict process machines are often simpler to use because they must keep their domain closed downwards under $\preceq$. In Theorem 5.1.8 we will see how this can be used to prove that there exists a universal strict process machine such that the set of strings random with respect to this machine is not truth-table complete. This proof does not generalise to process machines. In Theorem 6.2.4 we will establish a technique for building strict process machines that is related to the KC theorem.

Hence we will consider what happens if we use strict process machines to define a variant of process complexity. This is a new definition because Levin did not use strict process machines to define a notion of complexity in the same way as Schnorr.

Definition 2.1.9. The *strict process complexity* is $K_{M_S}(\sigma) = C^U(\sigma)$ where U is a universal strict process machine.

The final type of machine which we will examine is the monotone machine. Monotone machines fit into the picture differently. They are not a subclass of Turing machines. Monotone machines can be thought of as process machines that are allowed to describe both finite and infinite binary strings. The idea is that if 0^e1 is the index of a machine that computes an infinite sequence, then 0^e1 is a description of all initial segments of that sequence. It can be argued that monotone machines are more suited for characterising the complexity of real numbers as non-computable reals can be considered as limits of computable reals rather than limits of strings [11]. Monotone machines and the associated complexities were first introduced by Levin [55].

Definition 2.1.10. A *monotone machine* L is a computably enumerable set of pairs of finite binary strings $\langle \tau, \sigma \rangle$ such that if $\langle \tau_1, \sigma_1 \rangle, \langle \tau_2, \sigma_2 \rangle \in L$ and $\tau_1 \preceq \tau_2$, then $\sigma_1 \approx \sigma_2$.

For example, given a computable sequence X, a monotone machine L could be created by enumerating $\langle \tau, X \upharpoonright n \rangle$ into L at stage n. In this case, τ is a finite description of X.

For monotone machines, we say that τ is a description of σ if $\langle \tau, \sigma' \rangle$ is an element of the monotone machine for some σ' extending σ. With this small difference in mind, we define the monotonic descriptive complexity of a string to be the length of its shortest description.

Definition 2.1.11. Let L be a monotone machine. The *monotone complexity* of a

binary string σ with respect to L is:

$$K_m^L(\sigma) = \min\{|\tau| : \exists \sigma' \in 2^{<\omega} \text{ such that } \langle \tau, \sigma' \rangle \in L \text{ and } \sigma \preceq \sigma'\}.$$

We call a monotone machine U optimal if for all monotone machines L there is a constant d such that $K_m^U(\sigma) \leq K_m^L(\sigma) + d$. If L is a monotone machine, we take an enumeration of L and define L_t to be the result of enumerating L for t steps. Again we can take an enumeration $L^0, L^1, \ldots$ of all monotone machines and define a universal monotone machine U as follows, we add $\langle 0^e 1\tau, \sigma \rangle$ to U_t if and only if $\langle \tau, \sigma \rangle \in L_t^e$ for some $e \leq t$.

Definition 2.1.12. The *monotone complexity* is $K_m(\sigma) = K_m^U(\sigma)$ where U is a universal monotone machine.

2.2 Algorithmic probability

We will now examine a different approach to defining the Kolmogorov complexity of a finite string. Instead of considering the shortest description of a universal machine, we will look at the probability that a universal machine outputs a string on some random input. We will start with prefix-free machines.

Definition 2.2.1. Let P be a prefix-free machine. The *discrete algorithmic probability with respect to* P is $m_P^D(\sigma) = \mu\{X \in 2^\omega : (\exists \tau \prec X) P(\tau) = \sigma\}$.

If we regard $2^{<\omega}$ as a discrete space, then the discrete algorithmic probability m_P^D is a finite measure on this space.

Definition 2.2.2. A *c.e. discrete semimeasure* on $2^{<\omega}$ is a function $m^D : 2^{<\omega} \to \mathbb{R}^{\geq 0}$ such that:

(i). $\sum_{\sigma \in 2^{<\omega}} m^D(\sigma) \leq 1$.

(ii). $m^D(\sigma)$ is a uniformly left-c.e. real.

For any prefix-free machine P, m_P^D is a c.e. discrete semimeasure. In fact, this is the only way to create a c.e. discrete semimeasure. Given any c.e. discrete semimeasure m^D, it is not too difficult to construct a prefix-free machine P, such that $m^D = m_P^D$.

The relationship between c.e. discrete semimeasures and prefix-free machines does not end here. The coding theorem tells us that if U is a universal prefix-free machine, then K and $-\log m_U^D$ agree within an additive constant. Thus prefix-free complexity and discrete semimeasures are really different ways of looking at the same thing.

Theorem 2.2.3 (The Coding Theorem, Solomonoff [84], Levin [58, 56], Chaitin [16]). $K(\sigma) = -\log(m^D_U(\sigma)) + O(1)$.

For monotone machines, the relationship between algorithmic probability and descriptive complexity is very different.

Definition 2.2.4. A *c.e. continuous semimeasure* is a uniformly left-c.e. function $m : 2^{<\omega} \to \mathbb{R}^{\geq 0}$ such that:

(i). $m(\lambda) \leq 1$.

(ii). For all $\tau \in 2^{<\omega}$, $m(\tau) \geq m(\tau 0) + m(\tau 1)$.

Observe the similarity between the second condition above and the requirement (1.2.1) on a function to be a premeasure. We noted that any c.e. discrete semimeasure could be described by a prefix-free machine. The following theorem, due to Levin, shows that monotone machines play an analogous role for c.e. continuous semimeasures.

Theorem 2.2.5 (Levin [58, 94]). *(i). If L is a monotone machine, then the function $M_L : 2^{<\omega} \to \mathbb{R}^{\geq 0}$ defined by $M_L(\sigma) = \mu[\{\tau : (\exists \sigma' \succeq \sigma)(\langle \tau, \sigma' \rangle \in L)\}]$ is a c.e. continuous semimeasure.*

(ii). If m is a c.e. continuous semimeasure, then there exists a monotone machine L such that $M_L = m$.

(iii). If U is a universal monotone machine, then M_U multiplicatively majorizes all c.e. continuous semimeasures.

A full proof of this theorem is given in Downey and Hirschfeldt [30]. If U is a universal monotone machine, we call $M_U(\sigma)$ the *monotonic algorithmic probability*. Now if the coding theorem could be adapted to monotone machines and c.e. continuous semimeasures as well, then we would have that $K_m(\sigma) = -\log M_U(\sigma) + O(1)$. Gács showed this is false [36]. Hence there are two types of complexity that arise from monotone machines. It is easier to consider $-\log M_U$ so we make the following definition, again due to Levin [55].

Definition 2.2.6. The *a priori entropy* is $KM(\sigma) = -\log M_U(\sigma)$ where U is a universal monotone machine.

The term a priori entropy is commonly used for this complexity [30, 87]. This is because the word entropy is often used for a universal complexity function and monotonic algorithmic probability is also known in the literature as

a priori probability. One example of where the complexity KM has been used is Reimann's proof of Frostman's Lemma [72].

Both process machines and monotone machines maintain the initial segment relation on $2^{<\omega}$. Because of this they both induce continuous functions. For example, if P is a process machine then let $\mathcal{A} = \{X \in 2^\omega : (\forall c)(\exists n)\,|P(X \upharpoonright n)| > c\}$. The function $f : \mathcal{A} \to 2^\omega$ defined by $f(X) = \bigcup_{n\in\omega} P(X \upharpoonright n)$ is a continuous mapping. Hence we will refer to K_{M_S}, K_{M_D}, K_m and KM as *continuous Kolmogorov complexities*.

2.3 Relationship between complexities

Table 2.1 lists the varieties of Kolmogorov complexity that we have defined. It is natural to ask how they relate to one another. Both Levin and Solovay established a number of theorems relating plain and prefix-free complexity though as these require additional definitions we will not present these result here [30, 58]. The main question we are interested in is, given two complexities Q and P, is it true that $Q(\sigma) \leq P(\sigma) + O(1)$ (a situation we will describe by saying P additively majorizes Q)?

Complexity	Notation
Plain	C
Prefix-free	K
Process	K_{M_D}
Strict process	K_{M_S}
Monotone	K_m
A priori entropy	KM

Table 2.1: Varieties of Kolmogorov complexity

Some complexities obviously additively majorize others. First any strict process machine is a process machine. Secondly any process machine is both a Turing machine and a monotone machine. Finally, a prefix-free machine can be considered as a strict process machine. This is done as follows. Suppose $M : 2^{<\omega} \to 2^{<\omega}$ is a prefix-free machine. We define a strict process machine M' as follows:

$$M'(\sigma) = \begin{cases} M(\sigma) & \text{if } \sigma \in \text{dom}(M) \\ \lambda & \text{if there exists } \sigma' \succ \sigma \text{ and } \sigma' \in \text{dom}(M) \\ \text{undefined} & \text{otherwise.} \end{cases}$$

It can be shown that if M is partial computable then so is M'. Additionally, with the exception of the empty string, the complexities generated by the two machines agree. These observations show us that:

$$K_m(\sigma) \leq K_{M_D}(\sigma) + O(1) \leq K_{M_S}(\sigma) + O(1) \leq K(\sigma) + O(1),$$

and further that $C(\sigma) \leq K_{M_D}(\sigma) + O(1)$.

For any σ, $KM(\sigma) \leq K_m(\sigma)$ because if $K_m(\sigma) = n$ then there is some τ of length n and $\sigma' \succeq \sigma$ such that $\langle \tau, \sigma' \rangle \in U$ so $M_U(\sigma) \geq \mu[\tau] = 2^{-n}$ thus $KM(\sigma) \leq -\log 2^{-n} = n$. Figure 2.1 shows these relationships between these Kolmogorov complexities. An arrow points from complexity P to complexity Q if P additively majorizes Q.

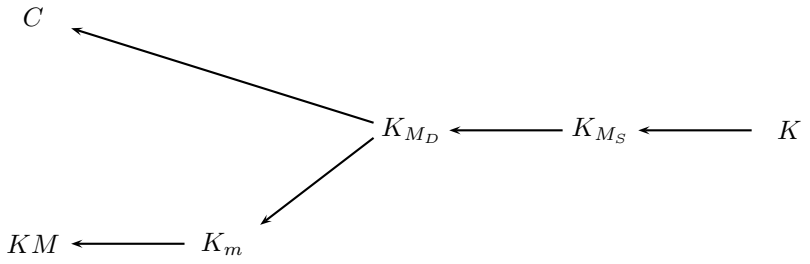

Figure 2.1: Relationship between complexities

It is not immediately apparent that these are the only such relationships between these complexities. However, this is indeed the case. First the following results are not difficult to establish.

Proposition 2.3.1. *(i). K_m does not additively majorize C.*

(ii). C does not additively majorize KM.

(iii). K_{M_S} does not additively majorize K .

Proof. (i). This holds because for some c, for all n, $K_m(1^n) \leq c$.
(ii). This will be shown in Corollary 2.4.2.
(iii). We can build a strict process machine that maps σ to 1^n where $n = \langle \sigma \rangle$. This means that $K_{M_S}(1^n) \leq \log(n) + O(1)$. However, if $K(1^n) \leq \log(n) + O(1)$ this would imply that

$$1 \geq \sum_n 2^{-K(1^n)} \geq \sum_n 2^{-\log(n)-c} = \sum_n 2^{-c} \frac{1}{n},$$

which is impossible as the harmonic series diverges. □

For further details on (i) and (ii) see Uspensky and Shen's paper *Relations Between Varieties of Kolmogorov Complexities* [87]. Uspensky and Shen's paper also discusses Loveland's decision complexity which we will not define here.

If P does not additively majorize Q then we can examine the difference $Q - P$. We call a non-decreasing function h an upper-bound for $Q - P$ if $Q(\sigma) - P(\sigma) \leq h(|\sigma|) + O(1)$. We call a non-decreasing function l a lower-bound for $Q - P$ if there are infinitely many $\sigma \in 2^{<\omega}$ such that $Q(\sigma) - P(\sigma) > l(|\sigma|)$. Uspensky and Shen's paper also contains a great deal of information about the upper and lower bounds between: C, K, K_m, KM and decision complexity [87].

In Chapter 3 we will prove that the complexities K_{M_D} and K_{M_S} are different, i.e. K_{M_D} does not additively majorize K_{M_S}. We will provide a lower bound by showing that for any $a \in \mathbb{R}$ with $0 < a < 1$, there are infinitely many σ such that $K_{M_S}(\sigma) - K_{M_D}(\sigma) > a \log\log|\sigma|$. Gács proved that KM does not additively majorize K_m [36]. Gács's proof established that some version of the inverse Ackermann function is a lower bound for $K_m - KM$. In Chapter 4 we will extend this result to show that for any $a \in \mathbb{R}$ with $0 < a < 1$, there are infinitely many σ such that $K_m(\sigma) - KM(\sigma) > a \log\log|\sigma|$. While this lower bound is the same as that for $K_{M_S}(\sigma) - K_{M_D}(\sigma)$ the proof is significantly more difficult.

2.4 Initial segment complexity

Theorem 2.1.2 states that any $X \in 2^\omega$ has initial segments whose plain Kolmogorov complexity is arbitrarily less than their length. This result is not true for the other varieties of Kolmogorov complexity we have defined. In fact, all these other varieties can be used to characterise the Martin-Löf random sequences. The first such characterisations were established by Schnorr and Levin. The techniques they used generalise to any complexity other than C and so we attribute the following theorem to them.

Theorem 2.4.1 (Levin, Schnorr [55, 80])**.** *If Q is any of the complexities: K, K_{M_S}, K_{M_D}, K_m or KM then for any $X \in 2^\omega$, X is Martin-Löf random if and only if $Q(X \upharpoonright n) \geq n - O(1)$.*

We note that Miller and Yu have established that it is possible to characterise Martin-Löf randomness in terms of plain Kolmogorov complexity, though presenting this result would take us too far afield [65]. We can use these characterisations to show that C does not additively majorize KM.

Corollary 2.4.2. *For all $c \in \omega$ there exists $\sigma \in 2^{<\omega}$ such that $KM(\sigma) \geq C(\sigma) + c$.*

Proof. Take $X \in MLR$. There is some d such that for all n, $KM(X \upharpoonright n) \geq n - d$. By Theorem 2.1.2, for any c there is some n_c such that $C(X \upharpoonright n_c) \leq n_c - c - d$. Hence $KM(X \upharpoonright n_c) \geq n_c - d \geq C(X \upharpoonright n_c) + c$. □

Any complexity measure can be used to place an ordering on Cantor space. Let Q be any Kolmogorov complexity and take $A, B \in 2^\omega$. We say that $A \leq_Q B$ if $Q(A \upharpoonright n) \leq Q(B \upharpoonright n) + O(1)$. Note that the ordering $\leq_Q$ is not a reducibility: if $A \leq_Q B$, then it is not necessarily true that A can be computed from B. In Chapter 9 we will examine an ordering of 2^ω in terms of randomness which is also a reducibility.

It is not difficult to show that if A is a computable sequence, then $A \leq_Q B$ for any sequence B. Hence the computable sequences can be regarded as having minimal information content. A natural question to ask is whether there are any non-computable sequences with this same property. A sequence is $\leq_Q$ below all sequences if and only if it is $\leq_Q$ below any fixed computable sequence. Hence we will fix 1^ω as the computable sequence we use for comparison and we will consider the set $\{A \in 2^\omega : A \leq_Q 1^\omega\}$. For any complexity measure Q, we will write $Q(n)$ to refer to $Q(1^\omega \upharpoonright n)$. Hence asking if $A \leq_Q 1^\omega$ is the same as asking if $Q(A \upharpoonright n) \leq Q(n) + O(1)$. Chaitin proved that the sequences with this property, when Q is taken to be C, are precisely the computable sequences.

Theorem 2.4.3 (Chaitin [17])**.** *Given any $X \in 2^\omega$, the following are equivalent:*

(i). $C(X \upharpoonright n) \leq C(n) + O(1)$.

(ii). $C(X \upharpoonright n) \leq \log n + O(1)$.

(iii). X *is computable.*

We can give a simple corollary to this theorem.

Corollary 2.4.4. *Given any $X \in 2^\omega$, the following are equivalent:*

(i). $K_{M_D}(X \upharpoonright n) \leq K_{M_D}(n) + O(1)$.

(ii). $K_{M_S}(X \upharpoonright n) \leq K_{M_S}(n) + O(1)$.

(iii). X *is computable.*

Proof. We observed in the proof of Proposition 2.3.1 that $K_{M_S}(n) \leq \log n + O(1)$. If for some $X \in 2^\omega$, we have that $K_{M_S}(X \upharpoonright n) \leq K_{M_S}(n) + O(1)$, then

$$C(X \upharpoonright n) \leq K_{M_S}(X \upharpoonright n) + O(1) \leq K_{M_S}(n) + O(1) \leq \log n + O(1)$$

so X is computable. The same argument holds for process complexity. □

For the complexities K_m and KM it is not difficult to see that $X \in 2^\omega$ is computable if and only if $KM(X \upharpoonright n) = O(1)$ if and only if $K_m(X \upharpoonright n) = O(1)$. Thus all the complexities examined have simple characterisations of the computable sequences except for prefix-free complexity. Surprisingly there are non-computable sequences A such that $A \leq_K 1^\omega$. This was first established in unpublished work of Solovay (see Downey and Hirschfeldt [30]). Such sequences are known as K-trivial. The class of all K-trivials has some remarkable properties. As this class is not central to this thesis, we will simply mention one key characterisation of the K-trivials. A sequence A is called *low for Martin-Löf randomness* if the set of Martin-Löf random sequences relative to A is just the set of Martin-Löf random sequences. Intuitively a sequence is low for Martin-Löf randomness if it has no power to compress information. Nies showed that the class of K-trivials coincides with the class of sequences low for Martin-Löf randomness [69].

Chapter 3

Process Complexity

(The results of this chapter appeared in the paper: On Process Complexity, Chicago Journal of Theoretical Computer Science, Vol 2010, Issue 4, June 2010. An initial version of this paper was presented at CATS 2009: Fifteenth Computing: The Australasian Theory Symposium.)

In this chapter we will prove two theorems about process complexities. In Section 3.1 we will show that process complexity and strict process complexity are different. We will also establish a lower bound for this difference. In Section 3.2 we will prove that neither process complexity nor strict process complexity is subadditive.

3.1 Strict process complexity and process complexity

In this section we will show that strict process complexity and process complexity are in fact different. As the universal strict process machine is a process machine, there is some constant d such that for all $\sigma \in 2^{<\omega}$:

$$K_{M_D}(\sigma) \leq K_{M_S}(\sigma) + d.$$

We want to show that this inequality cannot be reversed and so strict process complexity and process complexity do not agree within an additive constant. To show this, we will make use of the fact that the universal strict process machine has a computable approximation. Let U be a universal strict process machine. Because U is a strict process machine, we can take our approximation to have the property that if $U_s(\tau) \downarrow = \sigma$ and $\tau' \prec \tau$ then there is some $\sigma' \preceq \sigma$ such that $U_s(\tau') \downarrow = \sigma'$.

In Lemma 3.1.2, we will show that for any $i \in \omega$, we can construct a process machine f_i such that there exists a string σ_i with $C^{f_i}(\sigma_i) + i < K_{M_S}(\sigma_i)$. Once we have done this, it will not be too difficult to combine these machines

to prove that strict process complexity and process complexity do not agree within an additive constant.

To understand the ideas behind the proof of Lemma 3.1.2, let us take the case $i = 1$ as an example. We will construct a process machine f_1. When we construct this machine, we are able to first define f_1 for all strings of length 3. Then at a later stage, we have the option of defining f_1 for strings of length 2 and even later for strings of length 1. This option is not available to the universal strict process machine U. Once a string τ is added to the domain of U, all initial segments of τ must be added as well. Our definition of f_1 starts as follows. First let $\tau = abc$ be any binary string of length 3, i.e. a, b, and c are the first, second and third bits of τ respectively. We define f_1 by $f_1(abc) = a^8b^{16}c$ e.g. $f_1(010) = 000000001111111111111110$. We can consider this as 'stretching' all but the last bit of the string abc.

Now we wait until some stage s_1, when for all $\tau \in \{0,1\}^3$, $C^{U_{s_1}}(f_1(\tau)) \leq 4$. If this never happens then we have finished because for some $\tau \in \{0,1\}^3$,

$$\begin{aligned} K_{M_S}(f_1(\tau)) &= \lim_{s\to\infty} C^{U_s}(f_1(\tau)) \\ &> 4 \\ &= |\tau| + 1 \\ &= C^{f_1}(f_1(\tau)) + 1. \end{aligned}$$

So assume such a stage s_1 occurs. For all $\tau \in \{0,1\}^3$, let ρ_τ be some string in the domain of U_{s_1} such that $|\rho_\tau| \leq 4$ and $U_{s_1}(\rho_\tau) = f_1(\tau)$. Let A_1 be the set of all such ρ_τ. Note that A_1 must be a prefix-free set and $|A_1| = 2^3$. If A_1 is not prefix-free, then there is some $\tau, \upsilon \in \{0,1\}^3$ with: $\tau \neq \upsilon$; $\rho_\tau, \rho_\upsilon \in A_1$; and $\rho_\tau \preceq \rho_\upsilon$. But this would mean that $f_1(\tau) = U_{s_1}(\rho_\tau) \preceq U_{s_1}(\rho_\upsilon) = f_1(\upsilon)$ which contradicts our definition of f_1. It follows that $\mu[A_1] \geq |A_1|2^{-4} = \frac{1}{2}$.

We will now define f_1 for all binary strings of length 2. Given any two bit binary string ab, there must be some string a^8b^k with $1 \leq k \leq 16$ such that $C^{U_{s_1}}(a^8b^k) > 3$. This is true because there are at most 15 strings whose complexity is less than or equal to 3 (as there are only 15 such descriptions). Now we define $f_1(ab) = a^8b^k$. Consider for a moment how the universal strict process machine can respond to this. Because U is a strict process machine, if τ is any initial segment of a string in A_1, then $U(\tau)[s_1] \downarrow$. So if $|\tau| \leq 3$, then $U(\tau) \neq f_1(ab)$ for any $a, b \in \{0,1\}$. This means that in order to reduce the complexity of $f_1(ab)$ to 3 or less, U needs to find a short description that is not an initial segment of an element of A_1.

Again we wait until some stage s_2, when for all $\tau \in \{0,1\}^2$, $C^{U_{s_2}}(f_1(\tau)) \leq 3$. If this never happens then again our objective is achieved. If this stage does

occur then for all $\tau \in \{0,1\}^2$, let ρ_τ be some string in the domain of U_{s_2} such that $|\rho_\tau| \leq 3$ and $U_{s_2}(\rho_\tau) = f_1(\tau)$. Let A_2 be the set of all such ρ_τ. Again $\mu[A_2] \geq |A_2|2^{-3} = \frac{1}{2}$. We want to show that $[A_1] \cap [A_2] = \emptyset$. Take any $\rho_1 \in A_1$ and $\rho_2 \in A_2$. First $|U(\rho_1)| > |U(\rho_2)|$ so we know that $\rho_1 \not\preceq \rho_2$. Now let us show that $U(\rho_2)[s_1] \uparrow$. If $U(\rho_2)[s_1] \downarrow$, then $C^{U_{s_1}}(U(\rho_2)) \leq |\rho_2| \leq 3$. So by our construction of f_1, $f_1(ab) \neq U(\rho_2)$ for any $a, b \in \{0,1\}$. This is a contradiction and so $U(\rho_2)[s_1] \uparrow$. As $U(\rho_1)[s_1] \downarrow$ so it must be that $\rho_2 \not\succ \rho_1$ because U is a strict process machine. Hence ρ_1 and ρ_2 are incomparable and so $\mu[\text{dom}(U_{s_2})] \geq \mu[A_1] + \mu[A_2] = 1$.

Finally we define $f_1(0) = 0^k$ for some $1 \leq k \leq 8$ such that $C^{U_{s_2}}(0^k) > 2$ (there must be some k as there are only 7 possible descriptions of length less than or equal to 2). Consider any $\upsilon \in 2^{<\omega}$. As $\mu[A_1 \cup A_2] = 1$, either υ is an initial segment, or an extension, of some element $\rho \in A_1 \cup A_2$. If $\upsilon > \rho$ and $U(\upsilon) \downarrow$ then $U(\upsilon) \succeq U(\rho)$ and so $U(\upsilon) \neq f_1(0)$. If $\upsilon \preceq \rho$, then $U(\upsilon)[s_2] \downarrow$ so if $U(\upsilon) = f_1(0)$ then it must be that $|\upsilon| > 2$ (as we chose $f_1(0)$ so that $C^{U_{s_2}}(f_1(0)) > 2$). Hence:

$$\begin{aligned} C^{f_1}(f_1(0)) + 1 &= 1 + 1 \\ &< C^U(f_1(0)) \\ &= K_{M_S}(f_1(0)). \end{aligned}$$

The main idea is that if the universal strict process machine attempts to respond each time strings are added to the domain of f_1, then it will run out of measure. The key point is that during the construction of f_1, we can reuse measure by using initial segments of strings already in the domain of f_1. On the other hand, U needs to add new measure into its domain at each response. To adapt this argument to hold for any i, let us start with a lemma giving a lower bound on the measure of the domain of a strict process machine.

Lemma 3.1.1. *If $\{(\tau_1, \sigma_1), \ldots, (\tau_n, \sigma_n)\}$ is a set of ordered pairs such that for all $i, j \in \omega$, $1 \leq i, j \leq n$, $U(\tau_i) = \sigma_i$; and if $i \neq j$ then:*

(i). $\sigma_i \neq \sigma_j$; and

(ii). $\sigma_i \succ \sigma_j$ implies that there exists an s such that $U_s(\tau_i) \downarrow$ and $U_s(\tau_j) \uparrow$;

then $\mu[\text{dom}(U)] \geq \sum_{i=1}^n 2^{-|\tau_i|}$.

Proof. We will show that the τ_i form a prefix-free set. Choose any $i \neq j$. If $\sigma_i \mid \sigma_j$ then as U is a process machine, $\tau_i \mid \tau_j$. If $\sigma_i \succ \sigma_j$ then there exists an s such that $U_s(\tau_i) \downarrow$ and $U_s(\tau_j) \uparrow$, so as U is a strict process machine τ_j is not

an initial segment of τ_i. Further τ_j cannot extend τ_i as this would imply that $\sigma_j \succeq \sigma_i$. Hence $\tau_i \mid \tau_j$. Similarly if $\sigma_j \succ \sigma_i$ then $\tau_i \mid \tau_j$ as well. □

We will formalise the notion of stretching a string as follows. If $g : \omega \to \omega$, then $\hat{g} : 2^{<\omega} \to 2^{<\omega}$ is defined by:

$$\hat{g}(\tau) = \tau(0)^{g(0)}\tau(1)^{g(1)} \dots \tau(|\tau| - 1)^{g(|\tau|-1)}.$$

For example if $g(x) = x + 1$, then $\hat{g}(0101) = 0^1 1^2 0^3 1^4 = 0110001111$.

Lemma 3.1.2. *For all $i \in \omega$, there exists a process machine f_i and a string σ_i such that:*

$$C^{f_i}(\sigma_i) + i < K_{M_S}(\sigma_i).$$

Proof. Again we take U to be the universal strict process machine. We looked at the case f_1 as an example. In this case we started by defining f_1 for strings of length 3, and then for strings of progressively shorter lengths. For the general case, we will start by defining f_i for strings of length 2^i+1. If necessary we will define f_i for strings of length 2^i, $2^i - 1$, $2^i - 2$ and so on. Our prompt to extend the domain of f_i is if at some stage s, the universal strict process machine has made $C^{U_s}(\sigma) \leq C^{f_i}(\sigma) + i$ for all elements σ in the domain of f_i at stage s.

We define $g_i : \omega \to \omega$ by $g_i(n) = 2^{n+i+2}$. The construction of f_i proceeds as follows. At stage 0, first set $l_0 = 2^i + 1$. Then for all $\tau \in \{0,1\}^{l_0}$ set $f_i(\tau) = \hat{g}_i(\tau \upharpoonright (|\tau| - 1))\tau(|\tau| - 1)$ i.e. we use g_i to stretch all but the last bit of τ.

At stage $s+1$, if there exists some $\tau \in \{0,1\}^{l_s}$ such that $C^{U_s}(f_i(\tau)) > |\tau| + i$, then set $l_{s+1} = l_s$ and go to the next stage.

Otherwise, we know that for all $\tau \in \{0,1\}^{l_s}$, $C^{U_s}(f_i(\tau)) \leq |\tau| + i$. Now we will extend the domain of f_i. We set $l_{s+1} = l_s - 1$. For all $\tau \in \{0,1\}^{l_{s+1}}$ and for all $k \in \omega, 1 \leq k \leq g_i(|\tau| - 1)$ let $\sigma_{\tau,k} = \hat{g}_i(\tau \upharpoonright (|\tau| - 1))\tau(|\tau| - 1)^k$. As there are $2^{|\tau|+i+1}$ possible values of k, it follows that for any τ, there must be some k such that: $C^{U_s}(\sigma_{\tau,k}) > |\tau| + i$ because there are only $2^{|\tau|+i+1} - 1$ descriptions of length less than or equal to $|\tau| + i$. For all $\tau \in \{0,1\}^{l_{s+1}}$, let $\sigma_\tau = \sigma_{\tau,k}$ for such a k and set $f_i(\tau) = \sigma_\tau$.

To verify the construction we will first show that for all s, $l_s > 0$. We will prove this by showing that the alternative implies that U runs out of measure. If $l_s = 0$ for some s, then for all $\tau \in 2^{<\omega}$ such that $1 \leq |\tau| \leq 2^i + 1$, $K_{M_S}(f_i(\tau)) \leq |\tau| + i$ because this is the condition to extend the domain of f_i. So for all such τ we can choose a string ρ_τ such that (i) $|\rho_\tau| \leq |\tau| + i$, (ii) $U(\rho_\tau) = f_i(\tau)$ and such that no other string with properties (i) and (ii) halts before $U(\rho_\tau)$ halts.

Now take the set $A = \{(\rho_\tau, f_i(\tau)) : \tau \in 2^{<\omega}, 1 \leq |\tau| \leq 2^i + 1\}$. Consider any $\tau_1, \tau_2 \in 2^{<\omega}$, $1 \leq |\tau_1|, |\tau_2| \leq 2^i + 1$, and $\tau_1 \neq \tau_2$. First $f_i(\tau_1) \neq f_i(\tau_2)$ as f_i is

injective. If $f_i(\tau_1) \prec f_i(\tau_2)$ then by construction this implies that $\tau_1 \prec \tau_2$. Now f_i is defined for τ_1 after f_i is defined for τ_2. Further if $f_i(\tau_1)$ is first defined at stage $t+1$, it must be that:

(1) $C^{U_t}(f_i(\tau_2)) \leq |\tau_2| + i$.

(2) $C^{U_t}(f_i(\tau_1)) > |\tau_1| + i$.

(1) follows as this is required to extend the domain of f_i to strings of length $< |\tau_2|$. (2) follows by our choice of $f_i(\tau_1)$. Now by (i) and (ii), (1) implies that $U_t(\rho_{\tau_2})\downarrow$ and (2) implies that $U_t(\rho_{\tau_1})\uparrow$. The set A therefore meets the conditions of Lemma 3.1.1 and this implies that:

$$\begin{aligned}\mu[\mathrm{dom}(U)] &\geq \sum_{\tau\in 2^{<\omega}, 1\leq|\tau|\leq 2^i+1} 2^{-|\rho_\tau|} \\ &\geq \sum_{\tau\in 2^{<\omega}, 1\leq|\tau|\leq 2^i+1} 2^{-|\tau|-i} \\ &= 2^{-i}(2^i+1) > 1.\end{aligned}$$

A contradiction and so for all s, $l_s > 0$.

Let $n = \min\{l_s : s \in \omega\}$. It follows that for some $\tau \in \{0,1\}^n$, for all s, $C^{U_s}(f_i(\tau)) > |\tau|+i$ and hence $K_{M_S}(f_i(\tau)) > |\tau|+i$. Thus we can take $\sigma_i = f_i(\tau)$, and we have that $C^{f_i}(\sigma_i) + i = |\tau| + i < K_{M_S}(\sigma_i)$.

Finally, we will show that f_i is a process machine. Take any $\tau_1, \tau_2 \in \mathrm{dom}(f_i)$ such that $\tau_1 \prec \tau_2$. We have that: $f_i(\tau_1) \preceq \hat{g}_i(\tau_1) \preceq \hat{g}_i(\tau_2 \upharpoonright (|\tau_2|-1)) \preceq f_i(\tau_2)$. □

Note that Lemma 3.1.2 is uniform.

Theorem 3.1.3. *K_{M_D} and K_{M_S} do not agree within an additive constant.*

Proof. Define a process machine f by $f(0^i1\tau) = f_{2i}(\tau)$. Let c be the length of the index of f in the universal process machine. Now given any d, take $i = 2(c+d+1)$. By Lemma 3.1.2, there exists some σ_i such that $C^{f_i}(\sigma_i) + i < K_{M_S}(\sigma_i)$, so:

$$\begin{aligned}K_{M_D}(\sigma_i) + d &\leq C^f(\sigma_i) + d + c \\ &\leq C^{f_i}(\sigma_i) + d + c + \frac{i}{2} + 1 \\ &= C^{f_i}(\sigma_i) + i \\ &< K_{M_S}(\sigma_i).\end{aligned}$$

□

In fact we can go further and prove that this constant can be replaced with a function of order $\log\log(n)$ where n is string length. First note that we can determine an upper bound on the length of the σ_i from Lemma 3.1.2. Because $\sigma_i = f_i(\tau)$ for some τ with $|\tau| \leq 2^i + 1$, and as we stretch all but the last bit of τ we know that:

$$\begin{aligned}
|\sigma_i| &\leq 1 + \sum_{n=0}^{2^i-1} g_i(n) \\
&= 1 + \sum_{n=0}^{2^i-1} 2^{n+i+2} \\
&= 1 + 2^{i+2}(2^{2^i} - 1) \\
&< 2^{2^i+i+2}.
\end{aligned}$$

It will be easier to express this as $\log|\sigma_i| < 2^i + i + 2$.

Theorem 3.1.4. *Given any $a \in \mathbb{R}$, $0 < a < 1$, then there exist infinitely many σ such that:*

$$K_{M_S}(\sigma) - K_{M_D}(\sigma) > a \log\log|\sigma|.$$

Proof. Given any such a choose $k \in \omega$ such that $a < 1 - \frac{1}{k}$. Now define a process $f : 2^{<\omega} \to 2^{<\omega}$ by $f(0^i 1\tau) = f_{2ki}(\tau)$. Let c be the length of the index of f in the universal process machine. Choose any $d \in \omega$ such that $2k-1$ divides $c + d + 1$. Now let $i = \frac{2k(c+d+1)}{2k-1}$. Hence i is a positive integer, and $i = c + d + 1 + \frac{i}{2k}$. This implies that $\frac{i}{2k}$ is a positive integer too. Let $\sigma_d = \sigma_i$ from Lemma 3.1.2. We know that:

$$\begin{aligned}
K_{M_D}(\sigma_d) + d &\leq C^f(\sigma_d) + c + d \\
&= C^{f_i}(\sigma_d) + c + d + 1 + \frac{i}{2k} \\
&= C^{f_i}(\sigma_d) + i \\
&< K_{M_S}(\sigma_d).
\end{aligned}$$

Also we know that:

$$\begin{aligned}
\log|\sigma_d| &< 2^{\frac{2k(c+d+1)}{2k-1}} + \frac{2k(c+d+1)}{2k-1} + 2 \\
&= c_1 + \frac{2k}{2k-1}d + c_2 \cdot 2^{\frac{2k}{2k-1}d}
\end{aligned}$$

where c_1 and c_2 are constant. Let $j = \frac{k}{(2k-1)(2k-2)}$. As there are infinitely many d that we can choose, we can consider those d such that

$$2^{jd} \geq \max(c_1 + \frac{2k}{2k-1}d, c_2, 2).$$

So we have that:

$$\begin{aligned}\log|\sigma_d| &< 2^{jd} + 2^{jd}2^{\frac{2k}{2k-1}d} \\ &= 2^{jd}(1 + 2^{\frac{2k}{2k-1}d}) \\ &\leq 2^{jd}(2^{\frac{2k}{2k-1}d+jd}) \\ &= 2^{\frac{2k}{2k-1}d+2jd} \\ &= 2^{\frac{k}{k-1}d}.\end{aligned}$$

The last step follows because:

$$\begin{aligned}\frac{2k}{2k-1} + 2j &= \frac{2k}{2k-1} + \frac{2k}{(2k-1)(2k-2)} \\ &= \frac{4k^2-2k}{(2k-1)(2k-2)} \\ &= \frac{k}{k-1}.\end{aligned}$$

So $\log\log|\sigma_d| < \frac{k}{k-1}d$ and hence we have that:

$$a\log\log|\sigma_d| < \frac{k-1}{k}\log\log|\sigma_d| < d.$$

For these such d, we have that $K_{M_D}(\sigma_d) + d < K_{M_S}(\sigma_d)$. This implies that $K_{M_D}(\sigma_d)+a\log\log|\sigma_d| < K_{M_S}(\sigma_d)$ and rearranging gives $K_{M_S}(\sigma_d)-K_{M_D}(\sigma_d) > a\log\log|\sigma_d|$. There are infinitely many d that meet the conditions we require so the result follows. □

Theorem 3.1.4 gives a lower bound on $K_{M_S} - K_{M_D}$. A basic upper bound on the difference between these complexities is that $K_{M_S}(\sigma) - K_{M_D}(\sigma) \leq 2\log|\sigma| + O(1)$. This holds because $K - K_M$ is bounded above by the same amount and both $K_M(\sigma) \leq K_{M_D}(\sigma)+O(1)$ and $K_{M_S}(\sigma) \leq K(\sigma)+O(1)$ [60, 87]. This leaves an open question with respect to the difference between these two complexities – which of these two bounds can be improved?

3.2 Process complexity is not subadditive

A fundamental question about any complexity measure is whether or not it is subadditive. A complexity measure is subadditive if there is some constant d such that for all strings σ, τ the complexity of $\sigma\tau$ is less than or equal to the sum of the complexity of σ plus the complexity of τ plus d. Prefix-free complexity is subadditive. This is proved by observing that there is a prefix-free machine M that on input ρ tries to find two strings ρ_0, ρ_1 such that $\rho = \rho_0\rho_1$ and ρ_0, ρ_1 are both in the domain of the universal prefix-free machine U. Then

we let $M(\rho) = U(\rho_0)U(\rho_1)$. One reason this proof works is that if such a ρ_0, ρ_1 are found for ρ, then these must be the unique strings with this property.

Martin-Löf's proof of Theorem 2.1.2 can be adapted to show that plain Kolmogorov complexity is not subadditive. That is to say, for any d there exists strings σ, τ such that $C(\sigma\tau) > C(\sigma) + C(\tau) + d$. We can take a random finite string υ that has an initial segment σ with $C(\sigma) < |\sigma| - d$. Now if τ is chosen so that $\sigma\tau = \upsilon$ then $C(\sigma\tau) \geq |\sigma| + |\tau| > C(\sigma) + d + C(\tau) - i$ where i is the length of the index of the identity function in U. As i is fixed we can make $d - i$ arbitrarily large. Thus we have that plain Kolmogorov complexity is not subadditive.

We will now establish that both process complexity and strict process complexity are not subadditive. The proof Martin-Löf used for plain complexity cannot be adapted to process complexity. This is because given a Martin-Löf random sequence X, it is true that $K_{M_D}(X \upharpoonright n) \geq n - O(1)$ i.e. there are no arbitrary drops in initial segment complexity. Thus the question as to whether these complexities are subadditive requires new techniques. In particular, we need to use non-random strings. The new techniques used for building and analysing process machines introduced here may well have wider application.

We will present the proof for process complexity but it holds without modification for strict process complexity as well.

Theorem 3.2.1. *Let U be a universal process machine. For any $d \in \omega$, there exist $\sigma, \tau \in 2^{<\omega}$ such that:*

$$K_{M_D}(\sigma\tau) > K_{M_D}(\sigma) + K_{M_D}(\tau) + d.$$

We will prove this theorem by giving short descriptions to a lot of strings. We will argue combinatorially that one of these strings σ, must have an extension $\sigma\tau$ such that the desired property holds. The following lemma expresses a basic combinatorial fact about process machines.

Lemma 3.2.2. *If $A \subseteq 2^{<\omega}$ is a prefix-free set, then:*

$$\sum_{\sigma \in A} 2^{-K_{M_D}(\sigma)} \leq 1.$$

Proof. Consider $B = \{\tau_\sigma : \sigma \in A\}$ where τ_σ is a shortest description of σ with respect to the universal process machine U. If τ_1, τ_2 are distinct elements of B, then $U(\tau_1)$ and $U(\tau_2)$ are incomparable (as they are both in A). Therefore because U is a process machine we have that $\tau_1 \not\preceq \tau_2$. Hence B is a prefix-free set as well and the result follows because:

$$\sum_{\sigma\in A} 2^{-K_{M_D}(\sigma)} = \sum_{\tau\in B} 2^{-|\tau|} = \mu[B] \leq 1.$$

□

Let $g : \omega \to \omega$ be the constant function $g(x) = 2$. Now the function $\hat{g} : 2^{<\omega} \to 2^{<\omega}$ is a process (recall the definition of $\hat{g}$ in Section 3.1). For all i, we can define $A_i = \{\hat{g}(\tau) : \tau \in \{0,1\}^i\}$. So $A_0 = \{\lambda\}$, $A_1 = \{00, 11\}$, $A_2 = \{0000, 0011, 1100, 1111\}$, etc.

Because $\hat{g}$ is a process, there exists a constant c_g, such that for all $\rho \in A_i$, $K_{M_D}(\rho) \leq i + c_g = |\rho| - i + c_g$. The A_i's are our sets of strings with short descriptions.

For all m, i such that $m \geq 2i + 2$, we define

$$B_i^m = \{\sigma \in \{0,1\}^m : \exists \rho \in A_i (\rho 01 \preceq \sigma \text{ or } \rho 10 \preceq \sigma)\}.$$

The B_i^m's are the sets of extensions that we will examine. It is important to note that we have constructed the B_i^m's so that if $i \neq j$ then $B_i^m \cap B_j^m = \emptyset$. We will explain the reason for this using an example. Consider $\sigma = 00110100$. Because σ extends an element of A_2, 0011, we want to place σ in B_2^8 and not in B_1^8 even though σ extends 00 as well. This is because if we break σ into the strings 0011 and 0100 then the difference between the length of the first string 0011, and the length of its $\hat{g}$ description is 2 ($\hat{g}(01) = 0011$). This is larger than the difference between the length of 00 and the length of its $\hat{g}$ description. We need to use the larger difference for the following argument to hold.

Note that $|B_i^m| = |A_i| \cdot 2 \cdot 2^{m-2i-2} = 2^{m-i-1}$. Now we will use the following lemma to find the extension we want.

Lemma 3.2.3. *Given any $e \in \omega$, there exist $m, i \in \omega$, and $\sigma \in B_i^m$ such that $K_{M_D}(\sigma) > |\sigma| - i + e$.*

Proof. Take $m = 2^{e+3}$ and assume that no such i, σ exist. If so, then:

$$\sum_{\sigma\in\{0,1\}^m} 2^{-K_{M_D}(\sigma)} \geq \sum_{i=0}^{\frac{m}{2}-1} \sum_{\sigma\in B_i^m} 2^{-K_{M_D}(\sigma)} \geq \sum_{i=0}^{\frac{m}{2}-1} \sum_{\sigma\in B_i^m} 2^{-|\sigma|+i-e} = \sum_{i=0}^{\frac{m}{2}-1} |B_i^m| 2^{-m+i-e}.$$

But as,

$$\sum_{i=0}^{\frac{m}{2}-1} |B_i^m| 2^{-m+i-e} = \sum_{i=0}^{\frac{m}{2}-1} 2^{m-i-1} 2^{-m+i-e} = \left(\frac{m}{2}\right) 2^{-e-1} > 1,$$

we derive a contradiction by Lemma 3.2.2 as $\{0,1\}^m$ is a prefix-free set. □

Proof of Theorem 3.2.1. Because the identity function is a process, there exists a constant c such that for all $\sigma \in 2^{<\omega}$, $K_{M_D}(\sigma) \leq |\sigma| + c$. Now given any d, choose $e = d + c + c_g$. From the previous lemma, there exists some m, i, υ with $\upsilon \in B_i^m$ such that $K_{M_D}(\upsilon) > |\upsilon| - i + e$. Now set $\sigma = \upsilon \upharpoonright 2i$ so $\sigma \in A_i$ and thus $K_{M_D}(\sigma) \leq |\sigma| - i + c_g$. Choose τ so that $\sigma\tau = \upsilon$. Now $K_{M_D}(\tau) \leq |\tau| + c$. Combining these results gives us that:

$$\begin{aligned} K_{M_D}(\sigma) + K_{M_D}(\tau) + d &\leq |\sigma| - i + c_g + |\tau| + c + d \\ &= |\sigma\tau| - i + e \\ &< K_{M_D}(\sigma\tau). \end{aligned}$$

□

Chapter 4

Difference in Monotone Complexities

The work in this chapter has been accepted for publication in the Transactions of the American Mathematical Society.

4.1 Overview

In this chapter we investigate the relationship between the Kolmogorov complexities K_m and KM. We saw in Chapter 2 that these complexities are both based on monotone machines. K_m is the descriptive complexity and KM is the negative logarithm of the algorithmic probability. The relationship between these two complexities has interesting philosophical implications. The complexity K_m is an application of the principle of Occam's razor: the simplest, or in this case the shortest, description is preferred. Levin observed that KM is a natural candidate for any *a priori* probability used in Bayesian statistics because of its universal properties. Levin conjectured that an analogue of the coding theorem might hold for monotonic complexity i.e. that KM and K_m would agree within an additive constant [55]. Gács showed that Levin's conjecture is false.

Theorem 4.1.1 (Gács)**.** *Given any $d \in \omega$, there exists $\sigma \in 2^{<\omega}$, such that $K_m(\sigma) - KM(\sigma) > d$.*

One is naturally led to investigate the relationship between a given d and the corresponding σ. Now Gács's proof of Theorem 4.1.1 was set in the context of integer strings (as opposed to binary strings). However, out of the construction of the proof Gács developed the following lower bound for binary strings [37].

Theorem 4.1.2 (Gács)**.** *There exists a computable unbounded function A^{-1}, where A^{-1} is some version of the inverse Ackermann function, such that for infinitely many σ, $K_m(\sigma) - KM(\sigma) > A^{-1}(|\sigma|)$.*

The inverse Ackermann function is *extremely* slow growing. If the difference between these two complexities was no more than A^{-1} then the difference might be seen as essentially negligible. The following theorem gives an upper bound on the difference between the two complexities [37, 87]. First we inductively define $\log^k x$ by $\log^1 x = \log x$ and $\log^{n+1} x = \log \log^n x$.

Theorem 4.1.3. *For any $k \in \omega$, $\epsilon \in \mathbb{R}^{>0}$, with $k \geq 1$:*

$$\begin{aligned} K_m(\sigma) - KM(\sigma) &\leq K(|\sigma|) + O(1) \\ &\leq \log|\sigma| + \log\log|\sigma| + \ldots + (1+\epsilon)\log^k|\sigma| + O(1). \end{aligned}$$

The gap between the upper and lower bounds provided by Theorems 4.1.3 and 4.1.2 is enormous. In general, the upper and lower bounds between most Kolmogorov complexities are very close [60, 87]. This theorem is still true if K_m is replaced by K [87]. It would be surprising if this upper bound was tight as it would imply a prefix-free machine was just as good as approximating KM as a monotone machine.

Since Gács's result, over twenty-five years ago, an important open question in algorithmic randomness has been whether either of these bounds on the difference between K_m and KM can be improved. In this chapter we will establish a very strong lower bound by proving the following theorem.

Theorem 4.1.4. *If $c \in \mathbb{R}$, and $c < 1$, then there exist infinitely many $\sigma \in 2^{<\omega}$ such that:*

$$K_m(\sigma) - KM(\sigma) > c \log\log|\sigma|.$$

To prove Theorem 4.1.4 we will develop a proof of Theorem 4.1.1. The improvement in the lower bound follows from the construction used in the proof. While this proof builds on the original work of Gács, it does have a number of new features, in particular, the algorithm at the heart of the proof works on sets of strings as opposed to individual strings. It is fair to say that Gács's theorem is not well understood. We hope that the new proof, as well as improving the bound, clarifies the main ideas as to why K_m and KM are different.

From now on we will refer to a c.e. continuous semimeasure as just a c.e. semimeasure. We will also drop the word monotonic and just talk about descriptive complexity and algorithmic probability. We know that M_U majorizes

all c.e. semimeasures. So, if it was true that $K_m(\sigma) \leq KM(\sigma) + O(1)$, then for any c.e. semimeasure m that we could construct, there would be some constant c such that $K_m(\sigma) \leq -\log(m(\sigma)) + c$. The general approach of the proof will be to build a c.e. semimeasure for which this is false. This means constructing a c.e. semimeasure m with the property that for all $c \in \omega$, there exists $\sigma \in 2^{<\omega}$, such that $K_m(\sigma) > -\log(m(\sigma)) + c$.

Of course we want to do more than this and prove Theorem 4.1.4. This result will come out of the construction used in the proof. For now we will just focus on showing that K_m and KM are different. We will leave the analysis of the size of the difference until later.

Another way to look at this problem is to consider whether for any $n \in \omega$, we can construct a semimeasure m such that $m(\lambda) \leq 2^{-n}$ and there exists a σ such that $K_m(\sigma) > -\log(m(\sigma))$. Viewing the problem this way helps simplify the proof a little. This is because we can build a c.e. semimeasure $a(\sigma)$ such that:

(i). $a(1) \leq \frac{1}{2}, a(01) \leq \frac{1}{8}$, and for all $i \in \omega, a(0^i1) \leq 2^{-2i-1}$.

(ii). For all $i \in \omega$, $a(0^i) = \sum_{j\in\omega} a(0^{i+j}1)$.

(iii). For all $i \in \omega$ there exists a σ such that $K_m(0^i1\sigma) > -\log(a(0^i1\sigma))$.

Then we can *scale* the semimeasure a to construct a new semimeasure m with the desired properties.

Proposition 4.1.5. *If such a semimeasure exists then Theorem 4.1.1 holds.*

Proof. From a we define a c.e. semimeasure m. For all $\sigma \in 2^{<\omega}$, $i \in \omega$, let $m(0^i1\sigma) = 2^i a(0^i1\sigma)$ and additionally let $m(0^i) = \sum_{j\in\omega} m(0^{i+j}1)$. From this definition for all σ, $m(\sigma) \geq m(\sigma0) + m(\sigma1)$. Also

$$m(\lambda) = \sum_{i\in\omega} m(0^i1) = \sum_{i\in\omega} 2^i a(0^i1) \leq \sum_{i\in\omega} 2^i 2^{-2i-1} \leq 1$$

so m is a c.e. semimeasure. As M_U majorizes all c.e. semimeasures, there is some d, such that $M_U(\sigma) \geq 2^{-d}m(\sigma)$. Thus $KM(\sigma) \leq -\log(2^{-d} \cdot m(\sigma)) = -\log(m(\sigma)) + d$. Now given any $c \in \omega$, choose $i \geq c + d$. By our hypothesis on a, there is some σ such that,

$$\begin{aligned} K_m(0^i1\sigma) &> -\log(a(0^i1\sigma)) \\ &= -\log(2^{-i}m(0^i1\sigma)) \\ &= -\log(m(0^i1\sigma)) + i \\ &\geq -\log(m(0^i1\sigma)) + c + d \\ &\geq KM(0^i1\sigma) + c. \end{aligned}$$

□

Note that we could use the recursion theorem to determine the constant d in the above proof. However, this is not necessary for our construction.

4.1.1 The game: c.e. semimeasures vs monotone machines

In this chapter we will need a computable approximation to K_m so we will take $K_{m,t}$ to be defined as per K_m except with U_t in place of U. We will also want to form a set of strings by concatenating all elements in one set with all elements in another. Hence if $\Sigma, \Upsilon \subseteq 2^{<\omega}$, then we define

$$\Sigma\Upsilon = \{\sigma\upsilon : \sigma \in \Sigma \text{ and } \upsilon \in \Upsilon\}.$$

The semimeasure a is c.e. so we can construct it in stages depending on some computable enumeration of the universal monotone machine U. We will start with $a(\sigma, 0) = 0$ for all $\sigma \in 2^{<\omega}$. At each stage $t+1$, we have the option of choosing a single string ρ and a positive integer x such that $a(\lambda, t) + 2^{-x} \leq 1$, and defining $a(\sigma, t+1)$ as follows:

$$a(\sigma, t+1) = \begin{cases} a(\sigma, t) + 2^{-x} & \text{if } \sigma \preceq \rho \\ a(\sigma, t) & \text{otherwise.} \end{cases}$$

In the rest of the proof we will simply write $a(\rho, t+1) = a(\rho, t) + 2^{-x}$ to refer to this process. If we select ρ at stage $t+1$ and $a(\rho, t) = 0$ we will simplify the notation further and write $a(\rho, t+1) = 2^{-x}$. If no such ρ is chosen at stage $t+1$ then this simply means that $a(\sigma, t+1) = a(\sigma, t)$ for all σ.

Our objective is to construct a c.e. semimeasure a that meets the following requirements for all $i \in \omega$:

R_i: $\quad a(0^i1) \leq 2^{-2i-1}$ and $(\exists \sigma \in 2^{<\omega})(K_m(0^i1\sigma) > -\log(a(0^i1\sigma)))$.

For most of this proof we will focus on meeting a single requirement R_i for some arbitrary i. Once we can achieve a single requirement, it will be relatively easy to meet all requirements.

A natural way to view this construction is as a game between us constructing the semimeasure and an opponent constructing the universal monotone machine. For example, in order to meet the requirement that there is some string 1σ such that $K_m(1\sigma) > -\log(a(1\sigma))$ we could choose a string σ and set $a(1\sigma, 1) = 2^{-r}$. Then we could wait and see if a pair $\langle \tau, 1\sigma' \rangle$ is enumerated into U where $|\tau| \leq r$ and $\sigma \preceq \sigma'$. If such a pair is not enumerated into U then the requirement is met because $K_m(1\sigma) > r = -\log(a(1\sigma))$. If such a pair is enumerated into U then we can make another change to a at some future stage.

We need to develop a construction of a such that no matter how our opponent enumerates U, the requirements on a are met. We only have a finite amount of measure that we can use, and similarly, our opponent only has a finite number of descriptions of any length. Our goal is make our opponent run out of suitable descriptions before we run out of measure. There are some fundamental concepts used by Gács in his proof of Theorem 4.1.2 that we will make use of. We will introduce one of these with the following example.

Example 4.1.6. We choose a string σ and set $a(\sigma, 1) = 2^{-4}$. Now our opponent must respond to this by ensuring that $K_m(\sigma) \leq 4$. To do this, the opponent must enumerate some pair $\langle \tau, \sigma' \rangle$ into U where $|\tau| \leq 4$, and $\sigma' \succeq \sigma$. Let us suppose that the opponent enumerates $\langle 0000, \sigma \rangle$ into U. Here comes the trick. Either it is possible for the opponent to add $\langle 000, \sigma \rangle$ into U or not. If not, we can set $a(\sigma, 2) = a(\sigma, 1) + 2^{-4} = 2^{-3}$. Now as it is not possible for the opponent to add $\langle 000, \sigma \rangle$ to U, the opponent must find a new string, say 001, and add $\langle 001, \sigma \rangle$ to U. However, in this case, the opponent has used both 0000 and 001 to describe σ, a waste of resources. Alternatively, assume that the opponent can enumerate $\langle 000, \sigma \rangle$ into U. We can think of our opponent as holding 000 as a description in reserve for σ. Now we make this reservation useless. This is achieved by:

(i). Never again adding measure to a string comparable with σ.

(ii). Only using units of measure of size at least 2^{-3} from now on.

As we are using increments of measure of size 2^{-3} or more, our opponent must always respond with strings of length at most 3. However, our opponent cannot use 000 because 000 can only be used to describe a string comparable with σ. Hence, if we never increase the measure on a string comparable with σ our opponent can never use 000. This makes 0001 a *gap* in the domain of U as it cannot be used to respond to any of our increments in measure. Again this is a waste of our opponent's resources.

A proof of Theorem 4.1.1 can be developed by generalising the approach of this example. A first step towards this generalisation is the following. We take a string σ. We have two integer parameters r, g with $0 < r < g$. We want to increase the measure on σ in increments of 2^{-g}. Each time we increase the measure we want the opponent to respond. To do this, we do not increase the measure on σ directly but on some extension of σ. Let $\Xi = \{\sigma\}\{0,1\}^{g-r}$. We will take some $\xi \in \Xi$ and increase the measure on ξ instead. A first attempt at the algorithm we need is as follows.

Algorithm 1. Let t be the current stage in the enumeration of U. Choose some $\xi \in \Xi$ that has not been used before (so $a(\xi, t) = 0$). If there is some description of a string σ' with $\sigma' \succeq \sigma$ in the domain of U_t that can be *shortened* to a description of length r, then terminate the algorithm. Otherwise, we increase the measure on ξ by 2^{-g}, and wait until a stage t' when the opponent describes ξ with a string of length at most g. If we have not used all elements of Ξ, then we let $t = t'$ and repeat. If we have used all elements of Ξ, then the measure on σ must be $|\Xi|2^{-g} = 2^{-r}$ so we wait until the opponent uses a string of length r to describe σ.

If we apply this algorithm, then the opponent must either use a string of length r to describe σ, or have some string that describes a string comparable with σ that can be shortened to a string of length r. The crucial fact is at some point, we increased the measure on σ by *just* 2^{-g}, and the opponent had to respond by finding a *new* string of length r. The string is new in the sense that it is not an initial segment of a string already used.

As it stands, this algorithm can only make a small gap between K_m and KM. The hard part is amplifying it. Amplification can be done by using the algorithm recursively. In the algorithm above, we act by setting $a(\xi, t) = 2^{-g}$ for some ξ at some stage t. However, rather than just directly increasing the measure on ξ, we could instead run the algorithm again on some *extension* of ξ. This would have the effect of increasing the measure on ξ while forcing the opponent to waste even more resources. However, to achieve this a number issues need to dealt with:

- We want to make sure our algorithm increases the measure enough. Algorithm 1 will increase the measure on σ somewhere between 2^{-g} and 2^{-r}. This is too great a range.

- We need to ensure that any gaps the algorithm makes in the domain of U are never used again. The algorithm above does not address this.

- Each time we use an extension of a string, we go deeper into the tree of binary strings. We would like to minimise the depth that we go to in order to get the best possible lower bound on the difference between K_m and KM.

The concepts in Example 4.1.6 and Algorithm 1 were used by Gács in his original proof of Theorem 4.1.2. The approach of this chapter is still based on these ideas but differs from that of Gács in how the problems listed above are addressed.

The algorithm that we will present will make decisions based on the domain of U. The question of whether there is a description of σ that can be shortened will be crucial (e.g. whether or not 0000 could be shortened to 000 in Example 4.1.6). First we will define certain subsets of the domain of U. We will prove some technical lemmas about how these sets change as the parameters defining them change. These lemmas will be crucial to the verification of the proof. Secondly, we will present the algorithm that establishes a gap between K_m and KM. Thirdly, we will verify the algorithm works. Finally, we will analyse the algorithm to show how it improves the lower bound.

4.2 Examining the domain of U

At any stage t, U_t tells us the options our opponent has left. In particular, U_t tells us what our opponent can and cannot use certain strings for. We need a more exact notion for the idea of the opponent shortening an existing description. In his original proof, Gács came up with the notion of a string τ being *reserved* as a description for σ. The idea is not just that the opponent can use τ to describe σ, but additionally that the opponent cannot use τ to describe a string incomparable with σ. For τ to be reserved for σ, the opponent must have already used a string comparable with τ to describe an extension of σ.

There are really two ideas in this definition that are worth separating out and making more explicit. First we will say that a string τ is *fragmented* by σ if some string comparable to τ describes an extension of σ. This means the opponent cannot use τ to describe a string incomparable with σ (otherwise U would not be a monotone machine). Second, a string τ is *incompatible* with σ if some string comparable with τ describes a string incomparable with σ. This means that τ cannot be used to describe σ. Hence another way of saying a string τ is reserved for σ is that τ is fragmented by σ but τ is not incompatible with σ. As we are not so much interested in particular descriptions, but descriptions of a certain length, we will make the following definitions.

Definition 4.2.1. (i). The strings of length r *fragmented* by σ at stage t are:

$$F_r(\sigma, t) = \{\tau \in \{0,1\}^r : \exists \langle \tau', \sigma' \rangle \in U_t((\tau' \approx \tau) \wedge (\sigma \preceq \sigma'))\}.$$

(ii). The strings of length r *incompatible* with σ at stage t are:

$$I_r(\sigma, t) = \{\tau \in \{0,1\}^r : \exists \langle \tau', \sigma' \rangle \in U_t((\tau' \approx \tau) \wedge (\sigma \mid \sigma'))\}.$$

(iii). The strings of length r *reserved* for σ at stage t are:

$$R_r(\sigma, t) = F_r(\sigma, t) \setminus I_r(\sigma, t).$$

Example 4.2.2. Assume at stage t, $U_t = \{\langle 00, 11\rangle, \langle 000, 1\rangle, \langle 000, 11\rangle, \langle 001, 11\rangle, \langle 011,01\rangle, \langle 100,01\rangle, \langle 110, 011\rangle,\langle 111, 1\rangle\}$. If we consider the strings of length 2, working through the definitions will establish that: $F_2(01, t) = \{01, 10, 11\}$, $I_2(01, t) = \{00, 11\}$, and $R_2(01, t) = \{01, 10\}$.

The definitions given for F_r and I_r can be extended to sets of strings.

Definition 4.2.3. If Σ is a set of finite strings, then:

(i). $F_r(\Sigma, t) = \bigcup_{\sigma\in\Sigma} F_r(\sigma, t)$.

(ii). $I_r(\Sigma, t) = \bigcap_{\sigma\in\Sigma} I_r(\sigma, t)$.

We will explain later why we take the intersection in the definition of $I_r(\Sigma, t)$. The following lemmas show how the sets $F_r(\sigma, t)$, $I_r(\sigma, t)$, $F_r(\Sigma, t)$, and $I_r(\Sigma, t)$ change as the parameters that define them vary.

Lemma 4.2.4. *The following hold:*

(i). If $r_0 \leq r_1$ then $[F_{r_0}(\sigma, t)] \supseteq [F_{r_1}(\sigma, t)]$ and $[I_{r_0}(\sigma, t)] \supseteq [I_{r_1}(\sigma, t)]$.

(ii). If $\sigma_0 \preceq \sigma_1$ then $[F_r(\sigma_0, t)] \supseteq [F_r(\sigma_1, t)]$ and $[I_r(\sigma_0, t)] \subseteq [I_r(\sigma_1, t)]$.

(iii). If $t_0 \leq t_1$ then $[F_r(\sigma, t_0)] \subseteq [F_r(\sigma, t_1)]$ and $[I_r(\sigma, t_0)] \subseteq [I_r(\sigma, t_1)]$.

Proof. (i) If $\alpha \in [F_{r_1}(\sigma, t)]$ then $\alpha \restriction r_1 \in F_{r_1}(\sigma, t)$, so there are $\tau \approx \alpha \restriction r_1$ and $\sigma' \succeq \sigma$ such that $\langle \tau, \sigma'\rangle \in U_t$. Now as $r_0 \leq r_1$, $\tau \approx \alpha \restriction r_0$ we have $\alpha \restriction r_0 \in F_{r_0}(\sigma, t)$ and hence $\alpha \in [F_{r_0}(\sigma, t)]$. Similarly $[I_{r_0}(\sigma, t)] \supseteq [I_{r_1}(\sigma, t)]$.
(ii) If $\tau \in F_r(\sigma_1, t)$ then there are $\tau' \approx \tau$ and $\sigma \succeq \sigma_1$ such that $\langle \tau', \sigma\rangle \in U_t$. As $\sigma \succeq \sigma_1 \succeq \sigma_0$ we have $\tau \in F_r(\sigma_0, t)$. If $\tau \in I_r(\sigma_0, t)$ then there exist $\tau' \approx \tau$ and $\rho \mid \sigma_0$ such that $\langle \tau', \rho\rangle \in U_t$. As $\sigma_1 \succeq \sigma_0$, we have $\sigma_1 \mid \rho$ and thus $\tau \in I_r(\sigma_1, t)$.
(iii) Follows because $U_{t_0} \subseteq U_{t_1}$. □

In summary $[F_r(\sigma, t)]$ and $[I_r(\sigma, t)]$ are both increasing in t and decreasing in r. In length of σ, $[F_r(\sigma, t)]$ is decreasing and $[I_r(\sigma, t)]$ is increasing. To establish similar results for $F_r(\Sigma, t)$, and $I_r(\Sigma, t)$, we need to define a relationship $\preceq$ between sets of strings.

Definition 4.2.5. If $\Sigma_0, \Sigma_1 \subseteq 2^{<\omega}$, then $\Sigma_0 \preceq \Sigma_1$ if for all $\sigma_1 \in \Sigma_1$, there exists a $\sigma_0 \in \Sigma_0$ such that $\sigma_0 \preceq \sigma_1$.

Lemma 4.2.6. *The following hold:*

(i). If $r_0 \leq r_1$ then $[F_{r_0}(\Sigma, t)] \supseteq [F_{r_1}(\Sigma, t)]$ and $[I_{r_0}(\Sigma, t)] \supseteq [I_{r_1}(\Sigma, t)]$.

(ii). If $\Sigma_0 \preceq \Sigma_1$ then $[F_r(\Sigma_0, t)] \supseteq [F_r(\Sigma_1, t)]$ and $[I_r(\Sigma_0, t)] \subseteq [I_r(\Sigma_1, t)]$.

(iii). If $t_0 \leq t_1$ *then* $[F_r(\Sigma, t_0)] \subseteq [F_r(\Sigma, t_1)]$ *and* $[I_r(\Sigma, t_0)] \subseteq [I_r(\Sigma, t_1)]$.

Proof. (i) If $\alpha \in [F_{r_1}(\Sigma, t)]$, then for some $\sigma \in \Sigma$, $\alpha \in [F_{r_1}(\sigma, t)]$ thus by the previous lemma $\alpha \in [F_{r_0}(\sigma, t)] \subseteq [F_{r_0}(\Sigma, t)]$. If $\alpha \in [I_{r_1}(\Sigma, t)]$, then for all $\sigma \in \Sigma$, $\alpha \in [I_{r_1}(\sigma, t)]$ thus by the previous lemma $\alpha \in \bigcap_{\sigma\in\Sigma}[I_{r_0}(\sigma, t)] = [I_{r_0}(\Sigma, t)]$.
(ii) If $\alpha \in [F_r(\Sigma_1, t)]$ then for some $\sigma_1 \in \Sigma_1$, $\alpha \in [F_r(\sigma_1, t)]$. There exists $\sigma_0 \in \Sigma_0$ such that $\sigma_0 \preceq \sigma_1$, by the previous lemma $\alpha \in [F_r(\sigma_0, t)] \subseteq [F_r(\Sigma_0, t)]$.

If $\alpha \in [I_r(\Sigma_0, t)]$ then for all $\sigma_0 \in \Sigma_0$, $\alpha \in [I_r(\sigma_0, t)]$. Now if $\sigma_1 \in \Sigma_1$ then for some $\sigma_0 \in \Sigma_0$, $\sigma_0 \preceq \sigma_1$. Because $\alpha \in [I_r(\sigma_0, t)]$, by the previous lemma $\alpha \in [I_r(\sigma_1, t)]$. Thus $\alpha \in \bigcap_{\sigma_1\in\Sigma_1}[I_r(\sigma_1, t)] = [I_r(\Sigma_1, t)]$.
(iii) If $\tau \in F_r(\Sigma, t_0)$, then for some $\sigma \in \Sigma$, $\tau \in F_r(\sigma, t_0)$ thus by the previous lemma $\tau \in F_r(\sigma, t_1) \subseteq F_r(\Sigma, t_1)$.

If $\tau \in I_r(\Sigma, t_0)$, then for all $\sigma \in \Sigma$, $\tau \in I_r(\sigma, t)$ thus by the previous lemma $\tau \in \bigcap_{\sigma\in\Sigma} I_r(\sigma, t_1) = I_r(\Sigma, t_1)$.

□

4.3 The main algorithm

We are going to describe an algorithm that can be used to prove Theorems 4.1.1 and 4.1.4. The algorithm will be allocated a certain amount of amount of measure to spend on the construction of the c.e. semimeasure a. It will also be given a set of strings Σ, and it will only be able to increase the measure on extensions of these. Ultimately we want the algorithm to establish some difference between $K_m(\rho)$ and $-\log(a(\rho))$ for some string ρ that extends some element of Σ. We will argue that if this does not happen, then some contradiction must ensue. The basic idea is to make the opponent run out of short descriptions. Formalising this idea is a little difficult. We want to be able to say that if we spend 2^{-r} of measure on extensions of Σ, and we do not establish a difference between K_m and $-\log(a)$, then we can cause x amount of something to happen to U (where U is the universal machine controlled by the opponent). We will refer to this x, whatever it is, as the gain made by the algorithm.

Now the most obvious measurement to use would be $\mu([\Pi_1(U_t)])$ (where $\Pi_1(U_t)$ is the projection of U_t on the first coordinate) because this is analogous to the domain of U_t. However, remember that part of idea is to leave gaps in U that the opponent cannot do anything useful with. The problem with this measurement is that it does not account for gaps. An alternative is $F_r(\Sigma, t)$, the strings of length r that are fragmented by elements of Σ. This will count gaps as well. This is almost what we need. The size of this set must be at most 2^r.

So if the algorithm ends at t_1, and we could ensure that $|F_r(\Sigma, t_1)| \geq k$ for an arbitrary k we would be done. However, it is still more complicated than this. We will end up using the algorithm we define recursively. When verifying the algorithm we do not want to double-count any gain made. To make the verification possible, we will subtract $[I_{r'}(\Sigma, t_0)]$ from $[F_r(\Sigma, t_1)]$, where $r' \geq r$ and t_0 is the time the algorithm starts. This is what we will use to determine the gain made by the algorithm. The following definition, G is for gain, codifies this idea.

Definition 4.3.1. If $\Sigma \subseteq 2^{<\omega}$, and $r, r', t_0, t_1 \in \omega$ such that $r \leq r'$ and $t_0 \leq t_1$ then $G_r^{r'}(\Sigma; t_0, t_1) = [F_r(\Sigma, t_1)] \setminus [I_{r'}(\Sigma, t_0)]$.

$G_r^{r'}(\Sigma; t_0, t_1)$ can be thought of as the gain the algorithm obtains using the strings in Σ between stages t_0 and t_1, and using units of measure between r and r'. With this definition in hand, we can define what sort of algorithm is needed. Note that we describe an algorithm that works on a set of strings.

Definition 4.3.2. If $r, f \in \omega$, and Σ is a finite prefix-free subset of $2^{<\omega}$, then an (r, f, Σ)-*strategy* is an algorithm that if implemented at stage t_0, ensures that either:

(i). (a) There is some $\sigma \in \Sigma$ such that for some σ' extending σ, $K_m(\sigma') > -\log(a(\sigma'))$, and

(b) For all stages t, for all $\sigma \in \Sigma$, $a(\sigma, t) < \frac{5}{4}2^{-r}$; or

(ii). At some stage t_1, for some $r' \geq r$ computable from (r, f):

(a) For all $\sigma \in \Sigma$, $2^{-r} \leq a(\sigma, t_1) < \frac{5}{4}2^{-r}$, and

(b) $\mu(G_r^{r'}(\Sigma; t_0, t_1)) \geq \left(1 + \frac{f}{2}\right) a(\Sigma, t_1)$, and

(c) For all $\hat{\Sigma} \subseteq \Sigma$,
$\mu(G_r^{r'}(\hat{\Sigma}; t_0, t_1)) \geq \left(1 + \frac{f}{2}\right) \Big(a(\Sigma, t_1) - 2a(\Sigma \setminus \hat{\Sigma}, t_1)\Big)$

where $a(\Sigma, t) = \sum_{\sigma \in \Sigma} a(\sigma, t)$.

This definition is complicated so we will take some time to explain it. The input parameters for the strategy are r,f and Σ. Σ is a finite prefix-free set of binary strings. These are the strings that the algorithm will work on. The parameter r determines the measure that the algorithm should spend on *every* string in Σ. The parameter f determines the amount of gain that the algorithm should make. First we will establish the existence of an $(r, 0, \Sigma)$-strategy for any appropriate r, then we will inductively establish the existence of $(r, f + 1, \Sigma)$-strategies assuming the existence of (r, f, Σ)-strategies.

The definition gives two possible outcomes to a strategy. Outcome (i) is the preferred outcome. If outcome (i) occurs, then we have established a difference between K_m and KM. However, if we do not achieve this, then outcome (ii) is at least a step in the right direction. Provided $f > 0$, outcome (ii) ensures that some difference between $\mu(G_r^{r'}(\Sigma; t_0, t_1))$ and $a(\Sigma, t_1)$ is created.

In outcome (ii), condition (b) says that the gain on the set of strings Σ is bounded below. We do not want the gain to be too concentrated on some particular subset of Σ. The gain achieved need not be evenly distributed among the elements of Σ, but it is important that this distribution is not too skewed. This is the reason for condition (c). Condition (c) ensures that no individual string in Σ contributes too much to the overall gain made. We need condition (c) because when we use the algorithm recursively, we will not be able to count the gain that occurs on some individual elements of Σ. Hence we do not want an individual element of Σ providing too much of the overall gain. Note that (c) implies (b) by taking $\hat{\Sigma} = \Sigma$.

It is also important to observe that for all $\sigma \in \Sigma$, if the strategy is running at stage t, then $a(\sigma, t)$ is bounded above in outcomes (i) and (ii). In outcome (ii) there is also a lower bound for $a(\sigma, t_1)$. This lower bound is essential for using strategies recursively because we want these strategies to increase the measure on strings, as well as establish some gain. If we can implement an (r, f, Σ)-strategy with an arbitrarily high f then (ii) is not a possible outcome as the measure would be greater than 1. This would force outcome (i).

The other parameter in the above definition to discuss is r'. The parameter r' dictates where the strategy should make its gain. In practice, what it will mean is that $2^{-r'}$ will be the minimum sized increment in measure that the strategy will make to a. In other words, each time the strategy increases the measure on some string, the amount increased will be at least $2^{-r'}$. The idea is that if any previous strategy has created a gap of measure less that $2^{-r'}$, then the current strategy will not interfere with it.

4.3.1 A basic strategy

The idea behind the basic $(r, 0, \Sigma)$-strategy is simple. For every $\sigma \in \Sigma$ we set $a(\sigma) = 2^{-r}$. The opponent is then forced to respond by setting $K_m(\sigma) \leq r$. To do this, the opponent must find some string τ of length at most r and enumerate $\langle \tau, \sigma \rangle$ into U. The only difficulty is in showing that this basic idea meets our rather elaborate definition of a strategy.

Proposition 4.3.3. *If Σ is a finite prefix-free subset of $2^{<\omega}, r \in \omega$, and t_0 is the current stage in the construction of a, such that $a(\Sigma, t_0) = 0$, and $|\Sigma|2^{-r} \leq 1$, we can*

implement an $(r, 0, \Sigma)$*-strategy with* $r' = r$*.*

Proof. Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$. This strategy can be implemented by first for all $i \in \omega$, $1 \leq i \leq n$ setting $a(\sigma_i, t_0 + i) = 2^{-r}$ (we know that $a(\sigma_i, t_0 + i - 1) = 0$ because Σ is prefix-free). This ensures that $a(\Sigma, t_0 + n) = |\Sigma|2^{-r}$. The second step is to wait until a stage t_1, when the opponent responds by setting $K_{m,t_1}(\sigma) \leq r$ for all $\sigma \in \Sigma$. If this never happens, then for some $\sigma \in \Sigma$, $K_m(\sigma) > r = -\log(2^{-r}) = -\log(\lim_{t\to\infty} a(\sigma, t)) = -\log(a(\sigma))$ so outcome (i) occurs.

If at some stage t_1, $K_{m,t_1}(\sigma) \leq r$ for all $\sigma \in \Sigma$, then for all $i \in \omega$, such that $1 \leq i \leq n$, our opponent must have enumerated some $\langle \tau_i, \sigma_i' \rangle$ into U_{t_1} where $s_i = |\tau_i| \leq r$ and $\sigma_i' \succeq \sigma_i$. In this case we will show that outcome (ii) occurs. First (a) holds as $a(\Sigma, t_1) = a(\Sigma, t_0 + n)$.

For any i, let τ_r be any extension of τ_i of length r. By definition, $\tau_r \in F_r(\sigma_i, t_1)$ so $\tau_r \in \bigcup_{\sigma\in\Sigma} F_r(\sigma, t_1) = F_r(\Sigma, t_1)$. If $\tau_r \in I_r(\sigma_i, t_0)$, then for some τ' comparable with τ_r and some ρ incomparable with σ_i, $\langle \tau', \rho \rangle \in U_{t_0} \subseteq U_{t_1}$ but this would contradict the definition of a monotone machine as it implies both $\tau' \approx \tau_i$ and $\rho \mid \sigma_i'$. So $\tau_r \notin I_r(\sigma_i, t_0)$ and thus $\tau_r \notin \bigcap_{\sigma_i\in\Sigma} I_r(\sigma_i, t_0) = I_r(\Sigma, t_0)$. Hence $G_r^r(\Sigma; t_0, t_1)$ which by definition is $[F_r(\Sigma, t_1)] \setminus [I_r(\sigma, t_0)]$ contains all infinite extensions of τ_r and hence all infinite extensions of τ_i. So $[\tau_i] \subseteq G_r^r(\Sigma; t_0, t_1)$. Now if $i \neq j$ $[\tau_i] \cap [\tau_j] = \emptyset$ because if they are comparable then σ_i and σ_j must be comparable but Σ is a prefix-free set. Thus

$$\mu(G_r^r(\Sigma; t_0, t_1)) \geq \sum_{i=1}^{n} \mu[\tau_i] = \sum_{i=1}^{n} 2^{-s_i} \geq \sum_{i=1}^{n} 2^{-r} = |\Sigma|2^{-r} = a(\Sigma, t_1)$$

and as $a(\Sigma, t_1) = (1 + \frac{0}{2})a(\Sigma, t_1)$, we have that (b) holds.

Let $\hat{\Sigma} \subseteq \Sigma$. Let $I = \{1, \ldots, n\}$ and $J = \{i \in I : \sigma_i \in \hat{\Sigma}\}$. By the same logic,

$$\mu(G_r^r(\hat{\Sigma}; t_0, t_1)) \geq \sum_{j\in J} 2^{-r} = |I|2^{-r} - (|I \setminus J|)2^{-r} = a(\Sigma, t_1) - a(\Sigma \setminus \hat{\Sigma}, t_1)$$

and as $a(\Sigma, t_1) - a(\Sigma \setminus \hat{\Sigma}, t_1) \geq (1 + \frac{0}{2})(a(\Sigma, t_1) - 2a(\Sigma \setminus \hat{\Sigma}, t_1))$ so (c) holds and outcome (ii) occurs. □

4.3.2 Using strategies sequentially

We are going to work up to developing an $(r, f + 1, \Sigma)$-strategy, using (r, f, Σ)-strategies. Before we present the main algorithm that does this, there is some more preparatory work to do. Ideally, when we implement a strategy, we want to achieve outcome (i). However, if this does not happen then at some stage the strategy will terminate and we can start a new strategy. We want to ensure that if the second strategy also achieves outcome (ii), then the gain from

running the strategies sequentially is at least the sum of the minimum gain expected from the strategies individually. The definition of the (r, f, Σ)-strategy is designed to allow this as Proposition 4.3.5 will show. Before proving this proposition, we need the following lemma.

Lemma 4.3.4. *If $\sigma_0 \mid \sigma_1$ then $F_p(\sigma_0, t) \subseteq I_p(\sigma_1, t)$.*

Proof. If $\tau \in F_p(\sigma_0, t)$ then there exists $\langle \tau', \sigma_0' \rangle \in U_t$ such that $\tau' \approx \tau$ and $\sigma_0' \succeq \sigma_0$. Now this implies that $\sigma_0' \mid \sigma_1$ and so $\tau \in I_p(\sigma_1, t)$. □

Proposition 4.3.5. *If $r_0 \leq r_1 \leq \ldots \leq r_n$ and $t_0 \leq t_1 \leq \ldots \leq t_n$ are sequences in ω, and $\Sigma_0, \Sigma_1, \ldots, \Sigma_{n-1}$ are pairwise disjoint prefix-free subsets of $2^{<\omega}$, and $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \ldots \cup \Sigma_{n-1}$ is also prefix-free, then:*

$$\mu(G_{r_0}^{r_n}(\Sigma; t_0, t_n)) \geq \sum_{i=0}^{n-1} \mu(G_{r_{n-1-i}}^{r_{n-i}}(\Sigma_i; t_i, t_{i+1})).$$

Proof. First if $i \in \omega$ and $0 \leq i < n$, then $\Sigma \preceq \Sigma_i$ so by chaining together Lemma 4.2.6 $[F_{r_{n-i-1}}(\Sigma_i, t_{i+1})] \subseteq [F_{r_0}(\Sigma_i, t_{i+1})] \subseteq [F_{r_0}(\Sigma, t_{i+1})] \subseteq [F_{r_0}(\Sigma, t_n)]$ and additionally $[I_{r_n}(\Sigma, t_0)] \subseteq [I_{r_n}(\Sigma, t_i)] \subseteq [I_{r_n}(\Sigma_i, t_i)] \subseteq [I_{r_{n-i}}(\Sigma_i, t_i)]$.

Hence $[F_{r_0}(\Sigma, t_n)] \setminus [I_{r_n}(\Sigma, t_0)] \supseteq [F_{r_{n-i-1}}(\Sigma_i, t_{i+1})] \setminus [I_{r_{n-i}}(\Sigma_i, t_i)]$ which by definition means that $G_{r_0}^{r_n}(\Sigma; t_0, t_n) \supseteq G_{r_{n-i-1}}^{r_{n-i}}(\Sigma_i; t_i, t_{i+1})$. Now assume $0 \leq i < j < n$. We have that:

$$[F_{r_{n-i-1}}(\Sigma_i, t_i)] \subseteq [F_{r_{n-j}}(\Sigma_i, t_j)] \subseteq [I_{r_{n-j}}(\Sigma_j, t_j)].$$

The first inclusion is by Lemma 4.2.6. The second inclusion follows because if $\tau \in F_{r_{n-j}}(\Sigma_i, t_j)$, then $\tau \in F_{r_{n-j}}(\sigma_i, t_j)$ for some $\sigma_i \in \Sigma_i$. But for all $\sigma_j \in \Sigma_j$, σ_j is incomparable with σ_i. So $\tau \in I_{r_{n-j}}(\sigma_j, t_j)$ by Lemma 4.3.4. Thus $\tau \in \bigcap_{\sigma_j \in \Sigma_j} I_{r_{n-j}}(\sigma_j, t_j) = I_{r_{n-j}}(\Sigma_j, t_j)$. So we can conclude that:

$$([F_{r_{n-i-1}}(\Sigma_i, t_{i+1})] \setminus [I_{r_{n-i}}(\Sigma_i, t_i)]) \cap ([F_{r_{n-j-1}}(\Sigma_j, t_{j+1})] \setminus [I_{r_{n-j}}(\Sigma_j, t_j)]) = \emptyset.$$

This by definition means that $G_{r_{n-i-1}}^{r_{n-i}}(\Sigma_i; t_i, t_{i+1}) \cap G_{r_{n-j-1}}^{r_{n-j}}(\Sigma_j; t_j, t_{j+1}) = \emptyset$. □

Buried in the above proof is the reason that we define $I_r(\Sigma, t)$ to be the intersection of $I_r(\sigma, t)$ for all $\sigma \in \Sigma$. The reason is this. Take Σ_0 and Σ_1 to be disjoint sets whose union is prefix-free. If $\tau \in F_r(\Sigma_0, t)$ then τ is fragmented by some string in Σ_0. As all strings in Σ_0 are incomparable with all strings in Σ_1, we have $\tau \in I_r(\Sigma_1, t)$. This is why the above proposition works as it avoids double counting any gains.

Example 4.3.6. To illustrate why this proposition is useful, we will show how we can use it to gradually increase the measure on a string without losing any

gain made along the way. Consider the following implementation of three strategies in sequence and assume that they all have outcome (ii). Take some $r_0 \in \omega$. Fix an $f \in \omega$ and assume we have an (r, f, Σ) strategy for any appropriate r and Σ. Let r_1 be the r' computable from (r_0, f). Similarly let r_2 be the r' computable from (r_1, f) and r_3 be the r' computable from (r_2, f). Take any string σ and let $\Sigma_2 = \{\sigma 00\}$, $\Sigma_1 = \{\sigma 01\}\{0,1\}^{r_1 - r_0}$, $\Sigma_0 = \{\sigma 1\}\{0,1\}^{r_2 - r_0}$ and $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$.

Starting at stage t_0, first we implement an (r_2, f, Σ_0)-strategy. When this finishes at stage t_1, we implement an (r_1, f, Σ_1)-strategy. Finally when it finishes at stage t_2, we implement an (r_0, f, Σ_2)-strategy. Let t_3 be the stage that this final strategy finishes. Now from Proposition 4.3.5,

$$\begin{aligned}
\mu(G^{r_3}_{r_0}(\Sigma; t_0, t_3)) &\geq \sum_{i=0}^{2} \mu(G^{r_{3-i}}_{r_{3-1-i}}(\Sigma_i; t_i, t_{i+1})) \\
&\geq \sum_{i=0}^{2} \left(1 + \frac{f}{2}\right) a(\Sigma_i, t_{i+1}) \\
&= \left(1 + \frac{f}{2}\right) a(\Sigma, t_3).
\end{aligned}$$

The last step follows provided we do not increase the measure on any other string comparable with a string in Σ except as required by the strategies. Now $a(\sigma) \geq a(\Sigma) \geq |\Sigma_0| 2^{-r_2} + |\Sigma_1| 2^{-r_1} + |\Sigma_2| 2^{-r_0} = 3 \cdot 2^{-r_0}$. This approach has allowed us to use strategies to increase the measure on σ in three increments of 2^{-r_0} without losing any of the gain made by the three strategies individually. This ability to increase the measure on a string in small increments will be exploited in the main proof.

4.3.3 Implementing more difficult strategies

We have seen that we can use strategies sequentially without losing any gain. However, we need to do better than this. We need to be able to combine strategies in such a way as to increase the gain we make. To do this, we will make use of reservations. Our goal is to implement an $(r, f + 1, \Sigma)$-strategy using a finite sequence of (r_i, f, Σ_i)-strategies.

We will improve Algorithm 1. The basic idea remains the same. When we assign measure to a string, e.g. if at stage t_0 we set $a(\sigma, t_0) = 2^{-g}$, then our opponent has to respond by allocating a string of length at most g to describe σ. Say in response to our setting $a(\sigma, t_0) = 2^{-g}$, our opponent enumerates $\langle \tau, \sigma \rangle$ into U_{t_1} where $|\tau| = g$. Take $p < g$ and let τ_p be the first p bits of τ. If $\tau_p \notin R_p(\sigma, t)$, then $\tau_p \in I_p(\sigma, t)$ so our opponent can never enumerate $\langle \tau_p, \sigma \rangle$

into U. So if we set $a(\sigma, t_1) = 2^{-p}$, our opponent must find some new string υ of length at most p to describe σ. The original τ description of σ is effectively a waste of resources.

On the other hand while $\tau_p \in R_p(\sigma, t)$, we also have a gain because our opponent can only use τ_p to describe a string comparable with σ. If we only increase measure on strings incomparable with σ, then our opponent cannot use τ_p in response. If we only use increments of measure of size at least 2^{-p}, then our opponent cannot make use of any extension of τ_p (it would be too long to be useful). Note that this is why we need an r' in the definition of an (r, f, Σ)-strategy. The r' is the smallest amount of measure this strategy needs. Being able to compute r' means we know how much space strategies need in terms of units of measure. With this knowledge, we can ensure that the smallest amount of measure needed by a strategy will not interfere with any reservations established by previous strategies.

We noted earlier that part of the problem with Algorithm 1, was that the amount of measure that it assigned to a string ranged between 2^{-g} and 2^{-r}. This range was too great to allow the algorithm to be used recursively. Hence our algorithm has two objectives: create a certain amount of gain, and ensure a certain amount of measure is allocated. These two objectives will be achieved by splitting the algorithm into two phases. In the first, the advantage phase, the objective will be to force our opponent to reserve a number of strings of length p. We do not know how much measure we will need to use before our opponent makes this many reservations. If we do not use enough measure, then we proceed with the second phase, the spend phase. The spend phase is where we make sure that we have placed enough measure on each string so that the strategies can be used recursively. The use of these two phases is another new feature of this proof.

The spend phase must occur after the advantage phase because only then will we know how much more measure we need to allocate. Accordingly, the spend phase will use larger units of measure than the advantage phase. From r we will compute a p, q and r' such that $r < p < q < r'$. The advantage phase will use units of measure between q and r' while the spend phase will use units of measure between r and p (see Figure 4.1). This gives us a bit of problem. During the advantage phase we can only require reservations of length at most p. If p is much larger than r, then we will need to make a lot of reservations in order to achieve anything. To get around this problem, we define $\Upsilon = \Sigma\{0\}\{0,1\}^{p-r}$ (where Σ is the set of input strings for the algorithm). Note that the purpose of the $\{0\}$ in the definition of Υ is just to separate the strings used by the advantage phase from those used by the spend phase.

We will run the algorithm on the strings in Υ until enough of these have a reservation of length p. An important new idea in this proof is to run each pass of the algorithm *simultaneously* on all the strings in Υ that do not have a reservation.

Like in Algorithm 1, we will not increase the measure on the elements of Υ directly, but rather we will make a series of small increments of measure on some extensions of them. We will let $\Xi = \Upsilon\{0,1\}^e$ (e depends of f and will be defined shortly). The idea is this. If $\upsilon \in \Upsilon$ and $\xi_0, \ldots \xi_{2^e-1}$ are the elements of Ξ that extend υ, then we will sequentially set the measure of each ξ_i to 2^{-p-e} until a reservation of size p occurs for υ. If we do this for all such ξ_i, then the measure on υ will be 2^{-p} and the opponent must allocate a string of length p to describe υ. However, it is not quite that simple. We want to increase the measure on ξ_i by running some strategy on it. Every time we run a strategy we need to use larger units of measure. So instead what we will do is take many extensions of ξ_0. Then we can run our first strategy on all these extensions, spending a little bit of measure on each. Then we will take fewer extensions of ξ_1, and run a strategy that spends a little more measure on each of these extensions, and so on. This is just like what we did in Example 4.3.6. This gives us sets $\Psi_0, \Psi_1, \ldots$, where Ψ_i is the ith set of extensions of elements of Ξ that we want to increase measure on. The sets used by the advantage phase are shown in Figure 4.2. In this figure arrows represent extensions of strings.

The algorithm would ideally achieve a reservation of length p for every element of Υ but there is a problem. Once a reservation is achieved for some $\upsilon \in \Upsilon$, we do not want to increase the measure on that υ. However, reservations are not monotonic, they can appear at one stage and then disappear the next. Consider $\upsilon_1, \upsilon_2 \in \Upsilon$. Say we increase the measure on both υ_1, υ_2 by 2^{-p-e} and then the opponent responds by giving a p reservation to υ_1, but not υ_2. Then we could increase the measure on υ_2 and the opponent might respond by giving a p reservation to υ_2 but taking away the p reservation from υ_1. By adopting this sort of tactic, the opponent could force us to make a lot of separate increases in measure in order to achieve reservations for all elements of Υ. Every time we increase the measure on some subset of Υ we need a new Ψ_i. The more Ψ_i's we need, the deeper we need to go into the binary tree. To improve the lower bound, we want to limit the number of Ψ_i's we need. Each time we increase the measure, we do so simultaneously on all the elements of some subset of Υ. We want to make sure that this subset includes at least one-quarter of the elements of Υ. To do this, we will terminate the advantage phase of the algorithm when we have a reservation for three-quarters of the elements of Υ.

The spend phase is similar but simpler. In this phase we will identify those elements $\sigma \in \Sigma$ that do not have enough measure. We will increase the measure on σ in increments of 2^{-r-3} until we have exceeded 2^{-r}. Again like Example 4.3.6, we will not increase the measure directly on σ but rather on some set of extensions of it. These extensions will form the sets $\Phi_0, \ldots, \Phi_7$.

4.3.4 The S function

The length of extensions that we take to form the sets Ψ_i and Φ_i is governed by the space that a strategy needs to run, in terms of quantities of measure. The function $S(f)$ is used to determine this space. We will show that there exist (r, f, Σ)-strategies such that $r' = r + S(f)$. From our base strategy we can set $S(0) = 0$ because in this case $r = r'$.

In the process of determining $S(f + 1)$ we will also compute all the other variables we need for the main algorithm. These are illustrated in Figure 4.1. To determine $S(f + 1)$, first we compute an increasing sequence $r_0 \leq r_1 \leq \ldots \leq r_8$ by $r_0 = r + 3$ and $r_{i+1} = r_i + S(f)$. This gives us room for the spend phase. Let $p = r_8$. Let $e = \lceil \log(20 + 10f) \rceil$. We use this value of e so that $\frac{5}{2}2^{-e}\left(1 + \frac{f}{2}\right) \leq \frac{1}{8}$.

Let $q = p + e$. Now compute an increasing sequence $r_0^* \leq r_1^* \leq \ldots \leq r_n^*$, where $r_0^* = q$, $n = 2^{e+2}$ and $r_{i+1}^* = r_i^* + S(f)$. This gives us room for the advantage phase. Finally let $r' = r_n^*$, and $S(f+1) = r' - r$. This is well-defined as $S(f + 1)$ does not depend on the initial choice of r.

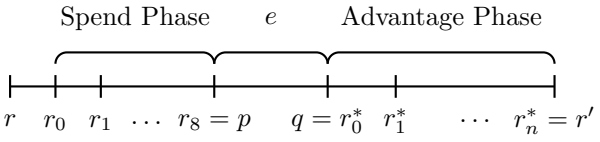

Figure 4.1: Algorithm parameters

4.3.5 An f+1 strategy algorithm

We are finally ready to present the main algorithm. We are given $r, f \in \omega$ and a prefix-free set of strings Σ such that $|\Sigma|\frac{5}{4}2^{-r} \leq 1$. We let t_0 be the stage that the algorithm starts and we require that $a(\Sigma, t_0)$=0. In order to implement an $(r, f + 1, \Sigma)$-strategy first we determine all the values $n, p, q, e, r_i, r_i^*, r'$ as shown in Figure 4.1. In the algorithm Υ' represents those elements of Υ whose measure we will increase in the current pass through the advantage phase of

the algorithm. Now with all the values that we need computed, the algorithm that we will use to implement an $(r, f+1, \Sigma)$-strategy is as follows:

Advantage Phase. Let t be the current stage.
Let $\Upsilon = \Sigma\{0\}\{0,1\}^{p-r}$. Let $\Xi = \Upsilon\{0,1\}^{e}$. Let $\Upsilon' = \{\upsilon \in \Upsilon : R_p(\upsilon, t) = \emptyset\}$. If $|\Upsilon'| < \frac{1}{4}|\Upsilon|$, then move to the spend phase.

For each $\upsilon \in \Upsilon'$, choose a $\xi_\upsilon \in \Xi$ such that $\xi_\upsilon \succeq \upsilon$ and $a(\xi_\upsilon, t) = 0$. Let $\Xi' = \{\xi_\upsilon : \upsilon \in \Upsilon'\}$. Choose the least $i \in \{0, \ldots, n-1\}$ that has not been used previously. Let $\Psi_i = \Xi'\{0,1\}^{r^*_{n-i-1}-q}$ and implement an (r^*_{n-i-1}, f, Ψ_i)-strategy. If this strategy finishes at stage t', then wait until stage t'' such that $K_{m,t''}(\upsilon) \leq p$ for all $\upsilon \in \Upsilon$ such that $a(\upsilon, t') \geq 2^{-p}$. Repeat the advantage phase.

Spend Phase. Let t be the current stage. Let $\Sigma' = \{\sigma \in \Sigma : a(\sigma, t) < 2^{-r}\}$. If Σ' is the empty set then terminate the algorithm. Otherwise choose the least $i \in \{0, \ldots, 7\}$ such that i has not been used previously in the spend phase. Let $\Phi_i = \Sigma'\{1\}\{0^i 1\}\{0,1\}^{r_{7-i}-r-3}$ and run a (r_{7-i}, f, Φ_i)-strategy. Repeat the spend phase.

4.4 Verification of algorithm

The algorithm starts at stage t_0. We will define the following parameters assuming that the algorithm does terminate at some stage t_1. Let n_a be the number of times the advantage phase is run. Let n_s be the number of times the spend phase is run. For all $0 \leq i < n_a$, let t^*_i be the stage the ith strategy in the advantage phase begins (starting with $i = 0$). Let t_{mid} be the stage when then the algorithm completes the advantage phase and let $t^*_{n_a} = t_{mid}$. For all $0 \leq i < n_s$, let t'_i be the stage the ith strategy in the spend phase begins (starting with $i = 0$). Let $t'_{n_s} = t_1$. Let $\Phi = \Phi_0 \cup \ldots \cup \Phi_{n_s-1}$ and $\Psi = \Psi_0 \cup \ldots \cup \Psi_{n_a-1}$.

The first step in the verification is to prove that the algorithm runs without error. To prove this, we need to show that each time we 'choose' something in the algorithm, there is something valid to choose. First we note that the opponent cannot add a description of σ of length p or less without making a reservation of length p.

Lemma 4.4.1. *If $t, p \in \omega$, and $\sigma \in 2^{<\omega}$, with $K_{m,t}(\sigma) \leq p$, then $R_p(\sigma, t) \neq \emptyset$.*

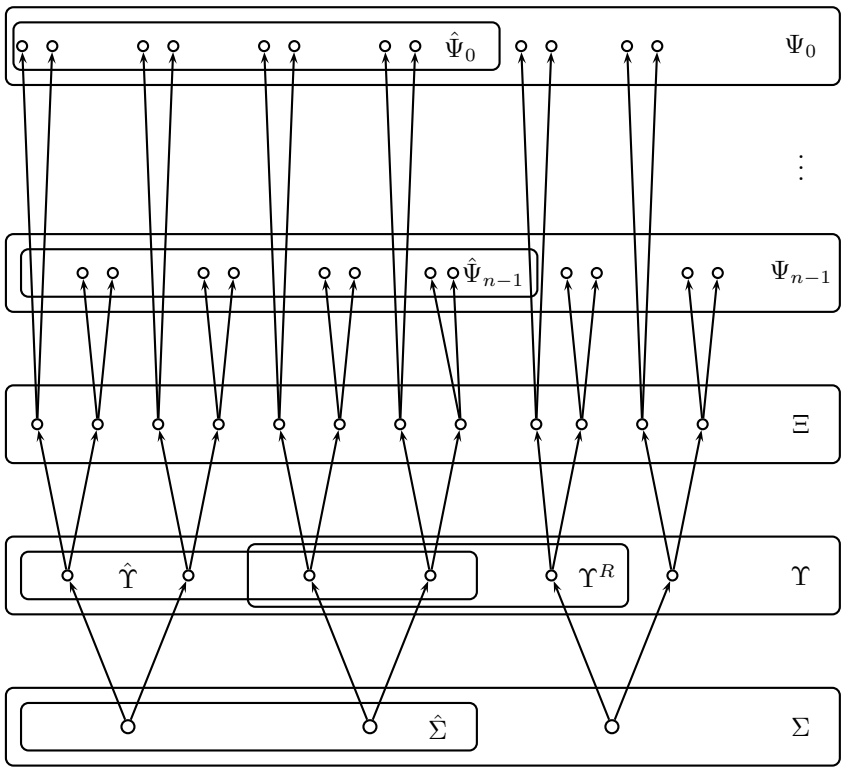

Figure 4.2: Algorithm strings

Proof. If $K_{m,t}(\sigma) \leq p$, then there exists $\langle \tau, \sigma' \rangle \in U_t$ such that $s = |\tau| \leq p$ and $\sigma' \succeq \sigma$. Consider $\tau' = \tau 0^{p-s}$. As $|\tau'| = p$, $\tau \preceq \tau'$ and $\sigma' \succeq \sigma$, so $\tau' \in F_p(\sigma, t)$. If $\tau' \in I_p(\sigma, t)$ then there would exist $\tau'' \approx \tau'$ and $\rho \mid \sigma$ such that $\langle \tau'', \rho \rangle \in U_t$. However, then $\tau'' \approx \tau$ and $\rho \mid \sigma'$ so this would contradict the definition of a monotone machine. Thus $\tau' \notin I_p(\sigma, t)$ and so $\tau' \in R_p(\sigma, t)$. □

Proposition 4.4.2. *The advantage phase of the algorithm never runs out of choices for ξ_υ or i, and the spend phase never runs out of choices for i.*

Proof. First we will show that there is always a choice for ξ_υ. Take any $\upsilon \in \Upsilon$. Take any $j \in \omega$ such that $0 \leq j < n_a$. If we assume that for any $\xi \in \Xi$ extending υ, we have $a(\xi, t_j^*) \neq 0$, then given any such ξ there is some $i < j$ such that an (r^*_{n-i-1}, f, Ψ_i)-strategy has been implemented with $\{\xi\}\{0,1\}^{r^*_{n-i-1}-q} \subseteq \Psi_i$. There are $2^{r^*_{n-i-1}-q}$ extensions of ξ in Ψ_i so

$$a(\xi, t_j^*) \geq 2^{r^*_{n-i-1}-q} 2^{-r^*_{n-i-1}} = 2^{-q}.$$

Hence $a(\upsilon, t_j^*) \geq 2^e 2^{-q} = 2^{-p}$, because there are 2^e extensions of υ in Ξ. This means that at the end of the iteration of the previous advantage phase, the algorithm will wait until a stage t when the opponent uses a string of length at most p to describe υ before running the advantage phase again. As this

implies that $R_p(\upsilon, t_j^*) \neq \emptyset$ by Lemma 4.4.1, it follows that no choice of ξ_υ will be needed.

The advantage phase stops if $|\Upsilon'| < \frac{1}{4}|\Upsilon|$. So each time the phase is run, at least $\frac{1}{4}|\Upsilon|$ elements of Ξ are selected. This can only happen $4 \cdot 2^e = 2^{e+2}$ times (otherwise the algorithm would run out of choices for some ξ_υ) so there is always a choice for i.

The spend phase will terminate if $a(\sigma, t_i') \geq 2^{-r}$ for all $\sigma \in \Sigma$. During each iteration through the phase, $a(\sigma, t_{i+1}') \geq a(\sigma, t_i') + 2^{-r-3}$ (if it is not greater than 2^{-r} already) so after a maximum of 8 steps, $a(\sigma, t) \geq 2^{-r}$ for all $\sigma \in \Sigma$ and so the algorithm will not run out of choices for i. □

The next step in the verification is to show that the required bounds on the measure that the algorithm uses are met.

Proposition 4.4.3. *For all $\sigma \in \Sigma$, if the algorithm terminates at stage t_1, then $2^{-r} \leq a(\sigma, t_1) < \frac{5}{4}2^{-r}$.*

Proof. As $a(\sigma, t_0) = 0$, any measure placed on σ by stage t_{mid} must be due to an increase on measure on some $\psi \in \Psi$ where $\psi \succeq \sigma$. Hence for any $\sigma \in \Sigma$, $a(\sigma, t_{mid}) = \sum_{\psi \in \Psi, \psi \succeq \sigma} a(\psi, t_{mid})$. If $\psi \in \Psi_i$, then we increase the measure on ψ by less than $\frac{5}{4}2^{-r^*_{n-i-1}}$. Now if $\psi \succeq \sigma$ then $|\psi| = |\sigma| + 1 + p - r + e + r^*_{n-i-1} - q = |\sigma| + 1 - r + r^*_{n-i-1}$ so $r^*_{n-i-1} = |\psi| - |\sigma| - 1 + r$. Thus $a(\psi, t_{mid}) < \frac{5}{4}2^{-(|\psi| - |\sigma| - 1 + r)}$.

All such ψ form a prefix free set above $\sigma 0$ so:

$$\begin{aligned} a(\sigma, t_{mid}) = a(\sigma 0, t_{mid}) &= \sum_{\psi \in \Psi, \psi \succeq \sigma 0} a(\psi, t_{mid}) \\ &< \sum_{\psi \in \Psi, \psi \succeq \sigma 0} \frac{5}{4} 2^{-|\psi| + |\sigma 0| - r} \\ &\leq \frac{5}{4} 2^{-r}. \end{aligned}$$

The spend phase will only run if $a(\sigma) < 2^{-r}$. Each iteration of the spend phase increases the measure on σ by less than $\frac{5}{4}2^{-r-3} < 2^{-r-2}$, so at then end of spend phase $2^{-r} \leq a(\sigma, t_1) < 2^{-r} + 2^{-r-2} = \frac{5}{4}2^{-r}$. □

Note that as $a(\Sigma, t_1) < |\Sigma|\frac{5}{4}2^{-r} \leq 1$, we do not run out of measure. There are two possible outcomes to a $(r, f + 1, \Sigma)$-strategy. These are related to whether or not the algorithm terminates.

Proposition 4.4.4. *If the algorithm does not terminate then outcome (i) is achieved.*

Proof. The number of times any phase is repeated is bounded so if the algorithm does not terminate, then this must be caused by waiting for a response

from the opponent or while waiting for another strategy to finish. Using the basic strategy as an inductive base case, this can only be caused if for some $\sigma \in \Sigma$, there is a $\sigma' \succeq \sigma$ with $K_m(\sigma') > -\log(a(\sigma'))$. Hence (i) (a) holds. Now (i) (b) holds by Proposition 4.4.3 because if the algorithm does not exceed the measure upper bound when it terminates, then it will not exceed the upper bound if it takes no action from some point onwards. □

Now for the difficult stage of the verification. Let $\hat{\Sigma} \subseteq \Sigma$. Let us examine what happens if the algorithm does terminate. We are going to look at the gain made during the advantage phase and the spend phase separately. This is because we know from Proposition 4.3.5, that the overall gain is at least the sum of the gain which occurs during these phases. That is:

$$\mu(G_r^{r'}(\hat{\Sigma}; t_0, t_1)) \geq \mu(G_p^{r'}(\hat{\Sigma}\{0\}; t_0, t_{mid})) + \mu(G_r^p(\hat{\Sigma}\{1\}; t_{mid}, t_1)).$$

We know what we can expect from the spend phase of the algorithm because this is just a series of (r, f, Φ_i)-strategies. However, in the advantage phase we have to show that the overall gain is increased by the fact that some reservations of length p are made i.e. for some $\upsilon \in \Upsilon$, $R_p(\upsilon, t_{mid}) \neq \emptyset$.

First let us look at the spend phase. For all $i \in \omega$, $0 \leq i < n_s$, let $\hat{\Phi}_i = \{\phi \in \Phi_i : \exists \sigma \in \hat{\Sigma}, \phi \succeq \sigma\}$. Note that $\hat{\Sigma}\{1\} \preceq \hat{\Phi}_i$. Let $\hat{\Phi} = \hat{\Phi}_0 \cup \ldots \cup \hat{\Phi}_{n_s-1}$.

Proposition 4.4.5. *If the algorithm terminates then:*

$$\mu(G_r^p(\hat{\Sigma}\{1\}; t_{mid}, t_1)) \geq \left(1 + \frac{f}{2}\right)\left(a(\Sigma\{1\}, t_1) - 2a(\Sigma\{1\} \setminus \hat{\Sigma}\{1\}, t_1)\right).$$

Proof. From Proposition 4.3.5 and the inductive hypothesis of the existence of (r, f, Σ)-strategies we know that:

$$\begin{aligned}
\mu(G_r^p(\hat{\Sigma}\{1\}; t_{mid}, t_1)) &\geq \sum_{i=0}^{n_s-1} \mu(G_{r_{7-i}}^{r_{8-i}}(\hat{\Phi}_i; t_i', t_{i+1}')) \\
&\geq \sum_{i=0}^{n_s-1} (1 + \frac{f}{2})(a(\Phi_i, t_{i+1}') - 2a(\Phi_i \setminus \hat{\Phi}_i, t_{i+1}')) \\
&= (1 + \frac{f}{2})(a(\Sigma\{1\}, t_1) - 2a(\Sigma\{1\} \setminus \hat{\Sigma}\{1\}, t_1)).
\end{aligned}$$

The second inequality is a consequence of condition (c) of outcome (ii). The final equality is due to the fact that we only increase measure on extensions of $\Sigma\{1\}$ during the spend phase, and that the sets $\Phi_0, \Phi_1, \ldots, \Phi_{n_s-1}$ are incomparable. □

Before we examine the advantage phase, we need some more definitions. We need to define those subsets of Υ that have p reservations, or extend elements of $\hat{\Sigma}$, or both.

(i). Let $\Upsilon^R = \{\upsilon \in \Upsilon : R_p(\upsilon, t_{mid}) \neq \emptyset\}$.

(ii). Let $\hat{\Upsilon} = \{\upsilon \in \Upsilon : \exists \sigma \in \hat{\Sigma}, \upsilon \succeq \sigma\}$.

(iii). Let $\hat{\Upsilon}^R = \Upsilon^R \cap \hat{\Upsilon}$.

For all $\upsilon \in \hat{\Upsilon}^R$, we choose a specific τ_υ in $R_p(\upsilon, t_{mid})$.

The gain made during the advantage phase can be broken down into two parts. Firstly, for every $\upsilon \in \hat{\Upsilon}^R$, $[\tau_\upsilon] \subseteq G_p^{r'}(\hat{\Sigma}\{0\}; t_0, t_{mid})$. This is true because $\tau_\upsilon \in R_p(\upsilon, t_{mid}) = F_p(\upsilon, t_{mid}) \setminus I_p(\upsilon, t_{mid})$. As $\upsilon \succeq \sigma 0$ for some $\sigma \in \hat{\Sigma}$ we have $\tau_\upsilon \in F_p(\sigma 0, t_{mid}) \subseteq F_p(\hat{\Sigma}\{0\}, t_{mid})$. Additionally, $\tau_\upsilon \notin I_p(\sigma 0, t_{mid})$, so $\tau_\upsilon \notin I_p(\hat{\Sigma}\{0\}, t_{mid}) \supseteq I_p(\hat{\Sigma}\{0\}, t_0)$. Now as $[I_p(\hat{\Sigma}\{0\}, t_0)] \supseteq [I_{r'}(\hat{\Sigma}\{0\}, t_0)]$ we have $[\tau_\upsilon] \cap [I_{r'}(\hat{\Sigma}\{0\}, t_0)] = \emptyset$. Thus:

$$[\tau_\upsilon] \subseteq [F_p(\hat{\Sigma}\{0\}, t_{mid})] \setminus [I_{r'}(\hat{\Sigma}\{0\}, t_0)] = G_p^{r'}(\hat{\Sigma}\{0\}; t_0, t_{mid}).$$

Secondly, we can count the gains made by the recursive use of the strategies in the advantage phase as:

$$\bigcup_{i=0}^{n_a-1} G_{r^*_{n-i-1}}^{r^*_{n-i}}(\hat{\Psi}_i; t_i^*, t_{i+1}^*) \subseteq G_p^{r'}(\hat{\Sigma}\{0\}; t_0, t_{mid}).$$

We would like to combine these to say:

$$\mu(G_p^{r'}(\hat{\Sigma}\{0\}; t_0, t_{mid})) \geq \mu(\bigcup_{\upsilon \in \hat{\Upsilon}^R} [\tau_\upsilon]) + \mu(\bigcup_{i=0}^{n_a-1} G_{r^*_{n-i-1}}^{r^*_{n-i}}(\hat{\Psi}_i; t_i^*, t_{i+1}^*)).$$

However we cannot do this as given υ and i, $[\tau_\upsilon]$ and $G_{r^*_{n-i-1}}^{r^*_{n-i}}(\hat{\Psi}_i; t_i^*, t_{i+1}^*)$ are not necessarily disjoint. Our goal is to find some subsets $\hat{\Psi}_i^R \subseteq \hat{\Psi}_i$, such that for all $\upsilon \in \hat{\Upsilon}^R$, $0 \leq i < n_a$ we have that $[\tau_\upsilon]$ and $G_{r^*_{n-i-1}}^{r^*_{n-i}}(\hat{\Psi}_i \setminus \hat{\Psi}_i^R, t_i^*, t_{i+1}^*)$ are disjoint. To this end we define $\hat{\Psi}_i^R = \{\psi \in \hat{\Psi}_i : \exists \upsilon, \tau_\upsilon \in F_p(\psi, t_{i+1}^*)\}$.

Lemma 4.4.6. *For all $\upsilon \in \hat{\Upsilon}^R$, $0 \leq i < n_a$ we have that $[\tau_\upsilon]$ and $G_{r^*_{n-i-1}}^{r^*_{n-i}}(\hat{\Psi}_i \setminus \hat{\Psi}_i^R, t_i^*, t_{i+1}^*)$ are disjoint.*

Proof. Assume that $\psi \in \hat{\Psi}_i \setminus \hat{\Psi}_i^R$. By definition it must be that $\tau_\upsilon \notin F_p(\psi, t_{i+1}^*)$ and hence $[\tau_\upsilon] \cap [F_{r^*_{n-i-1}}(\psi, t_{i+1}^*)] = \emptyset$. This implies that $[\tau_\upsilon] \cap G_{r^*_{n-i-1}}^{r^*_{n-i}}(\hat{\Psi}_i \setminus \hat{\Psi}_i^R, t_i^*, t_{i_1}^*) = \emptyset$. □

Our objective now is to figure out how much we lose by removing these sets $\hat{\Psi}_i^R$. To do this we want to move everything down to the level of the Υ strings. We define $\hat{\Upsilon}_i^R = \{\upsilon \in \hat{\Upsilon}^R : \tau_\upsilon \in F_p(\hat{\Psi}_i^R, t_{i+1}^*)\}$.

Lemma 4.4.7. *If $\tau_\upsilon \in F_p(\psi, t_{i+1}^*)$ then $\upsilon \preceq \psi$.*

Proof. If $\upsilon \not\preceq \psi$ then $\upsilon \mid \psi$ so $\tau_\upsilon \in I_p(\upsilon, t^*_{i+1})$ (Lemma 4.3.4) which means $\tau_\upsilon \notin R_p(\upsilon, t_{mid})$. □

Lemma 4.4.8. $|\hat{\Psi}_i^R| \leq 2^{r^*_{n-i-1}-q}|\hat{\Upsilon}_i^R|$.

Proof. If $\psi \in \hat{\Psi}_i^R$, then for some υ, $\tau_\upsilon \in F_p(\psi, t^*_{i+1})$. This implies that $\upsilon \preceq \psi$ and further $\upsilon \in \hat{\Upsilon}_i^R$. The result follows as for any $\upsilon \in \Upsilon$ there are at most $2^{r^*_{n-i-1}-q}$ extensions of τ_υ in Ψ_i. □

Lemma 4.4.9. *If $i \neq j$, then $\hat{\Upsilon}_i^R \cap \hat{\Upsilon}_j^R = \emptyset$.*

Proof. Assume that $i < j \leq n_a$ and $\upsilon \in \hat{\Upsilon}_i^R$. We have that $\tau_\upsilon \in F_p(\hat{\Psi}_i^R, t^*_{i+1})$. This means that for some $\psi \in \hat{\Psi}_i^R$, $\tau_\upsilon \in F_p(\psi, t^*_{i+1})$ which implies that $\upsilon \preceq \psi$ (Lemma 4.4.7) so $\tau_\upsilon \in F_p(\upsilon, t^*_{i+1}) \subseteq F_p(\upsilon, t^*_j)$.

Now as $\tau_\upsilon \in R_p(\upsilon, t_{mid})$ it must be that $\tau_\upsilon \notin I_p(\upsilon, t_{mid})$ which implies that $\tau_\upsilon \notin I_p(\upsilon, t^*_j)$. This gives us that $\tau_\upsilon \in R_p(\upsilon, t^*_j)$. Now during the advantage phase, our algorithm only chooses those elements of Υ that do not currently have a reservation of length p. As υ does have a reservation of length p at stage t^*_j, υ is not picked as so there is no extension of υ in $\Psi_j \supseteq \hat{\Psi}_j^R$. Hence by Lemma 4.4.7 we have that $\tau_\upsilon \notin F_p(\hat{\Psi}_j^R, t^*_{i+1})$. This means $\upsilon \notin \hat{\Upsilon}_j^R$. □

The final lemma that we need is a lower bound on the size of $|\hat{\Upsilon}^R|$.

Lemma 4.4.10. $|\hat{\Upsilon}^R| \geq (\frac{3}{4}|\Sigma| - |\Sigma \setminus \hat{\Sigma}|)2^{p-r}$.

Proof. Note that $|\Upsilon^R| \geq \frac{3}{4}|\Upsilon| = \frac{3}{4}|\Sigma|2^{p-r}$ as this is the condition for the algorithm to move to the spend phase. Additionally, $|\Upsilon \setminus \hat{\Upsilon}| = |\Sigma \setminus \hat{\Sigma}|2^{p-r}$. Hence:

$$\begin{aligned}
|\hat{\Upsilon}^R| &= |\Upsilon^R| + |\hat{\Upsilon}| - |\Upsilon^R \cup \hat{\Upsilon}| \\
&\geq |\Upsilon^R| + |\hat{\Upsilon}| - |\Upsilon| \\
&= |\Upsilon^R| - |\Upsilon \setminus \hat{\Upsilon}| \\
&\geq \frac{3}{4}|\Sigma|2^{p-r} - |\Sigma \setminus \hat{\Sigma}|2^{p-r} \\
&= (\frac{3}{4}|\Sigma| - |\Sigma \setminus \hat{\Sigma}|)2^{p-r}.
\end{aligned}$$

□

Proposition 4.4.11. *If the algorithm terminates and if $\hat{\Sigma} \subseteq \Sigma$, then:*

$$\begin{aligned}
\mu(G_p^{r'}(\hat{\Sigma}\{0\}; t_0, t_{mid})) \geq \left(1 + \frac{f}{2}\right)\Big(a(\Sigma\{0\}, t_1) - 2a(\Sigma\{0\} \setminus \hat{\Sigma}\{0\}, t_1)\Big) \\
+ \frac{5}{8}|\Sigma|2^{-r} - |\Sigma \setminus \hat{\Sigma}|2^{-r}.
\end{aligned}$$

Proof. Lemma 4.4.6 shows that the sets $[\tau_\upsilon]$ and $G^{r^*_{n-1+1}}_{r^*_{n-i}}(\hat{\Psi}_i \setminus \hat{\Psi}^R_i; t_i, t_{i+1})$ are disjoint. So:

$$\mu(G^{r'}_p(\hat{\Sigma}\{0\}; t_0, t_{mid})) \geq \sum_{\upsilon \in \hat{\Upsilon}^R} \mu([\tau_\upsilon]) + \sum_{i=0}^{n_a-1} \mu(G^{r^*_{n-i}}_{r^*_{n-i-1}}(\hat{\Psi}_i \setminus \hat{\Psi}^R_i; t^*_i, t^*_{i+1})).$$

This is the point that we use part (c) of outcome (ii). We know that the amount of gain that occurs on $\hat{\Psi}^R_i$ is bounded. So if we subtract it from $\hat{\Psi}_i$, we can determine the maximum amount we can lose. Note that as $\hat{\Psi}_i \subseteq \Psi_i$, it follows that: $\Psi_i \setminus (\hat{\Psi}_i \setminus \hat{\Psi}^R_i)$ is the disjoint union of $\Psi_i \setminus \hat{\Psi}_i$ and $\hat{\Psi}^R_i$. With this identity we can determine that:

$$\begin{aligned}
&\mu(G^{r^*_{n-i}}_{r^*_{n-i-1}}(\hat{\Psi}_i \setminus \hat{\Psi}^R_i; t^*_i, t^*_{i+1})) \\
\geq &(1 + \frac{f}{2})(a(\Psi_i, t^*_{i+1}) - 2a(\Psi_i \setminus (\hat{\Psi}_i \setminus \hat{\Psi}^R_i), t^*_{i+1})) \\
= &(1 + \frac{f}{2})(a(\Psi_i, t^*_{i+1}) - 2a(\Psi_i \setminus \hat{\Psi}_i, t^*_{i+1}) - 2a(\hat{\Psi}^R_i, t^*_{i+1})).
\end{aligned}$$

The first inequality comes from part (c) of outcome (ii) applied to our (r, f, Ψ_i)-strategies. Hence we can combine these results along with the facts that $a(\Psi_i, t^*_{i+1}) = a(\Psi_i, t_1)$ and $\sum_{\upsilon \in \hat{\Upsilon}^R} \mu([\tau_\upsilon]) = |\hat{\Upsilon}^R| 2^{-p}$ to get that:

$$\begin{aligned}
\mu(G^p_r(\hat{\Sigma}\{0\}; t_0, t_{mid})) \geq &|\hat{\Upsilon}^R| 2^{-p} + \sum_{i=0}^{n_a-1} (1 + \frac{f}{2}) a(\Psi_i, t_1) \\
&- \sum_{i=0}^{n_a-1} (1 + \frac{f}{2}) 2a(\Psi_i \setminus \hat{\Psi}_i, t_1) \\
&- \sum_{i=0}^{n_a-1} (1 + \frac{f}{2}) 2a(\hat{\Psi}^R_i, t_1). \qquad (4.4.1)
\end{aligned}$$

These terms can be simplified. First:

$$\begin{aligned}
&\sum_{i=0}^{n_a-1} (1 + \frac{f}{2}) a(\Psi_i, t_1) - \sum_{i=0}^{n_a-1} (1 + \frac{f}{2}) 2a(\Psi_i \setminus \hat{\Psi}_i, t_1) \\
= &(1 + \frac{f}{2})(a(\Sigma\{0\}, t_1) - 2a(\Sigma\{0\} \setminus \hat{\Sigma}\{0\}, t_1)). \qquad (4.4.2)
\end{aligned}$$

Further $a(\hat{\Psi}^R_i, t_1) < \frac{5}{4} 2^{r^*_{n-i-1}} |\hat{\Psi}^R_i|$ by condition (b) of outcome (ii). Hence:

$$\begin{aligned}
\sum_{i=0}^{n_a-1} (1 + \frac{f}{2}) 2a(\hat{\Psi}^R_i, t_1) &\leq \sum_{i=0}^{n_a-1} (1 + \frac{f}{2}) 2 \cdot \frac{5}{4} \cdot 2^{r^*_{n-i-1}} |\hat{\Psi}^R_i| \\
&\leq \frac{5}{2}(1 + \frac{f}{2}) \sum_{i=0}^{n_a-1} |\hat{\Upsilon}^R_i| 2^{-q} \\
&\leq \frac{5}{2}(1 + \frac{f}{2}) |\hat{\Upsilon}^R| 2^{-q}. \qquad (4.4.3)
\end{aligned}$$

The second and third inequalities are consequences of Lemmas 4.4.8 and 4.4.9 respectively. Putting (4.4.1), (4.4.2) and (4.4.3) gives that:

$$\begin{aligned}\mu(G_p^{r'}(\hat{\Sigma}\{0\}; t_0, t_{mid})) \;\geq\; & (1+\frac{f}{2})(a(\Sigma\{0\}, t_1) - 2a(\Sigma\{0\} \setminus \hat{\Sigma}\{0\}, t_1)) \\ & +|\hat{\Upsilon^R}|2^{-p} - \frac{5}{2}(1+\frac{f}{2})|\hat{\Upsilon}^R|2^{-q}. \end{aligned} \tag{4.4.4}$$

Now because $|\hat{\Upsilon}^R| \geq (\frac{3}{4}|\Sigma| - |\Sigma \setminus \hat{\Sigma}|)2^{p-r}$ (Lemma 4.4.10), $q = p + e$, and by choice of e, $\frac{5}{2}2^{-e}(1+\frac{f}{2}) \leq \frac{1}{8}$, it follows that:

$$\begin{aligned} |\hat{\Upsilon^R}|2^{-p} - \frac{5}{2}(1+\frac{f}{2})|\hat{\Upsilon}^R|2^{-q} &= |\hat{\Upsilon^R}|2^{-p}(1 - \frac{5}{2}2^{-e}(1+\frac{f}{2})) \\ &\geq |\hat{\Upsilon^R}|2^{-p}\frac{7}{8} \\ &\geq (\frac{3}{4}|\Sigma| - |\Sigma \setminus \hat{\Sigma}|)2^{p-r}2^{-p}\frac{7}{8} \\ &\geq \frac{5}{8}|\Sigma|2^{-r} - |\Sigma \setminus \hat{\Sigma}|2^{-r}. \end{aligned} \tag{4.4.5}$$

The proposition follows from (4.4.4) and (4.4.5). □

Proposition 4.4.12. *If the algorithm terminates then:*

$$\mu(G_r^{r'}(\hat{\Sigma}; t_0, t_1)) \geq (1 + \frac{f+1}{2})(a(\Sigma, t_1) - 2a(\Sigma \setminus \hat{\Sigma}, t_1)).$$

Proof. Proposition 4.3.5 tells us that:

$$\mu(G_r^{r'}(\hat{\Sigma}; t_0, t_1)) \geq \mu(G_p^{r'}(\hat{\Sigma}\{0\}; t_0, t_{mid})) + \mu(G_r^p(\hat{\Sigma}\{1\}; t_{mid}, t_1)). \tag{4.4.6}$$

All we need to do now is to combine and simplify Propositions 4.4.5 and 4.4.11. We have the following terms to deal with: $(1+\frac{f}{2})a(\Sigma\{1\}, t_1)$, $-(1+\frac{f}{2})2a(\Sigma\{1\} \setminus \hat{\Sigma}\{1\}, t_1)$, $(1+\frac{f}{2})a(\Sigma\{0\}, t_1)$, $-(1+\frac{f}{2})2a(\Sigma\{0\} \setminus \hat{\Sigma}\{0\}, t_1)$, $\frac{5}{8}|\Sigma|2^{-r}$ and $-|\Sigma \setminus \hat{\Sigma}|2^{-r}$.

Firstly, grouping the positive terms gives that:

$$\begin{aligned} & (1+\frac{f}{2})a(\Sigma\{1\}, t_1) + (1+\frac{1}{2})a(\Sigma\{0\}, t_1) + \frac{5}{8}|\Sigma|2^{-r} \\ =& (1+\frac{f}{2})a(\Sigma, t_1) + \frac{1}{2}\frac{5}{4}|\Sigma|2^{-r} \\ >& (1+\frac{f}{2})a(\Sigma, t_1) + \frac{1}{2}a(\Sigma, t_1) \\ =& (1+\frac{f+1}{2})a(\Sigma, t_1). \end{aligned} \tag{4.4.7}$$

Secondly, the negative terms give that:

$$\begin{aligned}
&(1+\frac{f}{2})2a(\Sigma\{1\}\setminus\hat{\Sigma}\{1\},t_1)+(1+\frac{f}{2})2a(\Sigma\{0\}\setminus\hat{\Sigma}\{0\},t_1)+|\Sigma\setminus\hat{\Sigma}|2^{-r}\\
=&(1+\frac{f}{2},t_1)2a(\Sigma\setminus\hat{\Sigma},t_1)+|\Sigma\setminus\hat{\Sigma}|2^{-r}\\
\leq&(1+\frac{f}{2})2a(\Sigma\setminus\hat{\Sigma},t_1)+a(\Sigma\setminus\hat{\Sigma},t_1)\\
=&(1+\frac{f+1}{2})2a(\Sigma\setminus\hat{\Sigma},t_1). \qquad (4.4.8)
\end{aligned}$$

Both inequalities in the above arguments follow from Proposition 4.4.3. The lemma follows by combining (4.4.6), (4.4.7) and (4.4.8) along with Propositions 4.4.5 and 4.4.11. □

Proposition 4.4.13. *If the algorithm terminates then outcome (ii) occurs.*

Proof. By Proposition 4.4.3 condition (ii) (a) holds. By Proposition 4.4.12 condition (ii) (c) holds. This implies condition (ii) (b) by taking $\hat{\Sigma}$ to be Σ. □

Hence we have established the existence of an $(r, f+1, \Sigma)$-strategy. By induction we can assert the existence of an $(r, f+1, \Sigma)$-strategy for any $f, r \in \omega$ such that $|\Sigma|\frac{5}{4}2^{-r} < 1$.

Proposition 4.4.14. *If $r \geq 1$ and $f2^{-r-1} \geq 1$, then for any $\sigma \in 2^{<\omega}$ such that $a(\sigma, t_0) = 0$, an $(r, f, \{\sigma\})$-strategy achieves outcome (i).*

Proof. Because $|\{\sigma\}|\frac{5}{4}2^{-r} < 1$ we have sufficient measure to implement such a strategy. However, if the algorithm terminates at some stage t_1, then:

$$\mu(G_r^{r'}(\{\sigma\}; t_0, t_1)) \geq (1+\frac{f}{2})a(\sigma, t_1) \geq (1+\frac{f}{2})2^{-r} = 2^{-r} + f2^{-r-1} > 1.$$

This is impossible so outcome (i) must occur. □

4.4.1 Construction of the semimeasure a

We can run a countable number of winning strategies in our construction of a. To do so, at stage s we find the least prime p that divides s. If p is the ith prime then we run one step of the ith strategy.

Proposition 4.4.15. *There is a c.e. semimeasure a such that for all i, $a(0^i1) \leq 2^{-2i-1}$ and there exists a σ_i such that $K_m(0^i1\sigma_i) > -\log(a(0^i\sigma_i))$.*

Proof. The semimeasure a can be constructed by for all i running a $(2i+2, 2^{2i+3}, \{0^i1\})$-strategy. Then for all i, $a(0^i1) < \frac{5}{4}2^{-2i-2} < 2^{-2i-1}$. Also by Proposition 4.4.14 because $2^{2i+3}2^{-2i-2-1} = 1$, the strategy achieves outcome (i) and hence there is some σ_i such that $K_m(0^i1\sigma_i) > -\log(a(0^i1\sigma_i))$. □

Now Theorem 4.1.1 holds by Proposition 4.1.5 and Proposition 4.4.15.

4.5 A lower bound on the difference

To determine a lower bound on the difference between K_m and KM, we need to determine the maximum string length used by an (r, f, Σ)-strategy. We will approach this by determining an upper bound for the number of bits appended to a string in Σ by the strategy. If $f = 0$, then the strategy does not use any extensions of strings in Σ so the number of extra bits appended is 0. If $f > 0$, an upper bound is the number of bits needed to extend a string in Σ to a string in Ψ_0. This requires $1+r^*_{n-1}-r = 1+r'-r-S(f-1) = 1+S(f)-S(f-1)$ bits (the extra 1 corresponds to the use of a 0 or 1 to distinguish between the advantage and spend phases).

Now an upper bound on the maximum string length used can be obtained by taking the maximum length l of any string in Σ and adding the upper bound for the number of bits added by an (r, i, Σ) strategy for all i, $1 \leq i \leq f$. Hence we get that the maximum possible string length is:

$$l + \sum_{i=1}^{f}(1 + S(i) - S(i-1)) = l + f + S(f) - S(0) = l + f + S(f).$$

Lemma 4.5.1. *If n is sufficiently large, then $S(n+1) \leq 2^{2n \log n}$.*

Proof. Unraveling the definition of $S(n)$ gives that: $S(0) = 0$, and $S(n+1) = (8 + 2^{e+2})S(n) + e + 3$ where $e = \lceil \log(20 + 10n) \rceil$. So for all $n \in \omega$,

$$S(n+1) \leq (2^3(20 + 10n))S(n) + \log(20 + 10n) + 4.$$

So there is some N such that for all $n \geq N$, $S(n+1) < 81nS(n)$. Hence for such n,

$$S(n+1) < S(N) \cdot 81^{n-N} \frac{n!}{(N-1)!} < (81S(N))^n n^n.$$

Thus the lemma holds if $n \geq 81S(N)$ as in such cases $S(n+1) < n^{2n} = 2^{2n \log n}$. □

We will now prove our main theorem.

Proof of Theorem 4.1.4. To prove this, we need to generalise Proposition 4.1.5 a little. Fix any $d \in \omega$, the series $\sum_{i \in \omega} 2^{-\frac{i}{d}}$ converges by the ratio test so we can take $k \in \omega$ to be such that 2^k is larger than the sum of this series. Now we can construct a semimeasure a such that for all $i \in \omega$, $a(0^i 1) \leq 2^{-(1+\frac{1}{d})i-k}$. We construct a by for all $j \in \omega$, running a $(j(d+1)+k+1, 2^{j(d+1)+k+2}, \{0^{jd}1\})$-

strategy. We do not exceed the bound on the measure for a because:

$$\begin{aligned} a(0^{jd}1) &< \frac{5}{4}2^{-j(d+1)-k-1} \\ &< 2^{-j(d+1)-k} \\ &= 2^{-(1+\frac{1}{d})jd-k}. \end{aligned}$$

As $f2^{-r-1} = 2^{j(d+1)+k+2}2^{-j(d+1)-k-2} = 1$, the strategy achieves outcome (i). Thus for all $j \in \omega$, there is some $\sigma_j \succeq 0^{jd}1$, such that $K_m(\sigma_j) > -\log a(\sigma_j)$. We can apply the same scaling to a, to construct a semimeasure m i.e. for all $\sigma \in 2^{<\omega}$, $m(0^i\sigma) = 2^i a(0^i\sigma)$. Again:

$$\begin{aligned} m(\lambda) &= \sum_{i\in\omega} m(0^i1) \\ &= \sum_{i\in\omega} 2^i a(0^i1) \\ &\leq \sum_{i\in\omega} 2^{-\frac{i}{d}-k} \\ &= 2^{-k}\sum_{i\in\omega} 2^{-\frac{i}{d}} \leq 1. \end{aligned}$$

We now get that for all $j \in \omega$, there is some $\sigma_j \succeq 0^{jd}1$ such that $K_m(\sigma_j) > -\log a(\sigma_j) = -\log(2^{-jd}m(\sigma_j)) = -\log m(\sigma_j) + jd$. By the discussion at the start of this section, $|\sigma_j| \leq |0^{jd}1| + S(n) + n$ where in this case $n = 2^{j(d+1)+k+2}$. If j is large enough, then by Lemma 4.5.1, the maximum string used by this strategy is less than:

$$jd + 1 + 2^{2(2^{j(d+1)+k+2}(j(d+1)+k+2))}.$$

Again if j is large enough this term is less than $2^{2^{j(d+2)}}$. Hence for infinitely many j, there exists a string σ_j such that $|\sigma_j| < 2^{2^{j(d+2)}}$ or $\log\log|\sigma_j| < j(d+2)$. As M_U majorizes all c.e. continuous semimeasures, there must be some constant b_d such that $K_m(\sigma_j) - KM(\sigma_j) > jd - b_d$. So:

$$\begin{aligned} K_m(\sigma_j) - KM(\sigma_j) &> jd\frac{j(d+2)}{j(d+2)} - b_d \\ &> \frac{d}{(d+2)}\log\log|\sigma_j| - b_d. \end{aligned}$$

Hence for any $c < 1$, there is a d such that $\frac{d}{(d+2)} > c + \epsilon$ for some $\epsilon \in \mathbb{R}^{>0}$. Now for sufficiently large j it must be that $\epsilon\log\log|\sigma_j| > b_d$. Hence:

$$\begin{aligned} K_m(\sigma_j) - KM(\sigma_j) &> (c+\epsilon)\log\log|\sigma_j| - b_d \\ &> c\log\log|\sigma_j|. \end{aligned}$$

□

While it has been significantly reduced, there is still a gap between the best known upper bound on the difference between KM and K_m, and this new lower bound. Hence there is further work that could be done lowering the upper bound, or potentially improving on the approach presented here.

Chapter 5

The Computational Power of Random Strings

(The results of this chapter appeared in the paper: On the computational power of random strings, Annals of Pure and Applied Logic, Vol 160 (2009), no. 2, pp. 214 – 228.)

5.1 Overview

Any variety of Kolmogorov complexity gives rise to two fundamental, computably enumerable sets.

Definition 5.1.1. Let Q be a standard complexity measure, e.g. C, K, K_{M_D}, K_{M_S}, K_m, or KM, and U an optimal machine for that complexity measure, then:

(i). The *set of non-random strings* is $\bar{R}_Q^U = \{\sigma \in 2^{<\omega} : Q^U(\sigma) < |\sigma|\}$.

(ii). The *overgraph* O_Q^U is $\{\langle \sigma, n\rangle \in 2^{<\omega} \times \omega : Q^U(\sigma) \leq n\}$.

The focus of this chapter is to investigate the computational power of these sets. This topic has seen significant recent interest with papers by Kummer [50], Muchnik and Positselsky [66], and Allender and co-authors [1, 2], which have looked at the power of these sets both in terms of computability theory and complexity theory.

The thesis that randomness can be used as a resource to enable efficient computation has been under intensive development in recent years by the computer science community. As an illustration, Allender, Buhrman, Koucký, van Melkebeek and Ronneburger have shown that sets related to these are complete for several complexity classes under probabilistic and non-uniform reductions [2].

If r is a reducibility, a computably enumerable set A is said to be *r-complete* if for any computably enumerable set W, $W \leq_r A$. A simple way to show that a c.e. set A is r-complete is to show that $\emptyset' \leq_r A$ where $\emptyset'$ is the halting problem. Of course completeness with respect to a more restrictive or *stronger* reducibility implies greater computational power.

For any complexity measure, both the set of non-random strings and the overgraph are easily seen to be weak truth-table complete [50]. The question is whether they are computationally stronger than this. This relatively long-standing question was answered by Kummer for plain Kolmogorov complexity [50].

Theorem 5.1.2 (Kummer)**.** *If U is any optimal Turing machine, then $\bar{R}^U_C$ is tt-complete.*

Kummer's proof is interesting because it is non-uniform, and uses conjunctive queries that grow exponentially in size.[1] The computational power of the set of non-random strings and the overgraph has also been examined for prefix-free complexity. Muchnik established the following surprising result [66].

Theorem 5.1.3 (Muchnik)**.** *There exist universal prefix-free machines U and V such that O^U_K is tt-complete and O^V_K is not tt-complete.*

Muchnik's result left open the question of whether there existed an optimal prefix-free machine for which the set of non-random strings is tt-complete. Allender, Buhrman and Koucký resolved this question [1].

Theorem 5.1.4 (Allender, Buhrman and Koucký)**.** *There exists a universal prefix-free machine U such that $\bar{R}^U_K$ is tt-complete.*

The technique used in the proof of Theorem 5.1.4 can be easily adapted to construct universal machines with tt-complete sets of non-random strings for the following classes of machines:

- Prefix-free machines.
- Strict process machines.
- Process machines.

[1]The reader may wonder if these sets are complete under even more powerful reducibilities such as $\leq_m$, $\leq_{bT}$, or $\leq_{btt}$, where for example $\leq_{bT}$ is a Turing reducibility that is only allowed to ask a fixed number of queries of the oracle. The answer is no as Muchnik proved that the overgraph of any Kolmogorov complexity function is not bT-complete [66]. However, the question for polynomial reducibilities is still open (see [1]).

- Monotone machines (for both K_m and KM complexities).

This chapter continues this work. In Section 5.2, the overgraphs of other types of universal machines are investigated. This section proves the following theorem.

Theorem 5.1.5. *For any optimal monotone machine U, the overgraph $O^U_{K_m}$ is tt-complete via a reduction that is non-uniform in $\emptyset'$.*

The construction used in the proof of Theorem 5.1.5 can be generalised to obtain the following corollaries.

Corollary 5.1.6. *For any optimal monotone machine U the overgraph O^U_{KM} is truth-table complete.*

Corollary 5.1.7. *For any optimal process machine U, or any optimal strict process machine V, the overgraphs $O^U_{K_{M_D}}$ and $O^V_{K_{M_S}}$ are truth-table complete.*

Hence, of the varieties of Kolmogorov complexity considered in this chapter, it is only prefix-free complexity for which there is an optimal machine whose overgraph is not tt-complete. In Section 5.3 we will shift our attention to the set of non-random strings. We will prove the following theorem.

Theorem 5.1.8. *There exists a universal strict process machine V such that $\bar{R}^V_{K_{M_S}}$ is not tt-complete.*

An important initial stage in this proof is showing that there exists a universal strict process machine whose set of non-random strings is closed under extension.

5.2 The overgraph of optimal monotone machines

The goal for this section is to answer a question of Muchnik and Positselsky by showing that for any optimal monotone machine U, the overgraph is truth-table complete [66]. Let us fix an optimal monotone machine U.

In order to prove Theorem 5.1.5 we will build a monotone machine N that ensures $\emptyset' \leq_{tt} O^U_{K_m}$. To give this proof the widest possible applicability, we will require N to be a strict process machine. For our construction, we would like to know some constant c such that $K^U_m(\sigma) \leq C^N(\sigma) + c$ (where $K_m = K^U_m$). Note that in this equation we compare the monotone complexity K_m to the standard descriptive complexity of Section 2.1.

To achieve this we will uniformly construct a family of strict process machines $L^0, L^2, L^4, \ldots$. We will combine these machines to form N by defining

$N(0^k) = \lambda$ and $N(0^k 1\sigma) = L^{2k}(\sigma)$. As $K_m(\sigma) \leq C^N(\sigma) + c$, it follows that $K_m(\sigma) \leq C^{L^d}(\sigma) + \frac{d}{2} + 1 + c$ and so if $\frac{d}{2} \geq c + 1$, then $K_m(\sigma) \leq C^{L^d}(\sigma) + d$.

In effect, each machine L^d is guessing that its constant with respect to U is no more than d. As U is an optimal monotone machine, for some machine L^d, the guess will be correct.

From now on let us just fix some d and refer to L^d as L. In addition to L, we need to build a corresponding truth-table reduction Γ. The reduction Γ will work if the following inequality holds:

$$K_m(\sigma) \leq C^L(\sigma) + d. \tag{5.2.1}$$

To avoid excessive superscripts, we will write $\Gamma(Z; x)$ for $\Gamma^Z(x)$.

For this proof we will omit the K_m subscript and write O^L for $O^L_{K_m}$. This allows us to reuse the subscript position to define:

$$O^L_k = \{\sigma \in 2^{<\omega} : \langle \sigma, k\rangle \in O^L\} = \{\sigma : C^L(\sigma) \leq k\}.$$

In this proof, we will consider the strict process machine L that we are building as both a partial computable function $L : 2^{<\omega} \to 2^{<\omega}$, and a c.e. set of ordered pairs where $\langle \tau, \sigma\rangle \in L$ if and only if $L(\tau) = \sigma$. We will use L_s to be the sth stage in the enumeration of L. Further we will also consider L as defining a c.e. semimeasure M_L, where $M_L([\sigma]) = \mu[\{\tau : \langle \tau, \sigma'\rangle \in L \text{ and } \sigma \preceq \sigma'\}]$. Finally M_{L_s} is the semimeasure obtained by using L_s instead of L.

The truth-table reduction that we will construct will work as follows. For each $x \in \omega$, a set of strings S_x will be specified. The reduction will determine which of these strings are in O^U_{d+x} and make a decision as to whether or not $x \in \emptyset'$ based on this information.

The simplest thing to do would be to try and encode $x \in \emptyset'$ by adding all such strings to O^U_{d+x} i.e. making $S_x \subseteq O^U_{d+x}$. However, this will not work because if we consider an opponent controlling both the optimal machine and $\emptyset'$, then the opponent could wait until S_x was defined, then add it to O^U_{d+x} and withhold x from $\emptyset'$. In fact, given any truth-table reduction Γ, the opponent could choose an x, wait until the truth-table used by $\Gamma(x)$ is defined, and then adopt a winning strategy to ensure either $\Gamma(O^U; x) = 0$ or $\Gamma(O^U; x) = 1$. By adding x to $\emptyset'$ in the first case and keeping it out in the second case, the opponent could ensure $\Gamma(O^U; x) \neq \emptyset'(x)$.

To overcome this problem, we make the reduction non-uniform by allowing it to be wrong on some initial segment of $\emptyset'$. The reduction will be constructed in such a way that the cost to the opponent of making the reduction incorrect for any x is so significant, that the reduction can only be incorrect a finite number of times.

The machine L we use for adding pairs to the overgraph must be a strict process machine. L will be constructed as follows. For each $\tau \in \{0,1\}^x$ we will choose some σ_τ, such that $\{\sigma_\tau : \tau \in \{0,1\}^x\}$ is a prefix-free set. The pairs $\langle \tau, \sigma_\tau \rangle$ are candidates for addition into our machine, and S_x will be defined as $\{\sigma_\tau : \tau \in \{0,1\}^x\}$. Further we will make sure that if $\tau' \prec \tau$, then either:

(i). $\sigma_{\tau'} \prec \sigma_\tau$, or

(ii). If $\sigma_{\tau'} \not\prec \sigma_\tau$, then the pair $\langle \tau', \sigma_{\tau'} \rangle$ is never added to our machine.

If we decide to add $\langle \tau, \sigma_\tau \rangle$ to our machine, then assuming (5.2.1) holds, $K_m(\sigma_\tau) \leq |\tau| + d = x + d$ and hence $\sigma_\tau \in O^U_{d+x}$.

Now our opponent has the ability to add σ_τ to O^U_{d+x} as well. If the opponent does this, then the opponent must have added some pair $\langle \rho, \sigma \rangle$ with $\sigma \succeq \sigma_\tau$ into U with $|\rho| \leq d + x$ at a certain stage s. If we consider the c.e semimeasure defined by U, M_U, then this implies that $M_U([\sigma_\tau]) \geq 2^{-d-x}$. Now provided we have not described any extension of σ_τ with L_s, then $M_{L_s}([\sigma_\tau]) = 0$. We will show how under these conditions, we can 'bypass' the measure spent by the optimal machine on σ_τ. Bypassing measure is a key idea in this proof. If the opponent has spent 2^{-d-x} of measure on the string σ_τ, then this measure cannot be reassigned by the c.e. semimeasure M_U to any strings that are incomparable with σ_τ. Our strategy is to avoid using any extensions of σ_τ when we define S_y for some new y. If there is no extension of σ_τ in S_y, then the opponent cannot use the measure placed on σ_τ to add any element of S_y into O^U_{d+y} (i.e. to add elements of S_y to O^U_{d+y} the opponent will need to find descriptions that are incomparable with any descriptions of σ_τ). Hence if the opponent wants to affect $\Gamma(O^U; y)$, then the opponent must use additional measure. Because we have not described any extension of σ_τ with L_s, we have not committed any measure to σ_τ and so we have not lost any measure in this action. This is called bypassing measure because some of the opponent's measure has been left stranded on σ_τ.

Bypassing measure allows us to ensure the reduction works on all but a finite set. We wait until an appropriate stage s when we have some bound on the measure that we can bypass. When we define S_s, we ensure that for any $\upsilon \in \{0,1\}^s$, σ_υ does not extend σ_τ. Instead we will set σ_υ to extend some ρ_τ incomparable with σ_τ.

However, our opponent still has one last trick up its sleeve. Before it adds some string σ_τ to O^U_{d+x}, it will try to force us to enumerate some $\langle \tau', \sigma_{\tau'} \rangle$ into our machine L with $\sigma_{\tau'} \succeq \sigma_\tau$. This action would make us commit some measure of M_L to σ_τ and prevent us from bypassing the measure on σ_τ. Our strategy to deal with this is complicated and will be detailed in the proof. The basic

idea is that if the opponent prevents us bypassing the measure on σ_τ in this way, then either something has been added to $\emptyset'$, or the opponent has spent measure somewhere else. The following reduction is designed to ensure that if the opponent has spent measure somewhere else we can bypass this instead, and to limit the impact of adding elements to $\emptyset'$.

The Γ reduction

Γ will be defined as follows. First $\Gamma(0) = 0$. If $x \neq 0$, then at stage x in the construction, a set S_x will be defined. This set will have 2^x elements and will be indexed by $\{0,1\}^x$ so for all $\tau \in \{0,1\}^x$ there is a unique string $\sigma_\tau \in S_x$. To determine if $x \in \emptyset'$, Γ runs the construction until S_x is defined and then determines which elements of the set S_x are in O^U_{d+x}. If $S_x \subseteq O^U_{d+x}$, then $\Gamma(x) = 0$. Otherwise Γ lexicographically orders $\{0,1\}^x$ with $0 < 1$, and finds the lex least $\tau \in \{0,1\}^x$ such that either:

(i). Exactly one of σ_τ and $\sigma_{\bar{\tau}}$ are in O^U_{d+x}, in which case $\Gamma(x) = 1$; or

(ii). Neither σ_τ nor $\sigma_{\bar{\tau}}$ are in O^U_{d+x}, in which case $\Gamma(x) = 0$.

Where the string $\bar{\tau}$ is obtained from τ by setting all 0's in τ to 1 and all 1's to 0.

The reduction can be thought of as checking pairs in some order. For example consider S_3. First the reduction checks if σ_{000}, and σ_{111} are in O^U_{d+3}. If they are not both in then the reduction can give an answer immediately. If they are both in, then the reduction checks if σ_{001}, and σ_{110} are in and so on. This can be described by simply looking at the indices of the σ's involved e.g. first 000 and 111; then 001 and 110; then 010 and 101; and finally 011 and 100.

Let us see how this would work in practice. The optimal monotone machine U is defined by a c.e. set so we will take U_s to be the set obtained by enumerating U for s steps. We define $\emptyset'_s$ similarly. We will take O^{U_s} to be the overgraph of U_s and $O^{U_s}_k = \{\sigma \in 2^{<\omega} : \langle \sigma, k\rangle \in O^{U_s}\}$. Note that if $s < t$, then $O^{U_s} \subseteq O^{U_t}$ and $O^{U_s}_k \subseteq O^{U_t}_k$. We can regard the construction of the reduction as a game between us and the opponent each with the ability to add strings to $O^{U_s}_k$.

As an example consider a game around $\Gamma(3)$. Assume that S_3, the set of strings used by $\Gamma(3)$ has been defined. Further assume that at stage 0, $3 \notin \emptyset'_0$ and $S_3 \cap O^{U_0}_{d+3} = \emptyset$. This means that $\Gamma(O^{U_0}; 3) = 0 = \emptyset'_0(3)$. Now suppose that at some stage s_0, the opponent enumerates σ_{000} into $O^{U_{s_0}}_{d+3}$. This would cause $\Gamma(O^{U_{s_0}}; 3) = 1$. So we would add the pair $\langle 111, \sigma_{111}\rangle$ to our machine L at the following stage. If we assume that this description appears in the optimal machine at stage s_1, then $\Gamma(O^{U_{s_1}}; 3) = 0$. Now if at a later stage s_2, $3 \in \emptyset'_{s_2}$, then

we would add $\langle 001, \sigma_{001} \rangle$ to L. If at any stage now the opponent adds σ_{110} to O^U_{d+x} then we will respond by adding $\langle 010, \sigma_{010} \rangle$, to L.

Note that for a given x, $\Gamma(x)$ can be changed from 0 to 1, and back again by adding a single string of S_x to O^U_{d+x} (provided $S_x \not\subseteq O^U_{d+x}$). If at some stage s of the construction $\Gamma(O^{U_s}; x) = 0$, then there are two possible choices of string for changing the reduction to 1. While if $\Gamma(O^{U_s}; x) = 1$ there is only one possible string that can be enumerated into the overgraph to change $\Gamma(x)$ to 0 again. Also note that if $\langle \tau, \sigma_\tau \rangle$ is enumerated into L, then the only reason to enumerate $\langle \bar{\tau}, \sigma_{\bar{\tau}} \rangle$ into L would be because we want to keep L a strict process machine.

Now, if we get to a stage s where for some x, $S_x \subseteq O^{U_s}_{d+x}$ and $x \in \emptyset'_s$, then we no longer have any ability to change the reduction. At this point we give up making the reduction work for x, in fact we go further and give up trying to make the reduction work on any value below $s + 2$. We have a marker which points to a value after which the reduction works. We move the marker to point to $s + 1$ and call $s + 1$ a marker move stage. The reason that the marker cannot be moved infinitely often, is that now when we define S_{s+1}, we can do it in such a way as to avoid extending some of the strings that have been enumerated into the optimal machine by our opponent and thus bypassing some of our opponent's measure.

In looking for strings that have measure we can bypass, we do not just consider those strings in S_x. We consider all strings σ_τ, where τ can have any length, such that for all ρ that occur no later than τ in the search order of Γ, $\sigma_\rho \in O^{U_s}_{d+|\tau|}$(e.g. $\{000, 111, 001, 110\}$ all occur no later than 110 in the search order). From this set of strings S, we set T to be the set of indices describing the strings in S. We use T instead of S because it is easier to deal with. As $S_x \subseteq O^{U_s}_{d+x}$, we have that $S_x \subseteq S$ so $\{0,1\}^x \subseteq T$ and hence $\mu[T] = 1$. From the set T we consider the set of maximal strings under the $\preceq$ order and use these to form a prefix-free set $\hat{T}$.

We can find some lower bound on $\mu[\hat{T}]$. This is because given any length, there can be at most two strings of that length that are in T, not in $\hat{T}$ and are not covered by $[\hat{T}]$. Hence the difference between $\mu[T] = 1$ and $\mu[\hat{T}]$ can be bounded.

We define B to be those strings in $\hat{T}$ which index strings enumerated into the overgraph by the opponent and whose measure we can bypass. Again we can find a lower bound on $\mu[B]$ because nearly half the strings in $\hat{T}$ must be in B. The reason for this is twofold. First it will be shown that if $\tau \in \hat{T}$, then $\bar{\tau} \in \hat{T}$. Secondly, we are unlikely to add both $\langle \tau, \sigma_\tau \rangle$ and $\langle \bar{\tau}, \sigma_{\bar{\tau}} \rangle$ to our machine L. The only reason we would do this would be to maintain L as a strict process

machine. Say we added $\langle \tau, \sigma_\tau \rangle$ to L to keep L a strict process machine, then there exists some $\tau' \succ \tau$ with $\tau' \in \mathrm{dom}(L)$. Further as $\tau' \notin T$ (as $\tau \in \hat{T}$), it must be that we added $\langle \tau', \sigma_{\tau'} \rangle$ to L in order to encode some x entering $\emptyset'$. However, this scenario can only affect a certain number of elements of $\hat{T}$. This is what we will use to find a lower bound for $\mu[B]$.

For the verification of the proof, it is useful to formalise the 'order' that the reduction Γ uses the strings in S_x. This is done by defining a relation on $\{0,1\}^k$ as follows: $\tau_1 \leq_\Gamma \tau_2$ if the $\min(\tau_1, \bar{\tau_1}) \leq_{lex} \min(\tau_2, \bar{\tau_2})$ where the minimum is with respect to the lexicographical order. Further $\tau_1 <_\Gamma \tau_2$ is defined to hold if $\tau_1 \leq_\Gamma \tau_2$ but not $\tau_2 \leq_\Gamma \tau_1$.

One way to think of the relation $\leq_\Gamma$ is as follows. Partition $\{0,1\}^k$ into equivalence classes each with two elements. If $\tau \in \{0,1\}^k$, then the equivalence class of τ is $\{\tau, \bar{\tau}\}$. Lexicographically order these equivalence classes using the element of each equivalence class that starts with 0. Then $\tau_1 \leq_\Gamma \tau_2$ if and only if the equivalence class of τ_1 is lexicographically less than or equal to the equivalence class of τ_2.

Note that while $\leq_\Gamma$ is reflexive and transitive it is not a pre-order as anti-symmetry fails. However, if $\tau \leq_\Gamma \rho$ and $\rho \leq_\Gamma \tau$, then either $\rho = \tau$, or $\rho = \bar{\tau}$. The relation $\leq_\Gamma$ is total in the sense that for all $\tau, \rho \in \{0,1\}^k$, $\tau \leq_\Gamma \rho$ or $\rho \leq_\Gamma \tau$. Note that this implies that if $\tau \not\leq_\Gamma \rho$ then $\rho <_\Gamma \tau$. The following lemma will be used in the verification of the proof.

Lemma 5.2.1. *If $\tau_1, \tau_2, \upsilon_1, \upsilon_2 \in 2^{<\omega}$ with $|\tau_1| = |\tau_2| < |\upsilon_1| = |\upsilon_2|$ and $\tau_1 \prec \upsilon_1$, and $\tau_2 \prec \upsilon_2$, then $\tau_1 <_\Gamma \tau_2$ implies $\upsilon_1 <_\Gamma \upsilon_2$.*

Proof. If $\tau_1 <_\Gamma \tau_2$, then $\tau_1 \neq \lambda$ and so either τ_1 or $\bar{\tau_1}$ begins with 0. Without loss of generality let us assume that τ_1 starts with 0. If $\tau_1 <_\Gamma \tau_2$, then by definition we have that $\tau_1 <_{lex} \tau_2$ and $\tau_1 <_{lex} \bar{\tau_2}$. Now as $\upsilon_1 \succ \tau_1$ and $\upsilon_2 \succ \tau_2$ it follows that $\upsilon_1 <_{lex} \upsilon_2$ and $\upsilon_1 <_{lex} \bar{\upsilon_2}$. Hence $\min(\upsilon_1, \bar{\upsilon_1}) <_{lex} \min(\upsilon_2, \bar{\upsilon_2})$ and so $\upsilon_1 <_\Gamma \upsilon_2$. □

In the construction and verification that follow we will assume that $O^{L_s}_k \subseteq O^{U_s}_{d+k}$ and thus $\{\langle \sigma, d+n \rangle : \langle \sigma, n \rangle \in O^{L_s}\} \subseteq O^{U_s}$. The reason we can make this assumption is that after we add some pair $\langle \tau, \sigma \rangle$ to L_s, we can wait until a stage s' such that $\langle \sigma, |\tau| + d \rangle$ enters $O^{U_{s'}}$. If (5.2.1) holds then we know such a stage s' must occur. If (5.2.1) does not hold then we could be waiting forever. In this case the construction presented below may stall at the end of some stage s. If this occurs then we can still verify that L is a strict process machine and so N will be a strict process machine too.

Construction

At stage 0: Set $\sigma_\lambda = 0$ and $S_0 = \{\sigma_\lambda\}$. S_0 is only used to start the construction and will not be used by Γ as $\Gamma(Z;0) = 0$ for any oracle Z by definition. Let $L_0 = \{\langle \lambda, \sigma_\lambda \rangle\}$. Let $C_0 = \{0\}$. The set C_s is used to determine the position of the marker at stage s.

Stage $s+1$: Let c_s, the marker, be the largest element of C_s. First we need to define S_{s+1}. If $s+1 \neq c_s$, then $s+1$ is not a marker move stage. In this case, for all $\tau \in \{0,1\}^s$ choose four extensions $\sigma_{\tau 0}, \sigma_{\tau 1}, \rho_{\tau 0}, \rho_{\tau 1}$ of σ_τ that are pairwise incomparable, and not in $O^{U_s}_{d+s+1}$. This is possible because $O^{U_s}_{d+s+1}$ is finite. Let $S_{s+1} = \{\sigma_\tau : \tau \in \{0,1\}^{s+1}\}$.

If $c_s = s+1$ then $s+1$ is a marker move stage. The construction of S_{s+1} will be done in such a way as to avoid extending some σ_τ that have been added to $O^{U_s}_{d+|\tau|}$ by the opponent thus bypassing measure. The procedure for finding σ_τ to avoid is as follows. First for $k \in \omega$ with $1 \leq k \leq s$, set:

$$T^{s+1}_k = \{\tau \in \{0,1\}^k : \forall \tau' \in \{0,1\}^k, \text{ if } \tau' \leq_\Gamma \tau, \text{ then } \sigma_{\tau'} \in O^{U_s}_{d+k}\}.$$

T^{s+1}_k is defined this way because this is the order that the reduction Γ examines the strings in S_k. Set $T^{s+1} = \bigcup_{1 \leq k \leq s} T^{s+1}_k$. We want to work with a prefix-free set so let $\hat{T}^{s+1} = \{\tau \in T^{s+1} : \forall \tau' \succ \tau, \tau' \notin T^{s+1}\}$. Note that this is a set of maximal elements of T^{s+1}, while prefix-free sets are usually constructed using minimal elements.

Finally, to ensure that we can bypass descriptions, let $B^{s+1} = \{\tau \in \hat{T}^{s+1} : \forall \tau' \succeq \tau, \tau' \notin \mathrm{dom}(L_s)\}$. For all $\upsilon \in B^{s+1}$, let $\{\upsilon_0, \ldots, \upsilon_n\}$ be the set of extensions of υ of length $s+1$. Choose $\sigma_{\upsilon_0}, \rho_{\upsilon_0}, \ldots, \sigma_{\upsilon_n}, \rho_{\upsilon_n}$ that are pairwise incomparable, not in $O^{U_s}_{d+s+1}$ and all extend ρ_υ. For all $\tau \in \{0,1\}^{s+1}$ that do not extend some $\upsilon \in B^{s+1}$, choose a σ_τ and a ρ_τ which are incomparable, not in $O^{U_s}_{d+s+1}$ and extend $\sigma_{\tau'}$ where $\tau' = \tau \upharpoonright (|\tau| - 1)$. Again let $S_{s+1} = \{\sigma_\tau : \tau \in \{0,1\}^{s+1}\}$.

Secondly, we need to determine which descriptions to commit to our machine L. Let $X_s = \{x \in \omega : c_s < x < s \text{ and } \Gamma(O^{U_s}; x) \neq \emptyset'_s(x)\}$. If $X_s = \emptyset$, set $L_{s+1} = L_s$. If $X_s \neq \emptyset$, and for some $x \in X_s$ $S_x \subseteq O^{U_s}_{d+x}$, then set $C_{s+1} = C_s \cup \{s+2\}$ and set $L_{s+1} = L_s$. This will cause the marker to be moved at the next stage.

Otherwise let x_s be the least element of X_s, choose $\tau \in \{0,1\}^{x_s}$ such that $\sigma_\tau \notin O^{U_s}_{d+x_s}$ and for all $\tau' <_\Gamma \tau$, $\sigma_{\tau'} \in O^{U_s}_{d+x_s}$. We are going to add $\langle \tau, \sigma_\tau \rangle$ to L_{s+1}. However, we want to make L a strict process machine so we need to ensure that $\mathrm{dom}(L_{s+1})$ is closed under substrings. Let υ be the longest initial segment of τ such that $\upsilon \in \mathrm{dom}(L_s)$. Consider any τ' such that $\upsilon \prec \tau' \preceq \tau$. If $|\tau'| \leq c_s$, then we will set $L_{s+1}(\tau') = L_s(\upsilon)$. Otherwise if $|\tau'| > c_s$, we will set $L_{s+1}(\tau') = \sigma_{\tau'}$. We also set $C_{s+1} = C_s$ because the marker has not moved.

Verification

First we will show that L is a strict process machine. To do this, we need the following lemma.

Lemma 5.2.2. *If $\tau_1 \prec \tau_2$ and $L(\tau_1) = \sigma_{\tau_1}$, then $\sigma_{\tau_1} \prec \sigma_{\tau_2}$.*

Proof. Assume $\tau_1 \prec \tau_2$. If $\sigma_{\tau_1} \not\prec \sigma_{\tau_2}$, then there must be some marker move stage $s+1$ with $|\tau_2| \geq s+1 > |\tau_1|$ and $\sigma_{\tau_2} \succ \rho_{\tau_0}$ where $\tau_1 \succeq \tau_0$ and $\tau_0 \in B^{s+1}$. This implies that $\tau_1 \notin \text{dom}(L_s)$ by definition of B^{s+1}. However, this means that for all stages $t > s$, $L_t(\tau_1) \neq \sigma_{\tau_1}$ because once the marker has moved past $|\tau_1|$, if τ_1 is added to the domain of L_t, then $L_t(\tau_1) = \sigma_\upsilon$ for some $\upsilon \prec \tau_1$ so $L_t(\tau_1) \neq \sigma_{\tau_1}$. The result follows from the contrapositive. □

Lemma 5.2.3. *L is a strict process machine.*

Proof. To prove this we induct on the stages of the construction. Clearly L_0 is a strict process machine. Now if L_s is a strict process machine then the construction ensures that L_{s+1} is at least a function whose domain is closed downward under $\preceq$. This is because if $L_{s+1} \neq L_s$, then L_{s+1} is formed by taking, some $\tau \notin \text{dom}(L_s)$ and finding the longest $\upsilon \prec \tau$ such that $\upsilon \in \text{dom}(L_s)$. The strings that we add to the domain of L_{s+1} are exactly those strings τ' such that $\upsilon \prec \tau' \preceq \tau$.

We also need to show that L_{s+1} is a process. In the construction, L_{s+1} is defined to be: $L_s \cup \{\langle \tau', L_s(\upsilon)\rangle : \upsilon \prec \tau' \preceq \tau \text{ and } |\tau'| \leq c_s\} \cup \{\langle \tau', \sigma_{\tau'}\rangle : \upsilon \prec \tau' \preceq \tau \text{ and } |\tau'| > c_s\}$.

Let $P_1 = L_s \cup \{\langle \tau', L_s(\upsilon)\rangle : \upsilon \prec \tau' \preceq \tau \text{ and } |\tau'| \leq c_s\}$. P_1 is a process. Let $P_2 = \{\langle \tau', \sigma_{\tau'}\rangle : \upsilon \prec \tau' \preceq \tau \text{ and } |\tau'| > c_s\}$. P_2 is a process because for any τ_1, τ_2 with $\upsilon \preceq \tau_1 \prec \tau_2 \preceq \tau$, we have that $\sigma_{\tau_1} \prec \sigma_{\tau_2}$ since the marker has not been moved since σ_{τ_1} was defined.

To show that the union of these two processes is a process, consider any $\tau_1 \prec \tau_2$ with τ_1 in the domain of P_1 and τ_2 in the domain of P_2. If $\tau_1 \preceq \upsilon$ then $P_1(\tau_1) \preceq P_1(\upsilon)$. Otherwise $\upsilon \prec \tau_1$ and so $P_1(\tau_1) = P_1(\upsilon)$. Now by construction there must be some $\upsilon' \preceq \upsilon$ such that $L_s(\upsilon) = L_s(\upsilon') = \sigma_{\upsilon'}$. Hence as $\upsilon' \preceq \tau_2$, the previous lemma implies that $\sigma_{\upsilon'} \prec \sigma_{\tau_2}$. Hence $P_2(\tau_2) = \sigma_{\tau_2} \succ \sigma_{\upsilon'} = L_s(\upsilon) = P_1(\upsilon) \succeq P_1(\tau_1)$.

If (5.2.1) does not hold then the construction could stall at the end of some stage. In this case $L = L_s$ for some s and hence L is a strict process machine. If the construction does not stall then $L = \bigcup_{s\in\omega} L_s$. In this case L must be a strict process machine because otherwise for some s, L_s would fail to be a strict process machine. □

From now on we will assume that (5.2.1) holds.

Lemma 5.2.4. *If there are only a finite number of marker move stages, then for all but finitely many x, $\Gamma(O^U; x) = \emptyset'(x)$.*

Proof. If there are only a finite number of marker move stages, then let s_0 be the last marker move stage. Choose any $x_0 > s_0$. Let $s_1 + 1$ be a stage such that:

(i). $s_1 > x_0$.

(ii). $\emptyset'_{s_1} \upharpoonright (x_0 + 1) = \emptyset' \upharpoonright (x_0 + 1)$.

(iii). For all $x \leq x_0$, $S_x \cap O^U_{d+x} = S_x \cap O^{U_{s_1}}_{d+x}$.

This last condition implies that $\Gamma(O^{U_{s_1}}; x) = \Gamma(O^U; x)$ for all $x \leq x_0$. If $x_0 \in X_{s_1}$, then as the marker does not move again, there must be some x with $x \leq x_0$ such that for some $\tau \in \{0,1\}^x$ with $\sigma_\tau \notin O^{U_{s_1}}_{d+x}$, $\langle \tau, \sigma_\tau \rangle$ is added to L_{s_1+1}. But this would add σ_τ to O^U_{d+x}, a contradiction as $S_x \cap O^U_{d+x} = S_x \cap O^{U_{s_1}}_{d+x}$ for all $x \leq x_0$. Hence $x_0 \notin X_{s_1}$ and so $\Gamma(O^U; x_0) = \Gamma(O^{U_{s_1}}; x_0) = \emptyset'_{s_1}(x_0) = \emptyset'(x_0)$. □

Now it is necessary to show that the number of marker move stages is finite. The reason for this is that each time the marker is moved, a portion of the measure that the optimal machine has spent is bypassed by the construction, and can no longer be used to affect Γ. By showing that there is a lower bound on the amount of measure that is bypassed each time the marker is moved, it follows that the marker can only be moved a finite number of times otherwise the optimal machine will run out of measure. For any x there is a direct relation between the index of a string in S_x and the measure needed to add the string to O^U_{d+x}. Hence to determine a lower bound on the amount of measure bypassed, it is useful to find a lower bound on $\mu[B^s]$. The first step towards achieving this will be to find a lower bound on $\mu[\hat{T}^s]$.

For the rest of the verification, fix s to be a particular marker move stage. As s is fixed, T_k will be used for T^s_k.

Lemma 5.2.5. *For all $k \in \omega$ with $1 \leq k < s$, if $\tau, \rho \in \{0,1\}^k$ with $\tau \in T_k$ and $\rho \leq_\Gamma \tau$ then $\rho \in T_k$.*

Proof. If $\rho \notin T_k$, then for some $\upsilon \in \{0,1\}^k$ such that $\upsilon \leq_\Gamma \rho$, $\sigma_\upsilon \notin O^{U_s}_{d+k}$. However, by the transitivity of the $\leq_\Gamma$ relation, $\upsilon \leq_\Gamma \tau$ and so $\tau \notin T_k$. □

Note that this lemma implies that if $\tau \in T_k$, then $\bar{\tau} \in T_k$ as well.

Lemma 5.2.6. *For all $k, j \in \omega$, if $1 \leq k < j < s$, then $[T_k] \subseteq [T_j]$ or $[T_j] \subseteq [T_k]$.*

Proof. If $[T_j] \not\subseteq [T_k]$, then there is some $\upsilon \in T_j$ such that if $\tau = \upsilon \upharpoonright k$, $\tau \notin T_k$. Now if $\tau' \in T_k$, then $\tau' <_\Gamma \tau$ (because the relation is total and if $\tau \leq_\Gamma \tau'$ then by definition $\tau' \notin T_k$). Now let υ' be any extension of τ' such that $|\upsilon'| = j$. By Lemma 5.2.1, $\upsilon' <_\Gamma \upsilon$, and hence $\upsilon' \in T_j$ by Lemma 5.2.5. Thus $[T_k] \subseteq [T_j]$. □

Take x to be the maximum integer such that $S_x \subseteq O^{U_s}_{d+x}$. By the construction such an x exists, as this is the reason for a marker move stage. Additionally, x is greater than the previous marker move stage. From the previous lemma, there exists an increasing integer sequence $j(0) < j(1) < \cdots < j(n)$ such that $j(0) = x$ and $j(n) < s$ with $[T_{j(0)}] \supsetneq \ldots \supsetneq [T_{j(n)}]$ and for all $l \in \omega$, if $j(i) < l \leq j(i+1)$ for some i, then $[T_l] \subseteq [T_{j(i+1)}]$. We have that $j(n) < s$ because the set S_s is chosen to ensure that T^s_s is empty.

If $0 \leq i < n$, let $\hat{T}_{j(i)} = \{\tau \in T_{j(i)} : \forall \tau' \in T_{j(i+1)}, \tau' \not\succ \tau\}$. Let $\hat{T}_{j(n)} = T_{j(n)}$.

Lemma 5.2.7. *For all* $i \in \{0, 1, 2, \ldots, n-1\}$, $\mu[\hat{T}_{j(i)}] \geq \mu([T_{j(i)}] \backslash [T_{j(i+1)}]) - 2^{-j(i)+1}$.

Proof. We know that $[T_{j(i)}] \supsetneq [T_{j(i+1)}]$, so let τ be a element of $T_{j(i)}$ such that there exists some $\upsilon \succ \tau$, with $|\upsilon| = j(i+1)$ and $\upsilon \notin T_{j(i+1)}$, but for all $\tau' <_\Gamma \tau$ for all $\upsilon' \succ \tau'$ with $|\upsilon'| = j(i+1)$, $\upsilon' \in T_{j(i+1)}$.

Now take any $\tau' \in T_{j(i)}$ such that $\tau <_\Gamma \tau'$. For any υ' of length $j(i+1)$, such that $\upsilon' \succ \tau'$ it follows by Lemma 5.2.1 that $\upsilon <_\Gamma \upsilon'$ and hence $\upsilon' \notin T_{j(i+1)}$ by Lemma 5.2.5. Thus $\tau' \in \hat{T}_{j(i)}$.

Thus for all $\tau' \in T_{j(i)}$, if $\tau' <_\Gamma \tau$, then $[\tau'] \subseteq [T_{j(i+1)}]$. If $\tau <_\Gamma \tau'$ then $[\tau'] \subseteq [\hat{T}_{j(i)}]$. If neither $\tau' <_\Gamma \tau$ nor $\tau <_\Gamma \tau'$, then τ' must be one of τ or $\bar{\tau}$.

This shows that $[\hat{T}_{j(i)}] \supseteq ([T_{j(i)}] \setminus [T_{j(i+1)}]) \setminus [\{\tau, \bar{\tau}\}]$. The result follows as $\mu[\{\tau, \bar{\tau}\}] = 2^{-j(i)+1}$. □

The following lemma shows us that the set $\hat{T}^s$ defined in the construction (now referred to as $\hat{T}$ because s is fixed) is just the same as $\bigcup_{i=0}^n \hat{T}_{j(i)}$.

Lemma 5.2.8. $\hat{T} = \bigcup_{i=0}^n \hat{T}_{j(i)}$.

Proof. If $\tau \in \hat{T}$, then by definition for all $\tau' \in T, \tau' \not\succ \tau$. Hence $\tau \in T_{j(i)}$ for some i and $\forall \tau' \succ \tau$, $\tau' \notin T_{j(i+1)}$. Thus $\tau \in \hat{T}_{j(i)}$, so $\hat{T} \subseteq \bigcup_{i=0}^n \hat{T}_{j(i)}$.

For the other direction, first note that $\hat{T}_{j(n)} = T_{s-1} \subseteq \hat{T}$ because any maximal length element must be a maximal element under $\preceq$. If for some $i, 0 \leq i < n$, $\tau \in \hat{T}_{j(i)}$, then for all $\tau' \in T_{j(i+1)}, \tau' \not\succ \tau$. Now for all $l > j(i)$, $[T_l] \subseteq [T_{j(i+1)}]$, thus for all $\tau' \in T_l$, $\tau' \not\succ \tau$ and so $\tau \in \hat{T}$. Hence $\bigcup_{i=0}^n \hat{T}_{j(i)} \subseteq \hat{T}$. □

Now as $j(0) = x$ and $S_x \subseteq O^{U_s}_{d+x}$, it follows that $\mu[T_{j(0)}] = 1$. We can assume that $x \geq 4$ because x is greater than any previous marker move stage and so this will be true after at most 3 marker move stages. This gives us that:

$$\begin{aligned}\mu[\hat{T}] &= \sum_{i=0}^{n} \mu[\hat{T}_{j(i)}] \\ &\geq \sum_{i=0}^{n-1} (\mu([T_{j(i)}] \setminus [T_{j(i+1)}]) - 2^{-j(i)+1}) + \mu[T_{j(n)}] \\ &= \mu[T_{j(0)}] - \sum_{i=0}^{n-1} 2^{-j(i)+1} \\ &> \frac{3}{4}.\end{aligned}$$

Now we have achieved the first step by finding a lower bound on $\mu[\hat{T}]$. The next step is to find a lower bound for $\mu[B^s]$. Recall that B^s was defined in the construction to be $\{\tau \in \hat{T} : \forall \tau' \succeq \tau, \tau' \notin \text{dom}(L_{s-1})\}$.

Lemma 5.2.9. *If $\tau \in \hat{T}$ then $\bar{\tau} \in \hat{T}$.*

Proof. If $\tau \in \hat{T}$, then for some i, $\tau \in \hat{T}_{j(i)}$. So $\tau \in T_{j(i)}$, and thus $\bar{\tau} \in T_{j(i)}$. Now if $\bar{\upsilon} \succ \bar{\tau}$ and $|\bar{\upsilon}| = j(i+1)$, then $\upsilon \succ \tau$ and hence $\upsilon \notin T_{j(i+1)}$ (as $\tau \in \hat{T}_{j(i)}$). Thus $\bar{\upsilon} \notin T_{j(i+1)}$ and so $\bar{\tau} \in \hat{T}_{j(i)}$. □

Lemma 5.2.10. *If $\tau \in \hat{T}$ and $\tau, \bar{\tau} \notin B^s$, then there exists $\upsilon \in dom(L_{s-1})$ with $\upsilon \succ \tau$ or $\upsilon \succ \bar{\tau}$.*

Proof. This lemma follows from the fact that we only add descriptions to L for two reasons. The first is to change the reduction and the second is to ensure that the domain is closed under substrings. Assume that there is no $\upsilon \in \text{dom}(L_{s-1})$ with $\upsilon \succ \tau$ or $\upsilon \succ \bar{\tau}$. In this case there is no need to add τ or $\bar{\tau}$ to the domain of L_{s-1} in order to close it under substrings. So if $\tau \notin B^s$, then it must be that $\tau \in \text{dom}(L_{s-1})$. Further we must have added τ to the domain of L_{s-1} to change the reduction Γ. In this case there is no reason why we should add $\bar{\tau}$ to change the reduction as well. Hence $\bar{\tau} \notin \text{dom}(L_{s-1})$ so $\bar{\tau} \in B^s$. □

Lemma 5.2.11. *If $\tau_1, \tau_2 \in \hat{T}$ with $|\tau_1| = |\tau_2|$ and $\tau_1 \prec_\Gamma \tau_2$, then at least one of τ_2 ,$\bar{\tau}_2$ are in B^s.*

Proof. First we know by Lemma 5.2.9 that $\bar{\tau}_1, \bar{\tau}_2 \in \hat{T}$. We will assume that $\tau_2, \bar{\tau}_2 \notin B^s$ and derive a contradiction.

By the last lemma if $\tau_2, \bar{\tau}_2 \notin B^s$, then there must be some $\upsilon_2 \in \text{dom}(L_{s-1})$ such that $\upsilon_2 \succ \tau_2$ or $\upsilon_2 \succ \bar{\tau}_2$. We will assume without loss of generality that $\upsilon_2 \succ \tau_2$ and that $|\upsilon_2|$ is maximal.

Let $k = |\upsilon_2|$. Note that $k < s$. As $\upsilon_2 \in \text{dom}(L_{s-1})$, then for all $\upsilon <_\Gamma \upsilon_2$, $\sigma_\upsilon \in O^{U_s}_{d+k}$ as this is how the construction chooses pairs to add to L. Hence for

all $\upsilon <_\Gamma \upsilon_2$, $\upsilon \in T_k$. Take any υ_1 extending τ_1 with $|\upsilon_1| = k$. Because $\tau_1 <_\Gamma \tau_2$, Lemma 5.2.1 gives us that $\upsilon_1 <_\Gamma \upsilon_2$ and thus $\upsilon_1 \in T_k$. Thus $[\tau_1] \subseteq T_k$ and so $\tau_1 \notin \hat{T}$ which contradicts our initial assumption. □

We can use this last lemma to put a lower bound on the measure of B^s.

Lemma 5.2.12. *If s is a marker move stage then $\mu[B^s] \geq \frac{1}{4}$.*

Proof. The previous lemma tells us that for any given length l, if there exists a $\tau \in \hat{T}$ of length l such that neither τ nor $\bar{\tau}$ are in B^s, then for any string $\upsilon \in \hat{T}$ of length l such that $\upsilon \neq \tau$ and $\upsilon \neq \bar{\tau}$, either υ or $\bar{\upsilon}$ are in B^s. Thus:

$$\mu[B^s] \geq \frac{1}{2}\left(\mu[\hat{T}^s] - \sum_{i=0}^{m} 2 \cdot 2^{-j(i)}\right) > \frac{1}{2}(\frac{3}{4} - \frac{1}{4}) = \frac{1}{4}.$$

□

Now for all $\tau \in B^s$, $\sigma_\tau \in O^{U_s}_{d+|\tau|}$ and so $M_U([\sigma_\tau]) \geq 2^{-d-|\tau|}$. Additionally, by construction, as B^s is a prefix-free set, so is $\{\sigma_\tau : \tau \in B^s\}$. Hence it follows that:

$$\begin{aligned} M_U([\{\sigma_\tau : \tau \in B^s\}]) &= \sum_{\tau \in B^s} M_U([\sigma_\tau]) \\ &\geq \sum_{\tau \in B} 2^{-d-|\tau|} \\ &= 2^{-d} \sum_{\tau \in B} \mu[\tau] \\ &\geq 2^{-d}\mu[B^s] \geq 2^{-d-2}. \end{aligned}$$

Lemma 5.2.13. *If s_1 and s_2 are both marker move stages and $s_1 \neq s_2$, then the sets $[\{\sigma_\tau : \tau \in B^{s_1}\}]$ and $[\{\sigma_\tau : \tau \in B^{s_2}\}]$ are disjoint.*

Proof. Take any $\tau \in B^{s_1}$, and $\upsilon \in B^{s_2}$. From the construction, $|\tau| < s_1$, and the length of υ is larger than any previous marker move stage so in particular $|\upsilon| > s_1 > |\tau|$. Now if $\upsilon \not\succ \tau$, then the construction ensures that σ_υ and σ_τ are incomparable. If $\upsilon \succ \tau$, then again by construction $\sigma_\upsilon \succ \rho_\tau$ and hence σ_υ and σ_τ are incomparable. □

Proof of Theorem 5.1.5. By Lemma 5.2.3, for any $k \in \omega$, L^{2k} is a strict process machine and hence N defined by $N(0^k) = \lambda$ and $N(0^k 1\sigma) = L^{2k}(\sigma)$ is a strict process machine. The argument at the start of this section shows that for some L^d, (5.2.1) holds. If we let Γ be the reduction constructed with L^d then by Lemma 5.2.4, if there are a finite number of marker moves then $\Gamma(O^{U_s}; x) = \emptyset'(x)$ for all but finitely many x. Now there can only be a finite number of

marker move stages because if C is the set of all marker move stages, then by the previous lemma:

$$M_U([\lambda]) \geq \sum_{s \in C} M_U([\{\sigma_\tau : \tau \in B^s\}]) \geq |C| 2^{-d-2}.$$

Hence $|C| \leq 2^{d+2}$ and in particular C is finite.

This construction is non-uniform in $\emptyset'$. This is because C is finite so $\emptyset'$ can determine the size of it by simply asking does another element enter enough times. The non-uniformity is due to the fact that the initial segment, and a d such that (5.2.1) holds, still need to be guessed. □

It is interesting to note that the construction used in Kummer's proof of Theorem 5.1.2 to show that for any universal machine is $\bar{R}^U_C$ is tt-complete is different. This construction uses a finite set of sequences $S_1, \ldots, S_n$. The key to unraveling the construction is to determine the maximum i such that S_i is infinite. This cannot be done using a $\emptyset'$ oracle. Additionally, as some initial segment still needs to be guessed, Kummer's construction is non-uniform in $\emptyset''$.

Proof of Corollary 5.1.6. The proof of theorem still works if $O^U_{K_m}$ is replaced by O^U_{KM}. First, if the construction enumerates some pair $\langle \sigma, n \rangle$ into L, this adds $\langle \sigma, n + d \rangle$ to O^U_{KM} as well as $O^U_{K_m}$ because $O^U_{K_m} \subseteq O^U_{KM}$.

During the verification of the construction, we proved by contradiction that the opponent could not force the reduction to be incorrect at a infinite number of points. During this proof, we did not consider the length of any description made by the opponent. We only considered the overall measure placed on elements of S_x. In fact we treated the opponent like a semimeasure. Hence as KM is the complexity derived from a optimal semimeasure, the same contradiction will ensue if $O^U_{K_m}$ is replaced by O^U_{KM}. □

Proof of Corollary 5.1.7. An optimal process machine is also a monotone machine. Hence the limitations exploited in the proof of an optimal monotone machine also apply to an optimal process machine. The machine N constructed in the proof is a strict process machine. □

5.3 Strict process complexity

In this section we will look at universal strict process machines. We will present a proof that there exists a universal strict process machine whose set of non-random strings is not tt-complete. For this section we will use $\bar{R}^M$ for $\bar{R}^M_{K_{M_S}}$.

First we will show that it is possible to construct a universal strict process machine whose set of non-random strings is closed under extension. In this section, we will take U to be a universal strict process machine. Again we will regard U as a c.e. set and as a function. We will take $\{U_s\}_{s\in\omega}$ to be an enumeration of U. Because U is a strict process machine, we can take our approximation to have the property that if $\langle\tau,\sigma\rangle \in U_s$ and $\tau' \prec \tau$ then there is some $\sigma' \preceq \sigma$ such that $\langle\tau',\sigma'\rangle \in U_s$. This can be done by simply waiting until $\langle\tau',\sigma'\rangle$ is enumerated into U. We will write $U_s(\tau)\downarrow$ if τ is an element of the domain of U_s and $U_s(\tau)\uparrow$ otherwise.

Theorem 5.3.1. *There exists a universal strict process machine V such that $\bar{R}^V$ is closed upwards under $\preceq$ i.e. if $\sigma \in \bar{R}^V$ and $\sigma' \succeq \sigma$ then $\sigma' \in \bar{R}^V$.*

Proof. In this proof, we construct V from a standard universal strict process machine U. We let 0 be the index of U in V, i.e. we set $V(0\tau) = U(\tau)$ for all $\tau \in 2^{<\omega}$. We use strings in the domain of V starting with 1 to get the desired closure property. If at some stage s, $U_s(\tau)\downarrow = \sigma$ and $|\tau| \leq |\sigma| - 2$, then $V(0\tau)\downarrow = \sigma$ with $|0\tau| < |\sigma|$. This means that $\sigma \in \bar{R}^V$. We want to make sure that all extensions of σ are added to $\bar{R}^V$. Because we are dealing with a strict process machine, we know that if $\tau' \prec \tau$ then $U_s(\tau')\downarrow$. We let τ' be the shortest initial segment of τ such that $|U_s(\tau')| \geq |\tau'| + 2$. Let $\upsilon = U_s(\tau')$. We need to make all extensions of υ non-random (σ must be one of these extensions). We do this by ensuring for all $\pi \in 2^{<\omega}$, that $V(1\tau'\pi) = \upsilon\pi$.

If all extensions of υ are non-random, then no string comparable with τ' can now be used by U to add a string to $\bar{R}^V$. If $\rho \prec \tau'$, then $U_s(\rho)\downarrow$ so if ρ made a string non-random with respect to V, we would have chosen ρ instead of τ'. If $\rho \succ \tau'$ then ρ can only describe extensions of υ which will already be in $\bar{R}^V$. Hence the set of descriptions that cause strings to be added to $\bar{R}^V$ is a prefix-free set. This means that if for all such τ', we set $V(1\tau'\pi) = \upsilon\pi$ for all $\pi \in 2^{<\omega}$ and $V(1\tau'') = \lambda$ for all $\tau'' \prec \tau'$, then V will remain a strict process machine. We now give a full construction and verification based on this idea.

Construction. At stage 0, set $V_0 = \{\langle\lambda,\lambda\rangle\}$. This is needed to keep V a strict process machine.

At stage $2s+1$ we add all descriptions of U_s to V by setting $V_{2s+1} = V_{2s} \cup \{\langle 0\tau,\sigma\rangle : \langle\tau,\sigma\rangle \in U_s\}$.

At stage $2s+2$, let $T_s = \{\tau \in 2^{<\omega} : |\tau| \leq s \text{ and } U_s(\tau)\downarrow \text{ and } |\tau| \leq |U_s(\tau)| - 2\}$. Let $\hat{T}_s$ be the set of minimal elements of T_s under the $\preceq$ relation. Set $V_{2s+2} = V_{2s+1} \cup \{\langle 1\tau\pi, U_s(\tau)\pi\rangle : \tau \in \hat{T}_s \text{ and } \pi \in 2^{<\omega} \text{ and } |\pi| \leq s\} \cup \{\langle 1\rho,\lambda\rangle : \exists\tau \in \hat{T}_s$ such that $\rho \prec \tau\}$.

Lemma 5.3.2. *V is a universal strict process machine.*

Proof. We will show that the set of ordered pairs V defined in the construction is in fact a strict process machine. We only need to consider pairs of the form $\langle 1\tau, \sigma\rangle$ because we know that U is a strict process machine. First by construction, it follows that if $\langle 1\tau, \sigma\rangle \in V_s$ for some stage s, then for all $\tau' \prec \tau$ there exists a $\sigma' \preceq \sigma$ such that $\langle 1\tau', \sigma'\rangle \in V_s$.

Thus all we need to show is that the set of ordered pairs is actually a function. First we will show that if $s_0 < s_1$, then $\hat{T}_{s_0} \subseteq \hat{T}_{s_1}$. Take any $\tau \in \hat{T}_{s_0}$. As $\tau \in \hat{T}_{s_0}$, $\tau \in T_{s_0}$ and so $\tau \in T_{s_1}$. Now if τ' is an initial segment of τ then as $U_{s_0}(\tau) \downarrow$ we have that $U_{s_0}(\tau') \downarrow$. However, $\tau' \notin T_{s_0}$ and so $\tau' \notin T_{s_1}$ and thus $\tau \in \hat{T}_{s_1}$.

Let $\langle 1\rho, \sigma_1\rangle, \langle 1\rho, \sigma_2\rangle \in V$. Let s_0 be the stage at which $\langle 1\rho, \sigma_1\rangle$ first entered V and s_1 the stage at which $\langle 1\rho, \sigma_2\rangle$ first entered V. Assume, without loss of generality, that $s_0 \leq s_1$. If $\sigma_1 = \lambda$ then $\rho \prec \tau$ for some $\tau \in \hat{T}_{s_0}$. Now $\tau \in \hat{T}_{s_1}$ and consequently we have that $\sigma_2 = \lambda$. If $\sigma_1 \neq \lambda$ then $1\rho = 1\tau\pi$ for some $\tau \in \hat{T}_{s_0}$ and $\pi \in 2^{<\omega}$ so $\sigma_1 = U(\tau)\pi$. As $\hat{T}_{s_0} \subseteq \hat{T}_{s_1}$ we have that $\sigma_2 = U(\tau)\pi$ as well. Thus $\sigma_1 = \sigma_2$ and V is a function. □

Lemma 5.3.3. *$\bar{R}^V$ is closed under extension.*

Proof. Assume that $\sigma \in \bar{R}^V$ and consider any $\upsilon \in 2^{<\omega}$. We will show that $\sigma\upsilon \in \bar{R}^V$. There must be some description $\rho \neq \lambda$ such that for some s_0, $V_{s_0}(\rho) = \sigma$ and $|\rho| < |\sigma|$.

If the first bit of ρ is 1 then by construction $\rho = 1\tau\pi$ and $\sigma = V_{s_0}(1\tau)\pi$ for some $\tau \in \hat{T}_{s_0}$ and $\pi \in 2^{<\omega}$. Let $s_1 = \max(s_0, 2|\pi\upsilon| + 2)$. As $\tau \in \hat{T}_{s_1}$, we have that $V_{s_1}(1\tau\pi\upsilon) = U_{s_1}(\tau)\pi\upsilon = V_{s_0}(1\tau)\pi\upsilon = \sigma\upsilon$. Now $|1\tau\pi\upsilon| = |\rho| + |\upsilon| < |\sigma| + |\upsilon| = |\sigma\upsilon|$. Hence $\sigma\upsilon \in \bar{R}^V$.

If the first bit of ρ is 0 then let $0\tau = \rho$. By construction $\tau \in \mathrm{dom}(U)$ so let s_0 be a stage such that $U_{s_0}(\tau) \downarrow = \sigma$. As $|\rho| < |\sigma|$, $|\tau| \leq |\sigma| - 2$ so $\tau \in T_{s_0}$ and hence there is some initial segment τ' of τ such $\tau' \in \hat{T}_{s_0}$. Let $\sigma' = U_{s_0}(\tau')$. Because $\tau' \in T_{s_0}$, we have that $|\tau'| \leq |\sigma'| - 2$. Let π be such that $\sigma = \sigma'\pi$. Let $s_1 = \max(s_0, 2|\pi\upsilon| + 1)$. As $\tau' \in \hat{T}_{s_1}$, we have that $V_{s_1}(1\tau'\pi\upsilon) = U_{s_1}(\tau')\pi\upsilon = \sigma'\pi\upsilon = \sigma\upsilon$. Now $|1\tau'\pi\upsilon| = 1 + |\tau'| + |\pi| + |\upsilon| < |\sigma'| + |\pi| + |\upsilon| = |\sigma\upsilon|$. So again $\sigma\upsilon \in \bar{R}^V$. □

□

The argument used does not generalise to process machines. To see why this is true, consider the following example. Let U be a universal process machine. Take stages $s_1 < s_2$ and assume that at stage s_1, $U_{s_1}(00) = 0000$,

$U_{s_1}(10) = 0001$ and $U_{s_1}(\lambda) \uparrow$. Now if we tried to follow the above construction, we would set $V_{2s_1+2}(100) = 0000$, and $V_{2s_1+2}(110) = 0001$. Now if at stage s_2, $U_{s_2}(\lambda) = 00$, then we would like to set $V_{2s_2+2}(1) = 00$, and somehow use extensions of 1 to make all extensions of 00 non-random. However, consider 001. It is not possible to set $V_{2s_2+2}(10) = 001$ or $V_{2s_2+2}(11) = 001$ and keep V as a process machine and so the argument fails. However, this does not rule out the possibility that another argument could be used.

For strict process machines, Theorem 5.3.1 allows us to remove a great deal of information from the set of non-random strings. We can use this theorem to prove that there exists a universal strict process machine whose set of non-random strings is not tt-complete. This proof is an adaptation of Muchnik's proof of the existence of a universal prefix-free machine whose overgraph is not tt-complete. However, as readers may not be familiar with this result we will present the proof in full. This proof technique uses the fact that the outcome of a finite game can be computably determined.

Consider the following game between two players. The game is played on a finite acyclic directed graph. Each vertex of the graph has the value 0 or 1. Each edge has a positive cost assigned to it. At any stage of the game a single vertex represents the game position. The game position begins at a designated start vertex. Player one and player two take turns. Each player starts with a finite amount of money. At each turn, a player can either pass or move the game position. If there is a directed edge from the current game position to another vertex, then the player can move the game position to that vertex. However, the player must pay the cost assigned to the edge. Because the graph is finite and acyclic, there must come a stage when there are no more moves that can be made, or both players elect to pass from that stage on. If player one and player two both start with the same amount of money, then either player one has a winning strategy to ensure that the game ends on a vertex labeled 0 or not. If not, then by passing on the first move, player one can adopt player two's strategy to prevent the game ending on a 0. Hence as all vertices are labeled, player one has a computable winning strategy to ensure the game ends on a 1.

We can turn a truth-table reduction from a computably enumerable set into such a game. Let Γ be a truth-table reduction. Let B be a c.e. set that both players can add elements to at some cost. Choose a witness n. Consider the truth-table for $\Gamma(n)$. Let $p_1, p_2, \ldots, p_k$ be the truth-table variables. The directed graph will be constructed as follows. The vertices represent the rows in the truth-table. The vertices are labeled with the value of the row, 0 or 1. There exists an edge from vertex v_1 to vertex v_2, if it possible to go from the row

associated with v_1 to the row associated with v_2 by changing some of the truth-table variables from 0 to 1. The edge cost is the cost to the players of adding to B the set of variables whose truth-table value changes. The game position starts at the vertex associated with the row with all variables 0. Players move by enumerating elements of $p_1, p_2, \ldots, p_k$ into B. The fact that player one has a winning strategy to ensure that either $\Gamma^B(n) = 0$ or $\Gamma^B(n) = 1$ will be used by the following proof to construct a universal strict process machine V such that $\bar{R}^V$ is not tt-complete. In this case we need to play an infinite number of games in order to diagonalise against all truth-table reductions.

In the proof that follows there will be three roles: the champion, the opponent and the arbitrator. The champion and the opponent will be players in the game. They will move by adding strings to $\bar{R}^V$. The arbitrator will make sure that the set of all non-random strings is closed under extension. The opponent represents the universal strict process machine. The index of the opponent in the proof is 000, the index of the champion is 01. We give the champion a shorter index because it will need more measure (measure will replace money in the games). The index of the arbitrator will be 1. The arbitrator acts just as in Theorem 5.3.1. Because the actions of the arbitrator can be determined, both players know that once a string σ is in $\bar{R}^V$, all extensions of σ will also be non-random. Hence when we consider $\bar{R}^{V_s}$, we will act under the assumption that this has already been closed under extensions.

We are now able to prove the main result of this section.

Proof of Theorem 5.1.8. In order to prove this theorem we will construct a universal strict process machine V and a c.e. set $A \subseteq \omega$ such that $A \not\leq_{tt} \bar{R}^V$. For this proof, we assume that the arbitrator acts behind the scenes and that for all s, $\bar{R}^{V_s}$ is closed under extensions. Let $\{\Gamma_n\}_{n\in\omega}$ be an enumeration of all partial truth-table reductions.

We will define $D_s = \{\tau \in 2^{<\omega} : U_s(\tau) \downarrow \text{ and } |\tau| < |U_s(\tau)| - 3\}$ so if $\tau \in D_s$, then $U_s(\tau) \in \bar{R}^V$. We will use D_s to determine how much the opponent spends in the games. We will show that if the opponent plays a move in a game between stages s_0 and s_1, then we can determine a lower bound for $\mu[D_{s_1}] - \mu[D_{s_0}]$.

Requirements. We have a requirement P_n for each $n \in \omega$.

P_n: There exist i, j such that $A(\langle n, i, j\rangle) \neq \Gamma_n^{\bar{R}^V}(\langle n, i, j\rangle)$.

The triples $\langle n, i, j\rangle$ will be used as follows. The n represents the reduction to be diagonalised. The i is incremented every time the requirement is injured by a higher priority requirement. It also provides an upper bound on

the measure that can be used by the players in the game. The j provides us with a series of games for each diagonalisation; it will be incremented if our opponent ever 'breaks' the rules of the game by using too much measure.

Construction. At stage 0, set $V_0 = \{\langle\lambda,\lambda\rangle, \langle 0,\lambda\rangle, \langle 00,\lambda\rangle\}$. Set $A_0 = \emptyset$.

At stage $2s+1$ do the following. For all $n < s$, if P_n does not have a witness, assign $\langle n, 0, 0\rangle$ to P_n. For all $n < s$, let $\langle n, i_n, j_n\rangle$ be the witness for P_n. If P_n does not have a game assigned, run $\Gamma_n(\langle n, i_n, j_n\rangle)$ for s steps to see if it returns truth-table. If it does return a truth-table, let $X_n = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ be the set of strings used as variables by this truth-table. For the purpose of this game, we will assume that higher priority requirements have stopped acting, i.e., that the associated games are finished. Because of this, we do not want to injure any higher priority games, so let: $Y_n = \{\sigma \in X_n$: for all $\tau \in \cup_{i<n} X_i, \sigma \not\preceq \tau$ or $\tau \in \bar{R}^{V_{2s}}\}$. Notice that $\sigma \in X_n \setminus Y_n$ if and only if adding σ to $\bar{R}^{V_{2s+1}}$ would change some variable used by a higher priority game (as $\bar{R}^{V_{2s+1}}$ is closed under extension).

The game $G_{\langle n,i_n,j_n\rangle}$ is defined as follows. We will assume that the strings in $X_n \setminus Y_n$ do not change (if they do change then this will affect a higher priority game). The vertices in the game correspond to possible truth assignments to the variables in Y_n. The vertices are labeled with the value of the corresponding line in the truth-table (assuming those variables in $X_n \setminus Y_n$ retain their current values). An edge exists from a vertex v_1 to a vertex v_2, if it is possible to go from the row associated with v_1 to the row associated with v_2 by changing some of the truth-table variables from 0 to 1. If going from vertex v_1 to vertex v_2 requires changing the variables in $\Sigma \subseteq Y_n$, then the cost associated with the edge is $\mu[\Sigma]$. The amount of measure each player has to spend on the game is 2^{-n-i_n-6}. The game $G_{\langle n,i_n,j_n\rangle}$, though defined, is said to be uninitialised. We allow the opponent to move by setting $V_{2s+1} = V_{2s} \cup \{\langle 000\tau, \sigma\rangle : \langle \tau, \sigma\rangle \in U_s\}$.

At stage $2s + 2$, we determine whether there is any game that the champion needs to attend to. We find all games assigned to requirements, that are uninitialised, or where the opponent has made a move. The opponent is considered to have made a move if some new strings used by the truth-table reduction have been enumerated into $\bar{R}^{V_{2s+1}} \setminus \bar{R}^{V_{2s}}$. If no such games exist, then we set $V_{2s+2} = V_{2s+1}$. Otherwise let $G_{\langle n,i_n,j_n\rangle}$ be the highest priority game (i.e. game with the smallest n) that needs attention. First we reset all lower priority games. For all p such that $n < p \leq s$, let $\langle p, i_p, j_p\rangle$ be the current witness assigned to R_p. Remove this witness and the associated game and let $\langle p, i_p+1, 0\rangle$ be the new witness.

If $G_{\langle n,i_n,j_n\rangle}$ is a game which is uninitialised, then we set the start position

for the game to be the vertex that corresponds to assigning $\langle n, i_n, j_n\rangle$ a truth value of 1 if and only if $\langle n, i_n, j_n\rangle \in \bar{R}^{V_{2s+2}}$. The champion decides whether to take a winning strategy to ensure that $\Gamma_n(\langle n, i_n, j_n\rangle) = 0$ or $\Gamma_n(\langle n, i_n, j_n\rangle) = 1$. In the first case we add $\langle n, i_n, j_n\rangle$ to A_s, in the second case we leave it out. We let Σ be the set of strings that the champion needs to enumerate into $\bar{R}^{V_{s+2}}$ for the first step of this strategy. We now say that the game $G_{\langle n,i_n,j_n\rangle}$ has been initialised.

If $G_{\langle n,i_n,j_n\rangle}$ is a game that our opponent has made a move on, then let $2s_0$ be the stage at which this game was initialised. If $\mu[D_s] - \mu[D_{s_0}] \geq 2^{-n-i_n-6}$, then the opponent has exceeded the allocated measure for the game $G_{\langle n,i_n,j_n\rangle}$. In this case, remove $\langle n, i_n, j_n\rangle$ as a witness for P_n and also remove the game. Let $\langle n, i_n, j_n + 1\rangle$ be the new witness. Let $\Sigma = \emptyset$. If the opponent has not exceeded the allocated measure then, let Σ be the set of strings that the champion needs to enumerate into $\bar{R}^{V_{s+2}}$ for the next move in the pre-determined winning strategy.

Now we need to add Σ to $\bar{R}^{V_{s+2}}$ in order to make the champion's next move. We know that the arbitrator will ensure that $\bar{R}^{V_{s+2}}$ is closed under extensions. We take $\hat{\Sigma} = \{\sigma_1, \ldots, \sigma_k\}$ to be a prefix-free set formed by taking the $\preceq$ minimal elements of Σ. The champion only needs to enumerate $\hat{\Sigma}$ into $\bar{R}^{V_{s+2}}$. We will use the Kraft Computable Theorem to find descriptions for these strings.

We use the Kraft Computable Theorem to request a string τ_i of length $|\sigma_i| - 3$ for all i, $1 \leq i \leq k$. We set $V_{2s+2} = V_{2s+1} \cup \{\langle 01\tau_i, \sigma_i\rangle : 1 \leq i \leq k\} \cup \{\langle 01\tau, \lambda\rangle : \exists i, 1 \leq i \leq k \text{ such that } \tau \prec \tau_i\}$.

Note that the champion decreases the measure available for future requests by $2^3\mu[\Sigma]$. However by scaling, we can regard the champion as having $\frac{1}{8}$ of measure to spend, and this move costing the champion $\mu[\Sigma]$.

Verification. The first step in verifying this proof is to show that if the opponent makes a move then it must pay the cost of the move.

Lemma 5.3.4. *If the opponent enumerates a set of strings Σ into $\bar{R}^{V_{2s_1}} \setminus \bar{R}^{V_{2s_0}}$, then $\mu[D_{s_1}] - \mu[D_{s_0}] \geq 2^4\mu[\Sigma]$.*

Proof. Let $\hat{\Sigma}$ be the set of minimal elements of Σ under the $\preceq$ relation. Let $\hat{\Sigma} = \{\sigma_1, \ldots, \sigma_k\}$. For all $i \in \omega$ with $1 \leq i \leq k$, there exists some $\tau_i \in 2^{<\omega}$ such that $\tau_i \notin \text{dom}(U_{s_0})$, and $U_{s_1}(\tau_i) \preceq \sigma_i$, $|\tau_i| < |U_{s_1}(\tau_i)| - 3$ (i.e. $\tau_i \in D_{s_1}$). Let $C = \{U_{s_1}(\tau_i) : 1 \leq i \leq k\}$. Now $[C] \supseteq [\Sigma]$ so we can let $\hat{C}$ be a minimal subset of C such that $[\hat{C}] \supseteq [\Sigma]$. Hence $\mu[\hat{C}] \geq \mu[\Sigma]$. For all $\upsilon \in \hat{C}$, choose an i such that $U(\tau_i) = \upsilon$ and let I be the set of all such i. The set $\{\tau_i : i \in I\}$ is prefix-free

because its image under U is prefix-free and U is a strict process machine. It follows that:

$$\begin{aligned}\mu[\{\tau_i : i \in I\}] &= \sum_{i\in I} 2^{-|\tau_i|} \\ &\geq \sum_{i\in I} 2^{4-|U_{s_1}(\tau_i)|} \\ &= 2^4\mu[\hat{C}] \\ &\geq 2^4\mu[\Sigma].\end{aligned}$$

Finally take any $i \in I$. If $\tau \succeq \tau_i$, then $\tau \notin \mathrm{dom}(U_{s_0})$, as the domain of a strict process machine is closed under substrings. So $\tau \notin D_{s_0}$. If $\tau \prec \tau_i$, and $\tau \in D_{s_0}$, then $U_{s_0}(\tau) \in \bar{R}^{V_{s_0}}$ and hence as $U_{s_0}(\tau) \preceq U_{s_1}(\tau_i) \preceq \sigma_i$, $\sigma_i \in \bar{R}^{V_{s_0}}$. But we assumed that $\sigma_i \notin \bar{R}^{V_{s_0}}$ so again $\tau \notin D_{s_0}$. Hence $[\tau_i] \cap [D_{s_0}] = \emptyset$. The result follows as $[\{\tau_i : i \in I\}] \cup D_{s_0} \subseteq D_{s_1}$. □

Again by scaling we can regard the opponent as having measure of $\frac{1}{16}$ and the cost of the move as being $\mu[\Sigma]$.

Lemma 5.3.5. *V is a strict process machine.*

Proof. U is by assumption a strict process machine, so to check that V is a strict process machine, we just need to check the strings enumerated into V by the champion. As the champion is effectively a prefix-free machine, we just need to show that the champion does not run out of measure.

To do this we divide the games into two sorts, those games $G_{\langle n,i,j\rangle}$ with $j = 0$ and those games with $j > 0$. We know the champion always keeps within the rules of the game. Let C be the cost to the champion of playing those games with $j = 0$. C is less than the sum of the measure allocated to each game. Hence $C \leq \sum_{n\in\omega}\sum_{i\in\omega} 2^{-n-i-6} = \sum_{n\in\omega} 2^{-n-5} = \frac{1}{16}$.

Now j is only incremented if the opponent exceeds the amount of measure allocated to a game. Hence the measure the champion spends on these games is always less than the measure the opponent spends overall. As the opponent only has $\frac{1}{16}$ to spend, it follows that the champion spends less than $C + \frac{1}{16} = \frac{1}{8}$, the amount of measure available to it. Hence the champion does not run out of measure and thus V is a strict process machine. □

Lemma 5.3.6. *All requirements are met.*

Proof. Take any requirement P_n. Assume that at some stage s_0 all higher priority requirements have stopped acting. Let $\langle n, i, j\rangle$ be the witness assigned to P_n at stage s_0. Because all higher priority requirements have stopped acting,

i is never incremented again. So if the witness is changed, it must be because j is increased. This in turn must be caused by the opponent exceeding their allocated measure of 2^{-n-i-6} in the previous game. This can only happen a finite number of times otherwise the opponent will run out of measure.

Thus there is some final witness $\langle n, i_n, j_n \rangle$ assigned to P_n. If $\Gamma_n(\langle n, i_n, j_n \rangle)$ never halts then the requirement is met. If $\Gamma_n(\langle n, i_n, j_n \rangle)$ does halt then the champion will adopt a winning strategy for the game $G_{\langle n,i_n,j_n \rangle}$ and so in either case $\Gamma_n^{\bar{R}^V}(\langle n, i_n, j_n \rangle) \neq A(\langle n, i_n, j_n \rangle)$. □

□

5.4 Open questions

The results on tt-completeness of the overgraphs and the sets of non-random strings obtained from the complexity measures: K, K_{M_S}, K_{M_D}, K_m, KM and C can be summarised as follows:

Complexity	$\bar{R}^U$ tt-complete?	O^U tt-complete?
K	Dependent on machine	Dependent on machine
K_{M_S}	Dependent on machine	Always
K_{M_D}, K_m or KM	True for some universal machines	Always
C	Always	Always

This table includes the results of: Kummer; Muchnik and Positselsky; Allender, Buhrman and Koucký. In this table, 'Always' means that for every optimal machine the set in question is tt-complete. 'Dependent on machine' means that there exists two different optimal machines such that for one the set is tt-complete and for the other the set is not tt-complete.

This leaves the following outstanding questions.

Question 5.4.1. Does there exist an optimal monotone machine U such that $\bar{R}^U_{K_m}$ or is not tt-complete?

Question 5.4.2. Does there exist an optimal monotone machine U such that $\bar{R}^U_{KM}$ is not tt-complete?

Question 5.4.3. Does there exist an optimal process machine U such that $\bar{R}^U_{K_{M_D}}$ is not tt-complete?

Part II

Randomness and Computability in Cantor Space

Chapter 6

Process and Truth-Table Characterisations of Randomness

6.1 Overview

In Chapter 1 we defined the Martin-Löf random sequences to be those sequences which avoided all Martin-Löf tests. In Chapter 2, we saw that the Martin-Löf random sequences could be characterised using Kolmogorov complexity. These two approaches are known as the test perspective and the compressibility perspective. In this chapter we will look at randomness from a third perspective known as the betting perspective. We will see how this perspective can be used to define notions of randomness that differ from Martin-Löf randomness. We will then characterise these notions using a variant of strict process machines.

The betting perspective is typically formalised using martingales. This use of martingales has found widespread application. For example, Lutz pioneered the use of martingales to study the exponential time complexity classes [61]. In this chapter, we show how to move between the betting perspective and the compressibility perspective by translating back and forth between martingales and a variant of process machines. This allows us to provide consistent compressibility-based definitions of many types of randomness. The main theorems of this chapter provide new characterisations of computable randomness, Schnorr randomness and weak randomness. The quick process machines that we will use come from work of Levin [58, 94].

Definition 6.1.1. A *martingale* is function $d : 2^{<\omega} \to \mathbb{R}^{\geq 0}$ such that for all $\sigma \in 2^{<\omega}$ we have:

$$d(\sigma) = \frac{d(\sigma 0) + d(\sigma 1)}{2}. \tag{6.1.1}$$

A martingale is a strategy for betting on the bits of a sequence. The martingale condition (6.1.1) ensures that the betting is fair. We say that a martingale

succeeds on a sequence $X \in 2^\omega$ if $\lim_{n\to\infty} d(X \upharpoonright n) = +\infty$.

The idea is that a sequence is not random if a gambler could make an infinite amount of money betting on the bits of the sequence. For this to work, we need to place some effectivity constraints on the martingales, and potentially the speed at which the gambler makes money. By changing the constraints we get different notions of randomness. Schnorr established that with appropriate effectivity constraints, martingales could be used to characterise Martin-Löf randomness [78].

Theorem 6.1.2 (Schnorr [78])**.** *A sequence $X \in 2^\omega$ is Martin-Löf random if and only if no computably enumerable martingale succeeds on X.*

In this chapter we will investigate the following notions of randomness: computable randomness, Schnorr randomness and weak randomness. These notions were originally defined in different ways. However, they can all be characterised in terms of martingales. For simplicity, we will take these martingale characterisations as our definitions.

Definition 6.1.3. A function $h : \omega \to \omega$ is called an *order function* if it is computable, non-decreasing, and unbounded.

Definition 6.1.4. (i). A sequence $X \in 2^\omega$ is *computably random* if no computable martingale succeeds on X.

(ii). A sequence $X \in 2^\omega$ is *Schnorr random* if for all computable martingales d, for all orders h, for almost all n, $d(X \upharpoonright n) < h(n)$.

(iii). A sequence $X \in 2^\omega$ is *weakly random* if for all computable martingales d, for all orders h, there exists an n such that $d(X \upharpoonright n) < h(n)$.

The definitions of Schnorr randomness and computable randomness are due to Schnorr [78]. Schnorr argued that the effectivity requirements in the definition of computable randomness were insufficient. He suggested that to make the martingale truly effective, the gambler should be able to identify when they were winning. Weak randomness was first defined by Kurtz [51]. The characterisation in terms of martingales was established by Wang [91].

For Martin-Löf randomness or computable randomness, if d is the martingale in question, it is only necessary that $\lim_{\sup} d(X \upharpoonright n) = +\infty$. This is because given such a d, using a simple procedure known as the savings trick, one can define another martingale $\hat{d}$ such that $\lim_{n\to\infty} \hat{d}(X \upharpoonright n) = +\infty$. Details of the savings trick can be found in [30, 70].

We defined strict process machines in Definition 2.1.7 and noted that this definition was due to Levin. In addition to defining strict process machines,

Levin defined a strict process machine P as being *applicable* to a sequence X, if there was an order h such that $|P(X \upharpoonright n)| \geq h(n)$ [58, 94]. We suggest the following name for a strict process machine that is applicable to all sequences.

Definition 6.1.5. A strict process machine P is a *quick process machine*, if it is total and there is an order function h such that for all $\tau \in 2^{<\omega}$, $|P(\tau)| \geq h(|\tau|)$.

Given a quick process machine P, there is a simple procedure to determine the complexity of any string σ with respect to P, i.e. $C^P(\sigma)$ is a computable function. This can be done by running $P(\tau)$ for all strings such that $h(|\tau|) \leq |\sigma|$. If σ has a P-description, it must be one of these strings.

In Theorem 6.2.16, a characterisation of computable randomness, Schnorr randomness, and weak randomness in terms of quick process machines is made. These are not the first compressibility characterisations of these types of randomness. Downey and Griffiths have characterised Schnorr randomness in terms of computable measure machines [28]. Mihailović provided a machine characterisation of computable randomness in terms of bounded measure machines [30]. Downey, Griffiths and Reid characterised weak randomness in terms of computably layered machines [29]. The value of these new characterisations lies in their simplicity and consistency.

Underlying Theorem 6.2.16 is a new technique for building strict process machines. This technique can be thought of as a KC Theorem (Theorem 2.1.5) for strict process machines. It allows strict process machines to be built by listing the strings that need descriptions, the description length and the relationship between described strings. This technique is presented in Theorem 6.2.4 and Theorem 6.2.6.

Demuth showed that there is a link between truth-table reducibility and randomness [25]. We provide further evidence for this link. There is a close relationship between quick process machines and truth-table functionals. We use this relationship to provide truth-table reducibility characterisations of computable randomness, Schnorr randomness and weak randomness in Theorem 6.3.1.

6.2 Quick process machines and randomness

Our goal is to show that quick process machines can be used to characterise computable randomness, Schnorr randomness and weak randomness. This is due to the fact that quick process machines are very similar to martingales.

Proposition 6.2.1 shows how to construct a martingale from a quick process machine. This proof is essentially due to Levin.

Proposition 6.2.1 (Levin [58]). *For any quick process machine P, there is a computable martingale d such that for all σ, $d(\sigma) \geq 2^{|\sigma| - C^P(\sigma)}$.*

Proof. Let P be a quick process machine with associated order function h. Define $g(n) = \min\{x : h(x) > n\}$. If $\tau \in \{0,1\}^{g(n)}$ then $|P(\tau)| \geq h(|\tau|) = h(g(n)) > n$. We define a computable martingale d as follows. First for any string σ define $E_\sigma = \{\tau \in \{0,1\}^{g(|\sigma|)} : P(\tau) \succ \sigma\}$. Now define d by:

$$d(\sigma) = \frac{|E_\sigma|}{2^{g(|\sigma|)-|\sigma|}}.$$

The function d is computable because P is total so E_σ is computable for all σ. We will now show that d is a martingale. For any σ, $E_{\sigma 0}$ and $E_{\sigma 1}$ are disjoint because P is a function. If $\tau \in E_{\sigma 0}$ then $\tau \upharpoonright g(|\sigma|) \in E_\sigma$ because $P(\tau \upharpoonright g(|\sigma|)) \preceq P(\tau)$, $\sigma \preceq P(\tau)$ and $|\sigma| < |P(\tau \upharpoonright g(|\sigma|))|$ so $\sigma \prec P(\tau \upharpoonright g(|\sigma|))$. Similarly if $\tau \in E_{\sigma 1}$, then $\tau \upharpoonright g(|\sigma|) \in E_\sigma$.

Now if $\tau \in E_\sigma$, and $\tau' \succeq \tau$ with $|\tau'| = g(|\sigma| + 1)$ then it must be that $|P(\tau')| \geq h(|\tau'|) = h(g(|\sigma| + 1)) > |\sigma| + 1$. Hence $\tau' \in E_{\sigma 0} \cup E_{\sigma 1}$. Thus we have that $(|E_{\sigma 0}| + |E_{\sigma 1}|)2^{g(|\sigma|)-g(|\sigma+1|)} = |E_\sigma|$. This gives us that:

$$d(\sigma) = \frac{|E_\sigma|}{2^{g(|\sigma|)-|\sigma|}} = \frac{|E_{\sigma 0}| + |E_{\sigma 1}|}{2^{g(|\sigma+1|)-|\sigma|}} = \frac{1}{2}\frac{|E_{\sigma 0}| + |E_{\sigma 1}|}{2^{g(|\sigma+1|)-(|\sigma|+1)}} = \frac{d(\sigma 0) + d(\sigma 1)}{2}.$$

Assume $C^P(\sigma) = |\sigma| - c$, and $|\sigma| = n$. Then for some τ with $|\tau| = n - c$, we have that $P(\tau) = \sigma$. This means that $h(|\tau|) \leq n$ and consequently that $g(n) > |\tau|$. If $\tau' \succeq \tau$ with $|\tau'| = g(n)$, then $P(\tau') \succ \sigma$ and so $\tau' \in E_\sigma$. Hence $|E_\sigma| \geq 2^{g(n)-(n-c)}$. Thus:

$$d(\sigma) = \frac{|E_\sigma|}{2^{g(n)-n}} \geq 2^c = 2^{|\sigma| - C^P(\sigma)}.$$

□

Our next objective is to build a quick process machine from a martingale. While Levin showed how to build a strict process machine from a computable measure [58], Levin's objective was to show that any computable measure could be obtained using similar techniques to Proposition 6.2.1. The strict process machine Levin built was not total but instead applicable to a set of uniform measure 1. Further, Levin did not relate the complexity of a string σ with respect to the strict process machine created, to the measure of the basic clopen set $[\sigma]$.

Before showing how to build a quick process machine from a martingale, we will present a new technique for building strict process machines. The benefit of this technique is that it allows us to build strict process machines without worrying about which descriptions to use. Instead, like the KC theorem, we request a description length.

Definition 6.2.2. We call a partial computable function $f : \omega^{<\omega} \to 2^{<\omega} \times \omega$ a *strict process request function*, if:

(i). The domain of f is closed downwards.

(ii). For all $\rho \in \text{dom}(f)$, $x \in \omega$, we have $2^{-f_2(\rho)} \geq \sum_{\rho x \in \text{dom}(f)} 2^{-f_2(\rho x)}$.

(iii). If $\rho_1, \rho_2 \in \text{dom}(f)$ and $\rho_1 \prec \rho_2$, then $f_1(\rho_1) \preceq f_1(\rho_2)$, and $f_2(\rho_1) < f_2(\rho_2)$.

In this definition, f_1 and f_2 are the co-ordinate functions of f, and $\omega^{<\omega}$ is the set of all finite strings of elements of ω.

The idea behind this definition is the following. We want to represent the essential combinatorics of a strict process machine by a tree. Each node of the tree maps to a pair (σ, n) where $\sigma \in 2^{<\omega}$ and $n \in \omega$. When we turn this tree into a strict process machine, this node generates a description of σ of length n. Suppose we have two nodes ρ_1, ρ_2. If ρ_1 is an initial segment of ρ_2, then the description generated by ρ_1 will be an initial segment of the description generated by ρ_2. We need the second condition so that the combined weight of the descriptions generated by the children of a node, does not exceed the weight of the description generated by the node itself. We need $f_1(\rho_1) \preceq f_1(\rho_2)$ in order to make a strict process machine. We need $f_2(\rho_1) < f_2(\rho_2)$ so that no two nodes generate the same description.

Definition 6.2.3. A strict process machine P *implements* a strict process request function f if for all $\sigma \in 2^{<\omega}$, $C^P(\sigma) \leq \min\{n : (\sigma, n) \in \text{rng}(f)\}$.

Theorem 6.2.4. *Any strict process request function is implemented by some strict process machine, and any strict process machine implements some strict process request function.*

Proof. Given a strict process machine P, there is a natural strict process request function f that P implements. We define f by $f(\tau) = (P(\tau), |\tau|)$. In this case we have $\text{dom}(f) = \text{dom}(P) \subseteq 2^{<\omega}$.

Given a strict process request function f, each node $\rho \in \text{dom}(f)$ defines a prefix-free machine M_ρ via the KC theorem in the following manner. Each time some $f(\rho x)$ halts equal to (σ, n) for some $x \in \omega$, we add $(\hat{\sigma}, n - f_2(\rho))$ to our KC request sequence where $f_1(\rho)\hat{\sigma} = \sigma$. Condition (ii) for f to be a strict process request function ensures that the weight of the requests does not exceed 1.

Given τ in the domain of M_ρ, we can determine the node ρx that made the request that returned τ. We will call ρx the node associated with τ and M_ρ. In the verification we will make use of the fact that $f_1(\rho)M_\rho(\tau) = f_1(\rho)\hat{\sigma} = f_1(\rho x)$.

We will now build a strict process machine P that implements f. If f is the empty function, then so is P. Otherwise $f(\lambda)$ halts and so we can define $\tau_0 = 0^{f_2(\lambda)}$, and $P(\tau') = f_1(\lambda)$ for any $\tau' \preceq \tau_0$.

We set $P(\tau) = \sigma$ if at any stage we find a decomposition of $\tau = \tau_0\tau_1 \dots \tau_n$, and a node $\rho \in \omega^{<\omega}$ of length $n-1$ with the following properties for all i with $0 \leq i \leq n-1$:

(i). $\tau_{i+1} \in \text{dom}(M_{\rho\restriction i})$.

(ii). $\rho \restriction (i+1)$ is the node associated with τ_{i+1} and $M_{\rho\restriction i}$.

(iii). $\sigma = f_1(\lambda)M_{\rho\restriction 0}(\tau_1) \dots M_{\rho\restriction(n-1)}(\tau_n)$.

To make P a strict process machine, we need to close the domain of P downwards. To achieve this, we also define $P(\tau') = P(\tau_0\tau_1 \dots \tau_{n-1})$ for all τ' such that $\tau_0 \dots \tau_{n-1} \prec \tau' \prec \tau$.

If P fails to be a strict process machine, then there must be some τ, π in the domain of P with $\tau \preceq \pi$ such that $P(\tau) \not\preceq P(\pi)$. Further we can assume that neither τ nor π were added to the domain of P in order to close the domain downwards. Let us take the decompositions used by the construction to be $\tau = \tau_0\tau_1 \dots \tau_m$ and $\pi = \pi_0\pi_1 \dots \pi_n$.

By construction, $\tau_0 = \pi_0 = 0^{f_2(\lambda)}$. Further, $\tau_1 = \pi_1$ because M_λ is a prefix-free machine. This also means that the node ρ_1 associated with τ_1 and M_λ, is the same node associated with π_1 and M_λ. Hence $\tau_2 = \pi_2$ because M_{ρ_1} is a prefix-free machine. By repeating this argument we establish that $\pi = \tau_0 \dots \tau_m\pi_{m+1} \dots \pi_n$. If $\pi = \tau$ then $m = n$ because $\pi_i \neq \lambda$ if $i \geq 1$ (this is a consequence of f_2 being strictly increasing). In this case $P(\tau) = P(\pi)$. If $m < n$, then $P(\pi) \succeq f_1(\lambda)M_{\rho_0}(\tau_1) \dots M_{\rho_{n-1}}(\tau_n) = P(\tau)$. Thus P is a strict process machine.

We will now verify that P implements f. Let $\tau_0 = 0^{f_2(\lambda)}$. Let $\rho \in \text{dom}(f)$, and let $(\sigma, n) = f(\rho)$. For $i \in \omega$, $0 \leq i < |\rho|$, for each machine $M_{\rho\restriction i}$ let τ_{i+1} be the element of the domain of this machine such that the node $\rho \restriction (i+1)$ is associated with τ_{i+1} and $M_{\rho\restriction i}$. For all such i, $f_1(\rho \restriction i)M_{\rho\restriction i}(\tau_{i+1}) = f_1(\rho \restriction (i+1))$. Hence we have that $f_1(\lambda)M_{\rho\restriction 0}(\tau_1) \dots M_{\rho\restriction(|\rho|-1)}(\tau_{|\rho|}) = f_1(\rho) = \sigma$. Thus $P(\tau_0\tau_1 \dots \tau_n) = \sigma$, and:

$$\sum_{i=0}^{|\rho|} |\tau_i| = f_2(\lambda) + \sum_{i=1}^{|\rho|} (f_2(\rho \restriction i) - f_2(\rho \restriction (i-1))) = f_2(\rho) = n.$$

□

The proof shows that the implementation is uniform; given an index for a strict process request function f, we can compute an index for a strict process machine P that implements f.

We can do a similar thing for quick process machines. The first two conditions below ensure that the quick process machine we create is total. The third condition ensures that once we build a process machine there is some order function h such that for all $\tau \in 2^{<\omega}$, $|P(\tau)| \geq h(|\tau|)$.

Definition 6.2.5. A strict process request function f is a *quick process request function* if:

(i). The domain of f is finitely branching.

(ii). For all $n \in \omega$, $\sum_{\rho \in \mathrm{dom}(f), |\rho|=n} 2^{-f_2(\rho)} = 1$.

(iii). For some order function h, for all $\rho \in \mathrm{dom}(f)$, $|f_1(\rho)| \geq h(|\rho|)$.

Theorem 6.2.6. *Any quick process request function is implemented by a quick process machine, and any quick process machine implements a quick process request function.*

Proof. Given a quick process machine P, the natural quick process request function that P implements is again defined by $f(\tau) = (P(\tau), |\tau|)$.

Given a quick process request function f, we use Theorem 6.2.4 to construct a strict process machine P which implements f. If we fix n, then the additional conditions on f mean that there is a finite number of strings of length n in the domain of f. Further the fact that

$$\sum_{\rho \in \mathrm{dom}(f), |\rho|=n} 2^{-f_2(\rho)} = 1 \tag{6.2.1}$$

implies that the domain of f is computable because once sufficient strings enter the domain of f in order for (6.2.1) to hold, we know that no more strings of length n will enter the domain of f.

If we use the construction of Theorem 6.2.4, then each string ρ of length n in the domain of f results in a unique string τ entering the domain of P. Further because $|\tau| = f_2(\sigma)$ and there are only finitely many such τ, in order for (6.2.1) to hold, we must have that the set of such τ form a covering of 2^ω. Because the domain of P is closed downwards, and $|\tau| \geq n$ we have that the function P is total.

Let $l(x) = \max\{f_2(\rho) : |\rho| = x \text{ and } \rho \in \mathrm{dom}(f)\}$. This function is computable because the domain of f is computable and finitely branching. If

$|\tau| \geq l(x)$, then for some $\tau' \preceq \tau$ we have that $P(\tau') = f_1(\rho)$ for some ρ of length x. This follows from condition (ii) for f to be a quick process request function. This means that $|P(\tau)| \geq |f_1(\rho)| \geq h(x)$.

We can define another order function h' by $h'(n) = \max\{x : l(x) \leq n\}$. As $|\tau| \geq l(h'(|\tau|))$, we have that $|P(\tau)| \geq h(h'(|\tau|))$ and because $h \circ h'$ is also an order function, P is a quick process machine. □

We will now present a method for translating from computable martingales to quick process machines. One difficulty is that, when running a computable martingale, the value of $d(\sigma)$ can increase at any stage, albeit by a very small amount. However, we can avoid this issue by using the following result of Schnorr [79].

Proposition 6.2.7 (Schnorr)**.** *If d is a computable martingale, then there exists a computable martingale $\hat{d}$ such that for all $\sigma \in 2^{<\omega}$:*

(i). $d(\sigma) \leq \hat{d}(\sigma) \leq d(\sigma) + 2$.

(ii). $\hat{d}(\sigma)$ *is a computable dyadic rational i.e. there is an integer pair (n, z) uniformly computable from σ such that $\hat{d}(\sigma) = n2^z$.*

From now on we will assume that all our martingales d are dyadic rational valued, and further that $d(\lambda) = 1$. Given such a computable martingale, we can define its *precision function* $r : \omega \to \omega$ by $r(n) = \min\{m \in \omega : \forall \sigma \in 2^{<\omega}, |\sigma| = n \Rightarrow d(\sigma)$ is an integer multiple of $2^{-m}\}$. The precision function of a computable martingale is computable.

We will also make use of the following lemma.

Lemma 6.2.8. *Let $(a_1, a_2, \ldots, a_n)$ be a tuple of positive integers and $m \in \omega$ such that $m \leq a_i$ for all $1 \leq i \leq n$. If $\sum_{i=1}^{n} 2^{-a_i} \geq 2^{-m}$, then for some $J \subseteq \{1, \ldots, n\}$ we have that $\sum_{i \in J} 2^{-a_i} = 2^{-m}$.*

Proof. We can assume that the elements of the tuple are non-decreasing. Consider the partial sum $S_k = \sum_{i=1}^{k} 2^{-a_i}$. $S_1 \leq 2^{-m}$. If $S_k < 2^{-m}$, it must be that $S_k \leq 2^{-m} - 2^{-a_k}$ (as S_k is some integer multiple of 2^{-a_k}). As $a_{k+1} \geq a_k$, we have that $S_{k+1} \leq 2^{-m}$. Hence for the least k such that $S_k \geq 2^{-m}$, it must be that $S_k = 2^{-m}$. □

It is not possible to establish a procedure that given any computable martingale d, constructs a quick process machine P such that for all strings σ, $C^P(\sigma) \leq |\sigma| - \lfloor \log d(\sigma) \rfloor$. For example, consider the martingale defined by $d(\sigma) = 2^n$ if $\sigma = 1^n$ for some n, and $d(\sigma) = 0$ otherwise. In this example, for any n, $|1^n| - \lfloor \log d(1^n) \rfloor = 0$ and a process machine has only one description

of length 0. The problem with this example is that the martingale wins too quickly on the sequence 1^ω. We will restrict ourselves to martingales that do not win "too quickly" on any sequence.

Our objective is to determine an infinite set M, and build a quick process machine P such that for all strings σ such that $|\sigma| \in M$, $C^P(\sigma) \leq |\sigma| - \lfloor \log d(\sigma) \rfloor$. We will take M to be the range of a strictly increasing computable function. Given a computable martingale d with precision function r, we will say that a strictly increasing computable function m is a *selection function* for d if:

(i). For all $n \in \omega$, $m(n+1) \geq r(m(n)) + n + 2$.

(ii). For all $n \in \omega$, for all $\sigma \in 2^{<\omega}$, $|\sigma| = m(n) \Rightarrow d(\sigma) \leq 2^n$.

Proposition 6.2.9. *Let d be a computable martingale and let m be a selection function for d. There exists a quick process machine P such that for all $\sigma \in 2^\omega$ with $|\sigma| \in rng(m)$, $C^P(\sigma) \leq |\sigma| - \lfloor \log d(\sigma) \rfloor$.*

Proof. We will assume that $m(0) = 0$ as this simplifies the exposition of the proof, and this is all we will need later. We construct P using a quick process request function f. Let r be the precision function for d. At stage s in the construction we will define f on all strings ρ in the domain of f of length s.

At stage 0, we set $f(\lambda) = (\lambda, 0)$. At stage $s+1$, for all $\sigma \in 2^{<\omega}$ such that $|\sigma| = m(s)$, we let $E_\sigma = \{\rho \in \omega^s : f_1(\rho) = \sigma\}$. We make the following inductive assumptions on the construction:

(i). For all $\rho \in E_\sigma$, $f_2(\rho) \leq r(m(s))$.

(ii). $\displaystyle\sum_{\rho \in E_\sigma} 2^{-f_2(\rho)} = d(\sigma) 2^{-|\sigma|}$.

If $s = 0$, then $E_\lambda = \{\lambda\}$ and the inductive assumptions hold.

Choose $\sigma \in \{0,1\}^{m(s)}$. Let $\sigma_0, \ldots, \sigma_n$ be all extensions of σ of length $m(s+1)$. Now as d is a dyadic rational valued martingale, for each i, there is a unique finite set of integers A_i such that

$$\sum_{a \in A_i} 2^{-a} = d(\sigma_i) 2^{-|\sigma_i|}. \tag{6.2.2}$$

Hence by applying the martingale condition and the second inductive assumption:

$$\sum_{i=0}^{n} \sum_{a \in A_i} 2^{-a} = \sum_{i=0}^{n} d(\sigma_i) 2^{-|\sigma_i|} = d(\sigma) 2^{-|\sigma|} = \sum_{\rho \in E_\sigma} 2^{-|f_2(\rho)|}.$$

Further as $d(\sigma_i) \leq 2^{s+1}$ (the second condition for m to be a selection function), we have that for all i, $\min A_i \geq |\sigma_i| - s - 1 > r(m(s))$ (the last inequality follows from the first condition for m to be a selection function). Hence if $\rho \in E_\sigma$, then $f_2(\rho) < \min A_i$. Additionally, $\max A_i \leq r(m(s+1))$ because r is the precision function for d. Let $I = \{(i, a) : 0 \leq i \leq n, a \in A_i\}$ be an index set for the finite sets A_i. By Lemma 6.2.8, we can partition I into subsets S_ρ for each $\rho \in E_\sigma$, such that

$$\sum_{(i,a)\in S_\rho} 2^{-a} = 2^{-f_2(\rho)}. \tag{6.2.3}$$

Now for each such ρ, we define $f(\rho\langle i, a\rangle) = (\sigma_i, a)$ if $(i, a) \in S_\rho$. We repeat this step for all σ of length $m(s)$. Once this has been done for all such σ, we move to the next stage. Note that our first construction assumption is maintained because $f_2(\rho x) \leq \max\{a : (i, a) \in I\} \leq r(m(s+1))$. Our second construction assumption is maintained because of (6.2.2) and because every element of A_i generates a new node in the domain of f.

This completes the definition of f. First we need to verify that f is a quick process request function. The construction ensures that the domain of f is closed downwards, and that each node is finitely branching (as the index set I is always finite). Additionally f_1 is strictly increasing because m is strictly increasing, and f_2 is strictly increasing because $f_2(\rho) \leq r(m(|\rho|)) < f_2(\rho x)$ for any $x \in \omega$ with $\rho x \in \mathrm{dom}(f)$. Equation (6.2.3) ensures that we meet condition (ii) for f to be a strict process request function and this along with the fact that $f_2(\lambda) = 0$ implies we meet condition (ii) for f to be a strict process request function. From f, we can construct a quick process machine P using Theorem 6.2.6.

We will establish that P has the desired property. For any n, take any σ with $|\sigma| = m(n)$. Let A be the set of integers used for σ in (6.2.2). If $a \in A$, then there is some description τ such that $P(\tau) = \sigma$ and $|\tau| = a$. For (6.2.2) to hold, it must be that $|\sigma| - \lfloor \log d(\sigma) \rfloor \in A$. □

The above construction has an additional property that will be useful when we consider truth-table reductions.

Lemma 6.2.10. *Let $X \in 2^\omega$, let d be a computable martingale and let m be a selection function for d. Let P be the quick process machine created by an application of Proposition 6.2.9. Then the set $\{\tau : \exists n, c \in \omega\ [P(\tau) = X \upharpoonright m(n)$ and $d(X \upharpoonright m(n)) = 2^c]\}$ is a chain with respect to $\preceq$.*

Proof. Assume that there exist τ_1, τ_2 such that $P(\tau_1) = X \upharpoonright m(n_1)$ and $P(\tau_2) = X \upharpoonright m(n_2)$ with $n_1 \leq n_2$ and $d(X \upharpoonright m(n_1)) = 2^c$ for some $c \in \omega$.

By construction, there is some initial segment $\tau \preceq \tau_2$ such that $|P(\tau)| = m(n_1)$ and hence because P is a strict process machine we have that $P(\tau) = X \upharpoonright m(n_1)$.

At stage n_1 of the construction, we have that $X \upharpoonright m(n_1) = \sigma_i$ for some i. Now because $d(X \upharpoonright m(n_1)) = 2^c$, we have that $|A_i| = 1$. Hence τ_1 is the unique element in $2^{<\omega}$ such that $P(\tau_1) = X \upharpoonright m(n_1)$. Thus $\tau_1 = \tau \preceq \tau_2$. □

We can use Proposition 6.2.9 to build a quick process machine that turns high martingale values into short descriptions. While this proposition only works for martingales with selection functions, we will show that this is not a significant limitation for our purposes. First we consider the case of Schnorr randomness and weak randomness.

Proposition 6.2.11. *Let d be a computable martingale and let h be a computable order. There exists a computable martingale $\hat{d}$ and m, a selection function for $\hat{d}$, such that for all $X \in 2^\omega$:*

(i). $\forall n\ d(X \upharpoonright n) \geq h(n) \Rightarrow \forall n\ \hat{d}(X \upharpoonright m(n)) = 2^n$.

(ii). $\exists^\infty n\ d(X \upharpoonright n) \geq h(n) \Rightarrow \exists^\infty n\ \hat{d}(X \upharpoonright m(n)) = 2^n$.

Proof. We build $\hat{d}$ by adopting the betting strategy of d unless this strategy would force $\hat{d}(\sigma) > 2^n$ for some string σ of length $m(n)$. In this case, we restrain the betting so that $\hat{d}(\sigma) = 2^n$.

We construct $\hat{d}$ and m as follows. At stage 0, define $m(0) = 0$ and $\hat{d}(\lambda) = 1$. At stage $s + 1$, define:

$$m(s+1) = \max\{r(m(s)) + s + 2, \min\{x : h(x) \geq 2^{m(s)+s+1}\}\}$$

where r is the precision function for the martingale d. For all $\sigma \in 2^{<\omega}$ such that $m(s) \leq |\sigma| < m(s+1)$, we inductively define $\hat{d}(\sigma)$ by:

$$\hat{d}(\sigma i) = \begin{cases} 2^{s+1} & \text{if } d(\sigma i) \cdot \frac{\hat{d}(\sigma)}{d(\sigma)} \geq 2^{s+1} \\ 2\hat{d}(\sigma) - 2^{s+1} & \text{if } d(\sigma(1-i)) \cdot \frac{\hat{d}(\sigma)}{d(\sigma)} \geq 2^{s+1} \\ d(\sigma i) \cdot \frac{\hat{d}(\sigma)}{d(\sigma)} & \text{otherwise.} \end{cases} \tag{6.2.4}$$

We will say that the construction *restrains* the betting at σi, if the first case of (6.2.4) holds when $\hat{d}(\sigma i)$ is defined. Note that if $m(s) \leq |\sigma| < m(s+1)$ and the construction restrains the betting at σi, then for all X extending σi we have that $\hat{d}(X \upharpoonright m(s+1)) = 2^{s+1}$.

To verify the construction first note that the construction ensures that m is a selection function for $\hat{d}$. Take any $X \in 2^\omega$ such that for all n, $d(X \upharpoonright n) \geq h(n)$.

We will inductively show that for all n, $\hat{d}(X \upharpoonright m(n)) = 2^n$. First we have that $\hat{d}(X \upharpoonright m(0)) = \hat{d}(\lambda) = 2^0$. Assume that $\hat{d}(X \upharpoonright m(n)) = 2^n$. It must be that $d(X \upharpoonright m(n)) \leq 2^{m(n)}$ and $d(X \upharpoonright m(n+1)) \geq h(m(n+1))$. Combining these facts establishes that

$$d(X \upharpoonright m(n+1)) \cdot \frac{\hat{d}(X \upharpoonright m(n))}{d(X \upharpoonright m(n))} \geq h(m(n+1)) \cdot \frac{2^n}{2^{m(n)}} > 2^{n+1}.$$

The final inequality holds because we have that $h(m(n+1)) \geq 2^{m(n)+n+1}$ by the definition of m. This implies that there are $\sigma \in 2^\omega$ and $i \in \{0,1\}$ such that $\sigma i \prec X$ with $m(n) \leq |\sigma| < m(n+1)$, and such that $d(\sigma i) \cdot \frac{\hat{d}(\sigma)}{d(\sigma)} \geq 2^{n+1}$. If we take such a σ such that $|\sigma|$ is minimal then this implies that the construction restrains the betting at σi which means that $\hat{d}(X \upharpoonright m(n+1)) = 2^{n+1}$.

Now assume that there are infinitely many n such that $d(X \upharpoonright n) \geq h(n)$. To show that there are infinitely many n such that $\hat{d}(X \upharpoonright m(n)) = 2^n$, it is sufficient to show that for any c, there exists an n such that $d(X \upharpoonright m(n)) \geq 2^{n+c}$. From this it follows that the construction must restrain the betting at infinitely many initial segments of X, and thus for infinitely many n, $\hat{d}(X \upharpoonright m(n)) = 2^n$. Fix any c and choose n such that $m(n-1) \geq c$, then $d(X \upharpoonright m(n)) \geq 2^{m(n-1)+n} \geq 2^{c+n}$. □

Corollary 6.2.12. *(i). If $X \in 2^\omega$ is not Schnorr random, then there exists a quick process machine P and a computable function m such that $\exists^\infty n\, C^P(X \upharpoonright m(n)) \leq m(n) - n$.*

(ii). If X is not weakly random, then there exists a quick process machine P and a computable function m such that $\forall n\, C^P(X \upharpoonright m(n)) \leq m(n) - n$.

Proof. These results follow from combining Propositions 6.2.9 and 6.2.11. □

To establish a similar result for computable randomness, we need to vary the construction slightly. In the previous proposition, there was an order function that told us what value the martingale "should" have. In that situation we were able to fix a single value at each stage of the construction and prevent the martingale from exceeding this value. For the case of computable randomness, we cannot do this because the martingale may win very slowly. However, we can construct a suitable martingale $\hat{d}$ by restraining the betting the first time a martingale exceeds 2^n, for any n, along any path.

Proposition 6.2.13. *Let d be a computable martingale. There exists a computable martingale $\hat{d}$ and m, a selection function for $\hat{d}$, such that for all $X \in 2^\omega$, if d succeeds on X then $\forall c\, \exists n\, \hat{d}(X \upharpoonright m(n)) = 2^c$.*

Proof. We define m by $m(0) = 0$ and $m(s+1) = r(m(s)) + s + 2$ where r is the precision function for the martingale d.

We construct $\hat{d}$ as follows. At stage 0, define $\hat{d}(\lambda) = 1$. At stage $s+1$, for all $\sigma \in 2^{<\omega}$ such that $|\sigma| = m(s)$, let $n_\sigma = \min\{n : \text{for all } \tau \preceq \sigma,\ 2^n > \hat{d}(\tau)\}$. We inductively define $\hat{d}(\sigma')$ for all $\sigma' \succ \sigma$ with $m(s) \leq |\sigma'| < m(s+1)$ by:

$$\hat{d}(\sigma\tau i) = \begin{cases} 2^{n_\sigma} & \text{if } d(\sigma\tau i) \cdot \frac{\hat{d}(\sigma)}{d(\sigma)} \geq 2^{n_\sigma} \\ 2\hat{d}(\sigma\tau) - 2^{n_\sigma} & \text{if } d(\sigma\tau(1-i)) \cdot \frac{\hat{d}(\sigma)}{d(\sigma)} \geq 2^{n_\sigma} \\ d(\sigma\tau i) \cdot \frac{\hat{d}(\sigma)}{d(\sigma)} & \text{otherwise.} \end{cases} \tag{6.2.5}$$

Again we will say that the construction *restrains* the betting at σi, if the first case of (6.2.5) holds when $\hat{d}(\sigma i)$ is defined.

Fix $X \in 2^\omega$ such that d succeeds on X. We will show that for all c, there exists an n such that $\hat{d}(X \upharpoonright m(n)) = 2^c$. First $\hat{d}(\lambda) = 2^0$. Assume that $\hat{d}(X \upharpoonright m(n)) = 2^c$ where n is the minimal number with this property. Because d succeeds on X, there exists a least $k > m(n)$ such that

$$d(X \upharpoonright k) \cdot \frac{\hat{d}(X \upharpoonright m(n))}{d(X \upharpoonright m(n))} \geq 2^{c+1}.$$

The construction ensures that the betting is restrained at $X \upharpoonright k$ and further if n' is the least number such that $m(n') \geq k$ then we have that $\hat{d}(X \upharpoonright m(n')) = 2^{c+1}$. □

Corollary 6.2.14. *If $X \in 2^\omega$ is not computably random, there exists a quick process machine P such that for all c, there exists some n with $C^P(X \upharpoonright n) \leq n - c$.*

Proof. Combine Propositions 6.2.9 and 6.2.13. □

We are now able to provide characterisations of computable randomness, Schnorr randomness and weak randomness in terms of quick process machines. First we establish the following lemma.

Lemma 6.2.15. *If m is a strictly increasing computable function, d a computable martingale and $X \in 2^\omega$ such that for all n, $d(X \upharpoonright m(n)) \geq 2^n$, then X is not weakly random.*

Proof. Construct a computable martingale $\hat{d}$ from d by following the betting strategy of d except that whenever $\hat{d}$ first exceeds 2^n along some path, d banks half its capital and only bets with the remaining half. In this case if $m(2n) \leq x < m(2(n+1))$, then $\hat{d}(X \upharpoonright x) \geq 2^{n-1}$ and so X is not weakly random as $\hat{d}(X \upharpoonright x) \geq h(x)$, where $h(x) = \max\{2^{n-1} : m(2n) \leq x\}$. □

Theorem 6.2.16. *(i). A sequence $X \in 2^\omega$ is not computably random if and only if there exists a quick process machine P, such that $\forall c, \exists n\, C^P(X \upharpoonright n) \leq n - c$.*

(ii). A sequence $X \in 2^\omega$ is not Schnorr random if and only if there exists a quick process machine P, and a strictly increasing computable function m, such that $\exists^\infty n\, C^P(X \upharpoonright m(n)) \leq m(n) - n$.

(iii). A sequence $X \in 2^\omega$ is not weakly random if and only if there exists a quick process machine P, and a strictly increasing computable function m such that $\forall n\, C^P(X \upharpoonright m(n)) \leq m(n) - n$.

Proof. The left to right directions are just Corollaries 6.2.12 and 6.2.14 restated. For the right to left direction, given such a quick process machine P, we use Proposition 6.2.1 to build a martingale d. Now $d(X \upharpoonright n) \geq 2^{n - C^P(X \upharpoonright n)}$. Hence if $\forall c\, \exists n\, C^P(X \upharpoonright n) \leq n - c$, we have that X is not computably random.

If for some strictly increasing computable m, $\exists^\infty n\, C^P(X \upharpoonright m(n)) \leq m(n) - n$, then let $h(x) = \max\{\{0\} \cup \{2^n : m(n) \leq x\}\}$. For infinitely many n, $d(X \upharpoonright m(n)) \geq 2^n = h(m(n))$ and so X is not Schnorr random.

Finally assume that for some strictly increasing computable m, $\forall n\ C^P(X \upharpoonright m(n)) \leq m(n) - n$. In this case, for all n, $d(X \upharpoonright m(n)) \geq 2^n$ and so by Lemma 6.2.15, X is not weakly random. □

6.3 Quick process machines and truth-table functionals

There is a simple technique to translate between truth-table functionals and quick process machines that allows further characterisations of computable randomness, Schnorr randomness and weak randomness in terms of truth-table reducibility. However, for these characterisations to hold, we need to be careful about how we define the use of a truth-table reduction. Nerode observed that a truth-table reduction can be regarded as a Turing reduction that is total on all oracles [68]. We will take this as our definition of a truth-table reduction. Given a truth-table reduction Φ we now define $\phi^X(n)$ to be the largest query made of the oracle X during the computation of $\Phi^X(m)$ for any $m \leq n$.

Theorem 6.3.1. *(i). A sequence $X \in 2^\omega$ is not computably random if and only if there exist a truth-table functional Φ, and a sequence $Y \in 2^\omega$, such that $\Phi^Y = X$ and $\forall c\, \exists n\, \phi^Y(n) \leq n - c$.*

(ii). A sequence $X \in 2^\omega$ is not Schnorr random if and only if there exist a truth-table functional Φ, a sequence $Y \in 2^\omega$, and a strictly increasing computable function m such that $\Phi^Y = X$ and $\exists^\infty n\, \phi^Y(m(n)) \leq m(n) - n$.

(iii). A sequence $X \in 2^\omega$ is not weakly random if and only if there exist a truth-table functional Φ, a sequence $Y \in 2^\omega$, and a strictly increasing computable function m such that $\Phi^Y = X$ and $\forall n\ \phi^Y(m(n)) \leq m(n) - n$.

Proof. The right to left direction of the above statements can be established by constructing a martingale d from a truth-table functional Φ as follows. Let $d(\sigma) = \mu\{X : \Phi^X \succeq \sigma\} \cdot 2^{|\sigma|}$. The fact that Φ is total on all oracles makes d computable. Now if $\Phi^Y = X$, then we have that $d(X \upharpoonright n) \geq 2^{n-\phi^Y(n)}$. The right to left direction for (i) and (ii) follow immediately. For (iii) an application of Lemma 6.2.15 is needed.

To establish the left to right direction, given a quick process machine P, we define a truth-table functional Φ that computes $\Phi^X(n)$ by finding the shortest initial segment of τ of X such that $|P(\tau)| > n$ and setting $\Phi^X(n)$ to be the nth bit of this output.

Now if X is not computably random, then by Corollary 6.2.14, there is some quick process machine P such that $\forall c\, \exists n\ C^P(X \upharpoonright n) = n - c$. Further by applying Lemma 6.2.10, for all c, we can take some τ_c with $P(\tau_c) = X \upharpoonright (|\tau_c|+c)$ and such that $\{\tau_c : c \in \omega\}$ is a chain. Let $Y = \bigcup_c \tau_c$. Thus $\Phi^Y = X$ and $\phi^Y(|P(\tau_c)| - 1) \leq |P(\tau_c)| - c$. If X is Schnorr random or weakly random then the proof proceeds similarly. □

The author would like to note that this characterisation of computable randomness in terms of truth-table reducibility has been independently arrived at by Laurent Bienvenu and Chris Porter.

Chapter 7

Non-Computable Measures

This chapter is joint work with Joseph Miller of the University of Wisconsin-Madison. It is based on research undertaken with Miller at the University of Heidelberg in July 2009. The work in this chapter has been accepted for publication in the Transactions of the American Mathematical Society.

7.1 Defining randomness

Let X be an element of Cantor space and μ a Borel probability measure on Cantor space. What should it mean for X to be random with respect to μ? If μ is a computable measure, then early work of Levin showed that μ-randomness can be seen as essentially a variant on randomness for Lebesgue measure [94]. This leaves the question of how to define randomness if μ is non-computable. We will show that the two approaches that have previously been used to define μ-randomness for non-computable measures μ, are equivalent. Later, in Theorem 7.4.12, we will provide another characterisation of μ-randomness using the enumeration degrees.

We would like to find a natural generalisation of Martin-Löf randomness to non-computable probability measures. One approach is to generalise Martin-Löf tests. This approach immediately runs into the difficult question of what sort of oracle access a test should have. It is reasonable to expect that a test for a measure μ should be able to compute the μ-measure of any basic clopen set. However, there are continuum many Borel probability measures on Cantor space, so in order to make these measures accessible to the techniques of computability theory, we will make use of some basic concepts of computable analysis. We will define all the concepts we need. For further background on computable analysis, the reader is referred to Weihrauch [92], who gives a modern development of the subject. Classical computability theory studies

Cantor space (2^ω) and Baire space (ω^ω). The main idea behind computable analysis is to transfer the notions of computability theory to other structures via representations of those structures. If $\mathcal{S}$ is a set, a *representation* of $\mathcal{S}$ is just a surjective function (possibly partial) $\rho\colon 2^\omega \to \mathcal{S}$. The representation induces a computability-theoretic structure on $\mathcal{S}$. We will also use the word "representation" in another, less standard, sense. If $R \in 2^\omega$ and $\rho(R) = x$, we call R a *representation* of x.

We will take $\mathcal{P}(2^\omega)$ to be the set of all Borel probability measures on Cantor space. We will let $\rho : 2^\omega \to \mathcal{P}(2^\omega)$ be a representation of $\mathcal{P}(2^\omega)$. In Section 7.2 we will give a detailed definition of such a representation ρ but for now it is enough to specify that if $\rho(R) = \mu$, then we can uniformly in R compute the μ measure of any basic clopen set in Cantor space.

As we access measures via representations, one approach is to define randomness in terms of representations. The following definitions, while not identical, are equivalent to that of Reimann [72] and Reimann and Slaman [73, 74].

Definition 7.1.1. Let $\mu \in \mathcal{P}(2^\omega)$ and let $R \in 2^\omega$ be a representation of μ.

(i). An *R-test* is a uniform (in R) sequence $\{V_i\}_{i\in\omega}$ of $\Sigma^0_1(R)$ sets such that $\mu(V_i) \leq 2^{-i}$.

(ii). $X \in 2^\omega$ *passes* an R-test if $X \notin \bigcap_i V_i$.

(iii). $X \in 2^\omega$ is *R-random* if it passes all R-tests.

A universal R-test exists for the same reason that a universal Martin-Löf test exists. Given R, we would like to enumerate all R-tests by enumerating all (uniform in R) sequences of R-c.e. sets, halting any enumeration if it would exceed the measure bound. There is a small technical obstruction. Using R, we can compute a sequence, approximating from above, the μ-measure of a basic clopen set. Hence, we can pause an enumeration until a stage when our approximation from above guarantees that we can add the next element without exceeding the measure bound. Note that this could cause a problem if some test $\{V^R_n\}_{n\in\omega}$ had $V^R_i = 2^{-i}$ for some i. The enumeration of this test could be paused forever. However, in this case, the test $\{V^R_{n+1}\}_{n\in\omega}$ defines the same null set and avoids this problem. This shows that we can (essentially) enumerate all R-tests, uniformly in R, so we can build a universal R-test. Even better, because the construction is uniform, there is a uniform sequence of c.e. sets U_n such that if $U^R_n = \{[\tau]\colon \langle\tau,\sigma\rangle \in U_n \text{ and } \sigma \prec R\}$, then $\{U^R_n\}_{n\in\omega}$ is a universal R-test. Call $\{U_n\}_{n\in\omega}$ a *universal oracle Martin-Löf test*.

As noted by Reimann, the problem with Definition 7.1.1 is that it is dependent on the representation. Given any measure, it is possible to encode any sequence into some representation of that measure. Hence for all $\mu \in \mathcal{P}(2^\omega)$ and all $X \in 2^\omega$, there is a representation R of μ such that X is not R-random. A natural way to overcome this problem is with the following definition.

Definition 7.1.2. A sequence $X \in 2^\omega$ is *μ-random* if there exists a representation R of μ such that X is R-random.

Our goal is to show that, at least in Cantor space, this definition gives the same class of randoms for a measure as does the concept of a *uniform test*. Uniform tests are an alternative approach to randomness for non-computable measures. They were introduced by Levin and developed by Gács, and also by Hoyrup and Rojas [38, 40, 57]. While uniform tests can be applied to general probability spaces, in this chapter, we will only be concerned with Cantor space. We take $\{B_i\}_{i\in\omega}$ to be an enumeration of open balls in $\mathcal{P}(2^\omega)$. The details of this enumeration will be provided in the following section.

Definition 7.1.3. Consider a function $t\colon \mathcal{P}(2^\omega) \times 2^\omega \to \mathbb{R}^{\geq 0} \cup \{\infty\}$.

(i). The *under-graph* of t is $\{(\mu, X, r)\colon t(\mu, X) > r\}$. We say that the under-graph of t is *c.e. open*, if it is equal to $\bigcup_{\langle i,\sigma,q\rangle\in W} B_i \times [\sigma] \times [0, q)$ for some c.e. set $W \subseteq \omega \times 2^{<\omega} \times \mathbb{Q}$.

(ii). We call t a *uniform test* if its under-graph is c.e. open and for every $\mu \in \mathcal{P}(2^\omega)$ we have $\int t(\mu, X)\, d\mu \leq 1$.

(iii). $X \in 2^\omega$ passes a test t for a measure μ if $t(\mu, X)$ is bounded.

(iv). $X \in 2^\omega$ is *μ-random for uniform tests* if it passes all tests for measure μ.

By a straightforward theorem of Gács, later refined by Hoyrup and Rojas, it is sufficient to consider a single universal uniform test.

Definition 7.1.4. A uniform test t is *universal* if for all uniform tests t' there is a constant $c > 0$ such that for all $\mu \in \mathcal{P}(2^\omega)$ and $X \in 2^\omega$, we have $t(\mu, X) \geq c \cdot t'(\mu, X)$.

Theorem 7.1.5 (Gács; Hoyrup and Rojas [38, 40])**.** *There exists a universal uniform test.*

The following theorem will establish that these two approaches, one based on representations and the other on uniform tests, are equivalent.

Theorem 7.1.6. *For any measure μ and $X \in 2^\omega$ we have that X is μ-random if and only if X is μ-random for uniform tests.*

Before proving this theorem, we need to take a more detailed look at $\mathcal{P}(2^\omega)$ and at representations of probability measures.

7.2 Probability measures

In this chapter, we will restrict our investigation to Borel probability measures on Cantor space. Let μ be such a measure. As observed in Section 1.2, we can identify μ with the values it takes on the basic clopen sets $[\sigma]$, where $\sigma \in 2^{<\omega}$. Hence we will often think of μ as function $\mu\colon 2^{<\omega} \to [0,1]$ and write $\mu(\sigma)$ instead of $\mu([\sigma])$.

Any measure μ such that $\int d\mu \leq 1$ can be thought of as an element $\alpha \in [0,1]^\omega$ where $\alpha(\langle\sigma\rangle) = \mu(\sigma)$. We define the following two subspaces of $[0,1]^\omega$:

(i). $\mathcal{M}(2^\omega) = \{\alpha \in [0,1]^\omega\colon (\forall \sigma \in 2^{<\omega})\ \alpha(\langle\sigma\rangle) = \alpha(\langle\sigma 0\rangle) + \alpha(\langle\sigma 1\rangle)\}$.

(ii). $\mathcal{P}(2^\omega) = \{\alpha \in \mathcal{M}(2^\omega)\colon \alpha(\langle\lambda\rangle) = 1\}$.

Our primary space of concern is $\mathcal{P}(2^\omega)$, the space of all probability measures on Cantor space. The space $\mathcal{M}(2^\omega)$, all measures μ such that $\int d\mu \leq 1$, will be of interest when we investigate neutral measures. We can regard $\mathcal{P}(2^\omega)$ and $\mathcal{M}(2^\omega)$ as compact subspaces of the topological vector space $\mathbb{R}^\omega$ with the topology provided by the metric

$$d(\alpha,\beta) = \sum_{\sigma\in 2^{<\omega}} 2^{-\langle\sigma\rangle}|\alpha(\langle\sigma\rangle) - \beta(\langle\sigma\rangle)|. \tag{7.2.1}$$

There is an alternative approach to topologizing the space $\mathcal{P}(2^\omega)$. We can topologize this space so that a sequence of measures $\{\mu_n\}_{n\in\omega}$ has limit μ if and only if $\mu_n(B) \to \mu(B)$ for all Borel sets B whose boundary has μ-measure 0. This topology is known as the *weak topology*.

We can view 2^ω as a metric space by using the metric

$$d_{2^\omega}(A,B) = \begin{cases} 0 & \text{if } A = B, \\ 2^{-i} & \text{where } i = \min\{A \triangle B\} \text{ otherwise.} \end{cases} \tag{7.2.2}$$

Cantor space is a compact and separable metric space under d_{2^ω}. This fact implies that the weak topology on $\mathcal{P}(2^\omega)$ is compact and further that it is metrizable using the Prohorov metric [9]. Given $\mu, \nu \in \mathcal{P}(2^\omega)$, the *Prohorov metric*

$p(\mu, \nu)$ is defined to be the infimum of those positive ϵ for which the follow two inequalities hold for all Borel subsets A of 2^ω:

$$\mu(A) \leq \nu(A^\epsilon) + \epsilon, \quad \nu(A) \leq \mu(A^\epsilon) + \epsilon,$$

where $A^\epsilon = \{X \in 2^\omega : (\exists Y \in A)\, d(X, Y) < \epsilon\}$.

Note that under the metric d_{2^ω}, if $A \subseteq 2^\omega$ and $\epsilon = 2^{-n}$, then $A^\epsilon = \bigcup\{[\sigma] : |\sigma| = n \wedge [\sigma] \cap A \neq \emptyset\}$. Using this observation, the following lemma is easy to show.

Lemma 7.2.1. *The Prohorov metric and the metric defined in* (7.2.1) *induce the same topologies on* $\mathcal{P}(2^\omega)$.

Proof. Let p be the Prohorov metric and d the metric as defined in (7.2.1). Pick any $\mu \in \mathcal{P}(2^\omega)$ and positive real number δ. Now consider the open ball using the Prohorov metric $B_p(\mu, \delta)$. Choose $n \in \omega$ such that $2^{-n} < \delta$. Choose ϵ so that for all $\nu \in B_d(\mu, \epsilon)$ (the open ball using the metric d) we have that for all σ of length n , $|\mu(\sigma) - \nu(\sigma)| < 2^{-2n}$. Then if $A \subseteq 2^\omega$ is Borel and $\nu \in B_d(\mu, \epsilon)$ we have that:

$$\mu A \leq \mu A^{2^{-n}} = \sum_{\substack{|\sigma|=n \\ [\sigma]\cap A\neq\emptyset}} \mu(\sigma) < \sum_{\substack{|\sigma|=n \\ [\sigma]\cap A\neq\emptyset}} (\nu(\sigma) + 2^{-2n}) = \nu A^{2^{-n}} + 2^{-n}.$$

Similarly $\nu A < \mu A^{2^{-n}} + 2^{-n}$ and so $p(\nu, \mu) \leq 2^{-n}$. Thus $B_d(\mu, \epsilon) \subseteq B_p(\mu, \delta)$.

For the other direction, consider $B_d(\mu, \delta)$. Let $\epsilon = 2^{-n}$ for some n such that $2^{-n+1} < \delta$. Now if $\langle\sigma\rangle \leq n$ we have that $|\sigma| \leq n$ so that $[\sigma]^\epsilon = [\sigma]$. If $\nu \in B_p(\mu, \epsilon)$ we have that $|\mu(\sigma) - \nu(\sigma)| < \epsilon$ and so

$$d(\mu, \nu) \leq \sum_{\langle\sigma\rangle\leq n} 2^{-\langle\sigma\rangle} |\mu(\sigma) - \nu(\sigma)| + 2^{-n} \leq 2^{-n+1}.$$

Thus $B_p(\mu, \epsilon) \subseteq B_d(\mu, \delta)$. □

We will treat the space $\mathcal{P}(2^\omega)$ of probability measures as a computable metric space. These were introduced by Lacombe [53], though our presentation is influenced by [92]. A *computable metric space* is a triple $(\mathcal{X}, \mathcal{Q}, d)$ where $\mathcal{X}$ is a complete separable metric space, $\mathcal{Q}$ is an enumeration of a countable dense subset of $\mathcal{X}$, and d is a metric computable on the elements of $\mathcal{Q}$. Given a computable metric space $(\mathcal{X}, \mathcal{Q}, d)$, with $\mathcal{Q} = \{q_1, q_2, \ldots\}$, the standard fast Cauchy representation of $(\mathcal{X}, \mathcal{Q}, d)$ is $\rho_C \colon 2^\omega \to \mathcal{X}$ and is defined by $\rho_C(0^{n(0)}10^{n(1)}10^{n(2)}1\ldots) = x$ if $(\forall i \in \omega)\ \ d(x, q_{n(i)}) \leq 2^{-i}$. Note that this representation ρ_C is a partial function.

In order to work with $\mathcal{P}(2^\omega)$ as a computable metric space, we need an enumeration of a countable dense subset of $\mathcal{P}(2^\omega)$ on which the metric is computable. For any $X \in 2^\omega$, we define the Dirac measure δ_X by

$$\delta_X(\sigma) = \begin{cases} 1 & \text{if } \sigma \prec X, \\ 0 & \text{otherwise.} \end{cases}$$

We will take our dense subset $\mathcal{Q}$ to be those measures that concentrate on finitely many sequences each containing finitely many 1s, and take rational values at those points. In other words, $\mu \in \mathcal{Q}$ if and only if $\mu = \sum_{i=1}^n a_i \delta_{\sigma_i 0^\omega}$, where $\sigma_1, \ldots, \sigma_n \in 2^{<\omega}$ and $a_1, \ldots, a_n$ are positive rationals such that $\sum_{i=1}^n a_i = 1$.

Fix an enumeration of these measures $m_1, m_2, \ldots$. At times, in order to avoid subscripts we will write $m(i)$ for m_i. Note that $d(m_i, m_j)$ is computable in i and j and that the open balls $B(m_i, 2^{-n})$ form an enumerable basis for the topology on $\mathcal{P}(2^\omega)$. We will call these the *rational open balls* and take B_i to be the ith such ball in some fixed enumeration. Let $\overline{B_i}$ be the closure of the rational open ball B_i.

Instead of using the standard fast Cauchy representation of $\mathcal{P}(2^\omega)$, we want to use the fact that $\mathcal{P}(2^\omega)$ is compact to define a representation that has some additional useful properties. Reimann showed that there is a computable surjection $\rho\colon P \to \mathcal{P}(2^\omega)$, where P is a Π^0_1 subset of 2^ω [72]. Our approach is similar to that of Reimann. It can also be seen as a generalisation of Turing's approach to coding the reals via overlapping intervals [85, 86], for which he acknowledges Brouwer.

To define our representation, we will first define a Turing functional φ such that φ^X is total for all oracles X, and for all $n \in \omega$,

$$d(m(\varphi^X(n)), m(\varphi^X(n+1))) \leq 2^{-n}.$$

Thus for any oracle X, the sequence $m(\varphi^X(0)), m(\varphi^X(1)), \ldots$ is Cauchy and so converges (because $\mathcal{P}(2^\omega)$ is complete). Thus we can define a total function $\rho\colon 2^\omega \to \mathcal{P}(2^\omega)$ by $\rho(X) = \lim_s m(\varphi^X(s))$.

We define φ^X inductively as follows. At stage s we will define $\varphi^X(s)$ for all oracles X. At stage 0, we will define $\varphi^X(0) = 1$ for all oracles X. Note that $B(m_1, 2^0) = \mathcal{P}(2^\omega)$.

At stage $s+1$, for all strings τ such that $\varphi^\tau(s)$ is defined but $\varphi^{\tau'}(s)$ is not defined if τ is a strict initial segment of τ' do the following. Let $x = \varphi^\tau(s)$. We claim that we can uniformly compute a finite open covering $\{B(m(n_1), 2^{-s-1}), \ldots, B(m(n_k), 2^{-s-1})\}$ of $\overline{B(m(x), 2^{-s})}$ with $m(n_i) \in B(m(x), 2^{-s})$. Given this

claim, we can determine a disjoint collection of cylinders $\{[\tau_1],\ldots,[\tau_k]\}$ that covers 2^ω and define $\varphi^{\tau\tau_i}(s+1) = n_i$. This completes the definition of φ. Our representation will be the continuous function $\rho\colon 2^\omega \to \mathcal{P}(2^\omega)$ defined by $\rho(X) = \lim_s m(\varphi^X(s))$.

To establish the claim, we build a finite set of probability measures (all in $\mathcal{Q}$) by adding together measures of the form $2^{-s-4}\cdot\delta_{\sigma 0^\omega}$ where $|\sigma| = s+4$. Define:

$$S_s = \{m_i \in \mathcal{Q} : m_i = \sum_{|\sigma|=s+4} a_\sigma 2^{-s-4}\delta_{\sigma 0^\omega} \text{ for some } a_\sigma \in \omega\}.$$

Then we take $\{B(m_i, 2^{-s-1}) : m_i \in S_s \wedge d(m(x), m_i) < 2^{-s}\}$ as our covering. To show that this is in fact a covering, take any $\mu \in \overline{B(m(x), 2^{-s})}$. Let $\nu = (m(x)+3\mu)/4$ so $d(m(x),\nu) = 3d(m(x),\mu)/4$ and $d(\mu,\nu) = d(m(x),\nu)/4$. We can easily find $m_i \in S$ such that $d(\nu, m_i) < 2^{-s-2}$. Hence the ball $B(m_i, 2^{-s-1})$ is in our covering and this ball contains μ.

Lemma 7.2.2. *The function ρ is total, surjective, and for all $X \in 2^\omega$, $\rho^{-1}(\rho(X))$ is a $\Pi^0_1(X)$ class.*

Proof. We have already seen that ρ is total. To see that it is surjective, take any $\mu \in \mathcal{P}(2^\omega)$. As ρ is continuous, for all n, the set $F_n = \{X \in 2^\omega\colon d(\rho(X),\mu) \leq 2^{-n}\}$ is closed. The construction ensures that it is nonempty, so by compactness there is an $X \in \bigcap_i F_i$. Clearly, $\rho(X) = \mu$.

If $X \in 2^\omega$, then $\rho^{-1}(X)$ is a $\Pi^0_1(\rho(X))$ class because $Y \in \rho^{-1}(X)$ if and only if, for all n,

$$d(m(\varphi^X(n)), m(\varphi^Y(n))) \leq 2^{-n+2}. \qquad \square$$

Because ρ is surjective, it is a representation of $\mathcal{P}(2^\omega)$. Furthermore, it is equivalent to the standard fast Cauchy representation ρ_C in the sense that we can computably translate between them. If $\rho(R) = \mu$, then $\varphi(R)$ is just a fast Cauchy representation of μ (i.e., the sequence $m(\varphi^R(0)), m(\varphi^R(1)),\ldots$). On the other hand if $\rho_C(S) = \mu$, then we can compute from S, a sequence R, such that $\rho(R) = \mu$ by running through the construction of φ. We start with $\tau_0 = \lambda$. At each stage $s+1$ we compute a sufficiently close approximation to μ so that we can choose $\tau_{s+1} \succ \tau_s$ with $\mu \in B(m(\varphi^{\tau_{s+1}}(s+1)), 2^{-s-1})$.

We will need one additional nice property of ρ: that the inverse image of the closure of an rational open ball is also a Π^0_1 class.

Lemma 7.2.3. *If $B(s,q)$ is an rational open ball in $\mathcal{P}(2^\omega)$, then $\rho^{-1}(\overline{B(s,q)})$ is a Π^0_1 subset of 2^ω.*

Proof. First we show that $X \in \rho^{-1}(\overline{B(s,q)})$ if and only if, for all i, j,

$$\rho(X) \in B(m_i, q_j) \text{ implies } d(m_i, s) \leq q + q_j.$$

If for some i, j, $\rho(X) \in B(m_i, q_j)$ and $d(m_i, s) > q + q_j$ then $q + q_j < d(s, m_i) \leq d(s, \rho(X)) + d(\rho(X), m_i) < d(s, \rho(X)) + q_j$ and hence $X \notin \rho^{-1}(\overline{B(s,q)})$. Conversely, assume that $\rho(X) \in B(m_i, q_j)$ implies $d(m_i, s) \leq q + q_j$, for all i, j. Then $d(\rho(X), s) \leq d(\rho(X), m_i) + d(s, m_i) < q + 2q_j$. As $\rho(X)$ is contained in arbitrarily small rational open balls, we have $d(\rho(X), s) \leq q$. Now the predicate $\rho(X) \in B(m_i, q_j)$ implies $d(m_i, s) \leq q + q_j$ is Π^0_1. This is true as $\rho(X) \in B(m_i, q_j)$ and $d(m_i, s) > q + q_j$ are both Σ^0_1. □

We are now ready to prove Theorem 7.1.6. We start with the easier direction.

Lemma 7.2.4. *If $X \in 2^\omega$ is μ-random, then it is μ-random for uniform tests.*

Proof. Let W be a c.e. set defining the under-graph of a universal uniform test t. Assume that X is not μ-random for uniform tests. Let R be any representation of μ. Build an R-test as follows. Let

$$V_n = \{X \in 2^\omega : t(\mu, X) > 2^n\}.$$

Immediately we have that if $t(\mu, X) = \infty$, then $X \in \bigcap_{n\in\omega} V_n$. To show that $\{V_n\}_{n\in\omega}$ is an R-test, first observe that $2^n \mu(V_n) \leq \int t(\mu, X)\, d\mu \leq 1$, so $\mu(V_n) \leq 2^{-n}$. Secondly, we have $X \in V_n$ if and only if $t(\mu, X) > 2^n$ if and only if, for some $\langle i, \sigma, q\rangle \in W$, we have $\mu \in B_i$, $X \in [\sigma]$ and $q > 2^n$. The set $\{i : \mu \in B_i\}$ is c.e. in R because if μ is contained in the rational open ball $B(m_i, 2^{-n})$ then $d(\mu, m_i) < 2^{-n}$ and $d(\mu, m_i)$ is computable in R. Hence the V_n are uniformly $\Sigma^0_1(R)$ sets, and X is not R-random. As this holds for any representation of μ, we have proved that X is not μ-random. □

For the other direction, we have to show that the failure of μ-randomness can be detected in a uniform way. Not surprisingly, we do this using the universal oracle Martin-Löf test $\{U_n\}_{n\in\omega}$ from above.

Lemma 7.2.5. *If X is not μ-random, then for all n, there exists an m, such that for all $R \in \rho^{-1}(B(\mu, 2^{-m}))$, $X \in U^R_n$.*

Proof. Take any $\mu \in \mathcal{P}(2^\omega)$. Assume that for some n, for all m, there is an R_m such that $\rho(R_m) \in B(\mu, 2^{-m})$ and $X \notin U^{R_m}_n$. Consider the tree $\{\sigma \in 2^{<\omega} : (\exists m)\ \sigma \preceq R_m \upharpoonright m\}$. This tree is infinite so it has an infinite path A. For all i, $\rho([A \upharpoonright i])$ includes the ρ-image of infinitely many R_m. Thus as $\rho([A \upharpoonright i])$ is closed, it contains μ. Hence $\rho(A) = \mu$. But note that $X \notin U^A_n$, or otherwise $X \in U^{R_m}_n$ for some m. Thus X must be μ-random. □

Proof of Theorem 7.1.6. Lemma 7.2.4 shows that if X is not μ-random for uniform tests, then X is not μ-random. To establish the other direction, we will construct a test f as follows. For all i, let $K_i = \rho^{-1}(\overline{B_i})$. By Lemma 7.2.3, K_i is a Π^0_1 class. So if X enters U^R_n for all R in K_i, compactness ensures that we can determine this at some finite stage. If this occurs then we can increase the value on some open set containing X for all measures in B_i. Let $S_1, S_2, \ldots$ be an enumeration of all finite sets of finite strings. The under-graph of our test f will be enumerated by the following c.e. set:

$$W = \{\langle i, \sigma, 2^n\rangle \colon (\exists j)(\exists s)\ K_i[s] \subseteq \bigcup_{\tau\in S_j} [\tau] \text{ and } [\sigma] \subseteq \bigcap_{\tau\in S_j} U^\tau_{2n}[s]\}.$$

Given any $\mu \in \mathcal{P}(2^\omega)$ we will show that $\int f(\mu, X)\ d\mu \leq 1$, so f is a uniform test. Take $R \in \rho^{-1}(\mu)$. Take any n and any $X \notin U^R_{2n}$. Given any i, if $\mu \in B_i$ then $R \in K_i$. So if S_j covers K_i, then for some $\tau \in S_j$, $\tau \prec R$. Thus $X \notin \bigcap_{\tau\in S_j} U^\tau_{2n}$. This implies that $\langle i, \sigma, 2^n\rangle \notin W$ for any $\sigma \prec X$, and so $f(\mu, X) \leq 2^{n-1}$. Hence $f(\mu, X) \leq \max\{2^n \colon X \in U^R_{2n}\}$ and so,

$$\int f(\mu, X)\ d\mu \leq \sum_{i=1}^{\infty} 2^i \mu(U^R_{2i}) \leq \sum_{i=1}^{\infty} 2^i 2^{-2i} = 1.$$

Now assume that X is not μ-random. Fix n. By Lemma 7.2.5, there is an m such that if $R \in \rho^{-1}(B(\mu, 2^{-m}))$, then $X \in U^R_{2n}$. Let $\overline{B_i}$ be a closed rational ball with $\mu \in \overline{B_i} \subseteq B(\mu, 2^{-m})$. Now because for all $R \in K_i$ we have $X \in U^R_{2n}$, the set $C = \{\tau \in 2^{<\omega} \colon X \in U^\tau_{2n}\}$ is an open covering of K_i. Hence there is a finite sub-covering S of C and a stage s such that S covers $K_i[s]$. So there is a σ with $X \in [\sigma] \subseteq \bigcap_{\tau\in S} U^\tau_{2n}[s]$. Thus $\langle i, \sigma, 2^n\rangle \in W$ and consequently $f(\mu, X) > 2^n$. This holds for all n, so $f(\mu, X) = \infty$. Therefore, X is not μ-random for uniform tests. □

7.3 Neutral measures

Our main goal in the remainder of the chapter will be to come to a better understanding of (*weakly*) *neutral measures*. Levin proved the existence of neutral measures [57]. However, the term "neutral measure" was introduced later by Gács [38]. As we will see in Section 7.4, where we derive several facts about neutral measures, it is often enough to assume a weaker property.

Definition 7.3.1. Let μ be a measure.

(i). μ is *neutral* for a uniform test t if $(\forall X)\ t(\mu, X) \leq 1$.

(ii). μ is a *neutral measure* if it is neutral for some universal test.

(iii). μ is *weakly neutral* if every sequence is μ-random.

Since a constant multiple of a universal test is also a universal test, and any two universal tests majorize each other up to a multiplicative constant, we can restate the second definition: μ is a *neutral measure* if and only if $(\exists c)(\forall X)\ t(\mu, X) \leq c$, where t is any universal test.

It is immediate that a neutral measure μ is weakly neutral. Indeed, this is the property that makes neutral measures seem so unlikely. One might think that it is impossible for every sequence to be μ-random, since if we have access to μ, we should be able to build a Martin-Löf μ-test covering *something*. Indeed, this is the case; in Lemma 7.4.1 we will see that for every representation R of μ, there is a non-R-random sequence. But if μ is weakly neutral, no sequence will be de-randomised by every representation of μ.

We start by giving a proof that neutral measures exist.

Theorem 7.3.2 (Levin [57])**.** *For any uniform test, there is a measure neutral for it.*

Our proof is fundamentally equivalent to that given by Levin [57] and Gács [38]. However, we will make use of the Kakutani fixed point theorem instead of Sperner's Lemma. Our exposition of the proof will also make clear some computability properties of neutral measures. A set $C \subseteq \mathbb{R}^n$ is *convex* if for all $\mu, \nu \in C$ and $x \in [0, 1]$, we have $x\mu + (1 - x)\nu \in C$.

Theorem 7.3.3 (Kakutani [45])**.** *If $\mathcal{S}$ is a nonempty compact convex subset of $\mathbb{R}^n$ and $\phi\colon \mathcal{S} \to \mathcal{S}$ a multi-valued map with closed graph such that for all $x \in \mathcal{S}$, the ϕ-image of x is convex and nonempty, then there is an $x \in \mathcal{S}$ such that $x \in \phi(x)$.*

A fixed point theorem is useful because we can use the uniform test to define a map from measures to measures. Given t, we define $\hat{t}\colon \mathcal{P}(2^\omega) \to \mathcal{M}(2^\omega)$ by letting $\hat{t}(\mu)$ (which we will write as $\hat{t}\mu$) be the measure that takes the following values on the basic clopen sets:

$$\hat{t}\mu(\sigma) = \frac{1}{2}\int_{[\sigma]} t(\mu, X)\ d\mu + \frac{1}{2}2^{-|\sigma|}.$$

We can partially order $\mathcal{M}(2^\omega)$ by $\mu \leq \nu$ if $\mu(\sigma) \leq \nu(\sigma)$ for all σ. In this case we will say that ν majorizes μ.

Lemma 7.3.4. *If $\mu \geq \hat{t}\mu$, then μ is neutral for $\frac{1}{2} \cdot t$.*

Proof. Assume μ is not neutral for $\frac{1}{2} \cdot t$, then for some $X \in 2^\omega$, $t(\mu, X) > 2$. This implies that there is a $\sigma \in 2^{<\omega}$ with $X \in [\sigma]$ such that $t(\mu, Y) > 2$ for all $Y \in [\sigma]$. But this would mean that

$$\hat{t}\mu(\sigma) \geq \frac{1}{2}\int_{[\sigma]} 2\ d\mu + 2^{-|\sigma|-1} > \mu(\sigma).$$

This is a contraction because μ majorizes $\hat{t}\mu$. □

Consider the sets $F_n = \{\mu \in \mathcal{P}(2^\omega) \colon (\forall \sigma)\ [|\sigma| = n \Rightarrow \hat{t}\mu(\sigma) \leq \mu(\sigma)]\}$. Note that $\mathcal{P}(2^\omega) = F_0$ and $F_{n+1} \subseteq F_n$. Now if we can show that F_n is nonempty and closed, for all n, then by the finite intersection property, there exists a $\mu \in \bigcap_{i\in\omega} F_i$. Thus μ has the property that $\mu \geq \hat{t}\mu$, hence it is neutral for t.

Lemma 7.3.5. *For all n, the following is a Π^0_1 class:*

$$M_n = \{(R_1, R_2) \in 2^\omega \times 2^\omega \colon (\forall \sigma)\ [|\sigma| = n \Rightarrow [\hat{t}\rho(R_1)](\sigma) \leq [\rho(R_2)](\sigma)]\}.$$

Proof. Take $R_1, R_2 \in 2^\omega$ and let $\mu = \rho(R_1)$ and $\nu = \rho(R_2)$. Now $(R_1, R_2) \notin M_n$ if and only if, for some σ of length n,

$$\hat{t}\mu(\sigma) > \nu(\sigma). \tag{7.3.1}$$

Note that $\nu(\sigma)$ is computable in R_2 and $\hat{t}\mu(\sigma)$ is left c.e. in R_1. We can assume that $\nu(\sigma)$ is computed using an approximation from above (i.e., for all s, $\nu(\sigma)[s] \geq \nu(\sigma)[s+1]$) and that $\nu(\sigma)[s]$ depends only on $R_2 \upharpoonright s$. We can also assume that $\hat{t}\mu(\sigma)[s]$ depends only on $R_1 \upharpoonright s$. Then equation (7.3.1) holds if and only if for some σ of length n and stage s, we have $\hat{t}\mu(\sigma)[s] > \nu(\sigma)[s]$, and this allows us to expel $[R_1 \oplus R_2 \upharpoonright 2s]$ from M_n. □

Proposition 7.3.6. *There is a measure $\mu \in \mathcal{P}(2^\omega)$ such that μ majorizes $\hat{t}\mu$.*

Proof. As M_n is closed, $F_n = \rho(\{R \colon (R, R) \in M_n\})$ is closed. The only remaining task is to show that F_n is nonempty. Let $S_n = \{\bar{x} \in [0,1]^{2^n} \colon \sum_{i=1}^{2^n} x_i = 1\}$. Define a continuous map $\psi \colon S_n \to \mathcal{P}(2^\omega)$ by

$$\psi(\bar{x}) = \sum_{i=1}^{2^n} x_i \delta_{\sigma_i 0^\omega},$$

where σ_i is the ith string of length n. Define the multi-valued map $\phi \colon S_n \to S_n$ by

$$\phi(\bar{x}) = \{\bar{y} \in S_n \colon (\forall i \leq 2^n)\ y_i \geq [\hat{t}\psi(\bar{x})](\sigma_i)\}.$$

Note that $y_i = [\psi(\bar{y})](\sigma_i)$. So if $\bar{x}$ is a fixed point of ϕ, then $\psi(\bar{x}) \in F_n$.

Define the function $\rho_2 \colon 2^\omega \times 2^\omega \to \mathcal{P}(2^\omega) \times \mathcal{P}(2^\omega)$ by setting $\rho_2(R_1, R_2) = (\rho(R_1), \rho(R_2))$, and the function $\psi_2 \colon S_n \times S_n \to \mathcal{P}(2^\omega) \times \mathcal{P}(2^\omega)$ by $\psi_2(\bar{x}, \bar{y}) = (\psi(\bar{x}), \psi(\bar{y}))$. Both ρ_2 and ψ_2 are continuous mappings. The graph of ϕ is precisely $\psi_2^{-1}(\rho_2(M_n))$ and hence closed. Since S_n is a nonempty compact convex subset of $\mathbb{R}^{2^n}$ and for all $x \in S_n$, $\phi(x)$ is a nonempty convex subset of S_n, Kakutani's theorem tells us that ϕ has a fixed point. Therefore, F_n is nonempty. □

This proposition establishes that the Π^0_1 class of Lemma 7.3.5 is not empty. Further if t is a universal uniform test then any representation in this Π^0_1 class is a representation of a neutral measure. In order to prove Theorem 7.3.2, we need one more application of compactness.

Proof of Theorem 7.3.2. Given any $n \in \omega$, we could redefine $\hat{t}\mu$ by

$$\hat{t}\mu(\sigma) = \frac{n}{n+1} \int_{[\sigma]} t(\mu, X)\, d\mu + \frac{1}{n+1} 2^{-|\sigma|}.$$

The argument of Lemma 7.3.5 and Proposition 7.3.6 shows there exists a measure μ_n which is neutral for $\frac{n}{n+1}t$. This implies that if $n \neq 0$, then for all $X \in 2^\omega$ we have that $t(X, \mu_n) \leq \frac{n+1}{n}$. Because the space $\mathcal{P}(2^\omega)$ is compact, the sequence $\{\mu_n\}_{n\in\omega}$ has a convergent subsequence which converges to some measure μ. If μ is not neutral for t then there is some X such that $t(\mu, X) > 1$. This implies that for some open ball B including μ, and some $c > 1$ we have that $t(\nu, X) \geq c$ for all $\nu \in B$. This is a contradiction because B must include μ_n for some n with $\frac{n+1}{n} < c$. □

It is interesting that every known proof of the existence of a neutral measure uses a fixed point theorem or equivalent. Their existence seems to be a fundamentally topological fact. However, once we know such measures exist, they are relatively easy to find. There is Π^0_1 class of (representations of) neutral measures, as we can take the intersection of the diagonals of the Π^0_1 classes M_n when t is a universal test. Recall that a Turing degree is a *PA degree* if it can compute a member of every nonempty Π^0_1 subclass of 2^ω. So every PA degree computes a neutral measure. This lets us give a simple proof of the following theorem of Reimann and Slaman.

Theorem 7.3.7 (Reimann and Slaman [73, 74])**.** *For any $X \in 2^\omega$, X is not computable if and only if there exists a representation R of a measure such that X is R-random and X is not an atom of $\rho(R)$.*

Proof. Let $X \in 2^\omega$ be computable. Take any $\mu \in \mathcal{P}(2^\omega)$ such that X is not an atom of μ. If R is a representation of μ, then it is simple to build an R-test that contains X by finding initial segments of X such that $\mu(X \upharpoonright n)$ is sufficiently small.

For the other direction, assume that X is not computable. Then there is a P of PA degree such that $P \not\geq_T X$. For example, Jockusch and Soare showed that there is a set of PA degree which is also of hyperimmune-free degree and another set of PA degree which is low and hence these cannot both compute X [44]. As there is a Π^0_1 class of representations of neutral measures, P computes

a representation R of some neutral measure μ. Since μ is neutral, X is μ-random. Hence there is a representation R' of μ such that X is R'-random. Note that if A is any atom of μ then $\mu(\{A\}) > q$ for some rational q. Now the tree $\{\tau \in 2^\omega : \mu([\tau]) > q\}$ is computable in R and contains finitely many paths. Hence R can compute A. Finally as R cannot compute X and R can compute any atom of μ, we know that X is not an atom of μ. □

7.4 Locating neutral measures

In this section we study the computability-theoretic complexity of (weakly) neutral measures. In the previous section we noted that every PA degree computes a neutral measure. The reverse is true in a strong sense: if μ is a weakly neutral measure, then some PA degree is computable from every representation of μ. As we will see, the story is complicated by the fact that weakly neutral measures themselves cannot have Turing degree. We will show that their complexity can be measured using the *continuous degrees*, which were introduced by Miller. That will give us a better understanding of what is computable from (every representation of) a weakly neutral measure. We will prove that the ideal of Turing degrees below such a measure is a *Scott ideal*, and that every Scott ideal arises in this way. This, in turn, tells us about the atoms of weakly neutral measures (see Proposition 7.4.8).

One reason the existence of a weakly neutral measure may seem counter-intuitive is that such a measure does not exist for representation tests.

Lemma 7.4.1. *For all $R \in 2^\omega$, there exists an $X \in 2^\omega$ such that X is not R-random.*

Proof. Let $\mu = \rho(R)$. Construct an R-test as follows. Compute $\mu(\sigma)$ for all σ of length 2 with precision 2^{-2}. Take σ_1 to be the lexicographically least string of length 2 such that $\mu(\sigma_1)$ is within $[0, 2^{-2}]$ for this level of precision. Let $V_1 = [\sigma_1]$. Note that $\mu(\sigma_1) \leq 2^{-2} + 2^{-2} = 2^{-1}$. Once $V_i = [\sigma_i]$ has been defined with $\mu(\sigma_i) \leq 2^{-i}$, compute $\mu(\sigma_i\tau)$ for all τ of length 2 with precision 2^{-i-2}. Take the lexicographically least τ such that $\mu(\sigma_i\tau) \leq 2^{-i-2}$ with this precision. Take $\sigma_{i+1} = \sigma_i\tau$, so $\mu(\sigma_{i+1}) \leq 2^{-i-1}$. Let $V_{i+1} = [\sigma_{i+1}]$. Thus $\bigcap_{i\in\omega} V_i$ is an R-test with nonempty intersection. □

So for any representation R of a weakly neutral measure μ, there is an X that is not R-random. However, there must be another representation R' of μ such that X is R'-random. The test constructed in the previous lemma cannot be made representation independent. The obstruction is that there is no canonical representation of a weakly neutral measure, and in fact, every representation contains extraneous information. Recall that if $A \in 2^\omega$ then $\deg(A)$

is the Turing degree containing A. We say that a measure μ has a *least Turing representation* if $\{\deg(R) : R \in 2^\omega \text{ and } \rho(R) = \mu\}$ has a minimum element.

Theorem 7.4.2. *A weakly neutral measure has no least Turing degree representation.*

Proof. Let μ be a measure with least Turing degree representation R. First, by Lemma 7.4.1 there is an R-test that witnesses that some X is not R-random. Let R' be any other representation of μ. Since R' computes R, the R-test is also a R'-test (as R and R' represent the same measure). Hence X is not R'-random for any representation R' of μ, and thus X is not μ-random. Therefore, μ is not weakly neutral. □

If weakly neutral measures have no least Turing degree representation, then how should their computational power be examined? For this we turn to the continuous degrees introduced by Miller [64].

Definition 7.4.3. Let $\mathcal{M}_0$ and $\mathcal{M}_1$ be computable metric spaces and let $a \in \mathcal{M}_0$ and $b \in \mathcal{M}_1$. We define $a \leq_r b$ (a is *representation* reducible to b) if there is an index e such that for every fast Cauchy representation R of b, φ_e^R is a fast Cauchy representation of a. The *continuous degrees* are the equivalence classes under $\equiv_r$.

Miller showed that the uniformity in the above definition is not required. In other words, $a \leq_r b$ if every fast Cauchy representation of b computes a fast Cauchy representation of a, without fixing the index of the computation. This follows from the natural embedding of the continuous degrees into the enumeration degrees, see (7.4.1), and the fact that uniform and nonuniform enumeration reducibility are equivalent.

The finite sets are a countable dense subset of 2^ω under the metric d_{2^ω} of Section 7.2. Thus for any $A \subseteq \omega$, we can talk about the $\deg_r(A)$. This gives us an embedding of the Turing degrees into the continuous degrees. The continuous degrees that contain subsets of ω are called the *total degrees*. Note that if R is a representation of μ, then $R \geq_r \mu$.

Using the fact that a continuous degree a is a total degree if and only if it has a least Turing degree representation [64], we obtain the following corollary to Theorem 7.4.2.

Corollary 7.4.4. *Weakly neutral measures have non-total continuous degrees.*

This indicates that our study of (weakly) neutral measures can be enhanced by understanding the non-total continuous degrees. We start with the following definitions.

Definition 7.4.5. If a and b are Turing degrees, then a is a *PA degree relative* to b ($a \gg b$) if every nonempty $\Pi^0_1(b)$ class contains an element computable from a. For $A, B \subseteq \omega$, we write $A \gg B$ to mean that $\deg_T(A) \gg \deg_T(B)$. If a is a PA degree relative to 0, then a is simply called a *PA degree*.

Definition 7.4.6. A nonempty countable class $S \subseteq 2^\omega$ is called a *Scott set* if

(i). $A, B \in S$ implies that $A \oplus B \in S$.

(ii). $A \in S$ and $B \leq_T A$ implies $B \in S$.

(iii). For every $A \in S$, there is a $B \in S$ such that $B \gg A$.

If S is a Scott set then $\{\deg_T(A) \colon A \in S\}$ is a *Scott ideal*.

We summarise some results of Miller. The Hilbert cube $[0,1]^\omega$ can be regarded as a computable metric space by using the metric defined at 7.2.1. A suitable countable dense subset is given by the sequences of rationals which are zero at all but finitely many places.

Theorem 7.4.7 (Miller [64])**.**

(i). Every continuous degree contains an element of $[0,1]^\omega$.

(ii). Let a and b be total degrees. Then $a \ll b$ if and only if there is a non-total degree v with $a <_r v <_r b$.

(iii). The Turing ideal below a non-total continuous degree is a Scott ideal.

(iv). Any Scott ideal is the Turing ideal below some non-total continuous degree.

From Corollary 7.4.4 and Theorem 7.4.7 (iii), we know that the ideal below any weakly neutral measure is a Scott ideal. One reason this is interesting is that understanding what can be computed from a weakly neutral measure is the same as understanding its atoms.

Proposition 7.4.8. *$A \in 2^\omega$ is an atom of a weakly neutral measure μ if and only if $A \leq_r \mu$ (i.e., iff every representation of μ computes A).*

Proof. Assume that every representation of μ computes A. If μ does not concentrate on A, then any representation R of μ can compute an initial segment of A with arbitrarily small μ-measure, and hence capture A in an R-test. Therefore, A is not μ-random and μ is not weakly neutral.

For the other direction, assume that A is an atom of μ and let R be any representation of μ. Choose $\sigma \prec A$ such that $\mu(\sigma) < 2\mu(\{A\})$. Given σ, we can compute A from R by following the path consisting of all $\tau \succ \sigma$ such that $\mu(\sigma) < 2\mu(\tau)$. Therefore, $A \leq_r \mu$. □

Every Scott ideal contains a PA degree, and hence contains a member of every nonempty Π^0_1 class. There is a Π^0_1 class containing only Martin-Löf random sequences, hence:

Corollary 7.4.9. *Every weakly neutral measure has a Martin-Löf random atom.*

This result allows us to answer a question of Gács [38, Question 1] in the negative. The question was speculative and, unfortunately, a negative answer does little more than shut down this speculation. The full context of the question would take too much space, but briefly, Gács was interested in capturing the *mutual information* of two sequences $X, Y \in 2^\omega$. Let μ be a neutral measure. Gács asked if it is would be reasonable to define the mutual information of X and Y as $\log t(\mu \times \mu, (X, Y))$. More specifically, he asked if this could, for the right choice of μ, coincide with another definition he was considering. To see that this is not the case, let A be a Martin-Löf random atom of μ. Then (A, A) is an atom of $\mu \times \mu$, hence $\log t(\mu \times \mu, (A, A))$ must be finite. But a definition of mutual information that allows a Martin-Löf random sequence to have finite mutual information *with itself* is fairly absurd and, more concretely, behaves quite differently than other proposed definitions.

We have seen that the Turing ideal below a (weakly) neutral measure is always a Scott ideal. It turns out that the converse holds; the ideals below neutral measures are exactly the Scott ideals.

Theorem 7.4.10. *Every Scott ideal is the Turing ideal below some neutral measure.*

To prove this theorem, we make further use of some prior work of Miller. To prove the existence of non-total continuous degrees, Miller developed the construction of a sequence $\alpha \in [0, 1]^\omega$ that could not be diagonalised computably.

Definition 7.4.11. A sequence $\alpha \in [0, 1]^\omega$ is *diagonally non-computably diagonalisable*, or *d.n.c.d.*, if for all e, there exists a representation R of α such that $\alpha(e)$ is an element of the convex closure of $\Pi_e(R)$ or $\Pi_e(R)$ is empty (where $\Pi_e(R)$ is the eth $\Pi^0_1(R)$ subclass of $[0, 1]$).

The convex closure of $\Pi_e(R)$ is $\{x \in [0, 1] \colon \inf \Pi_e(R) \leq x \leq \sup \Pi_e(R)\}$. This definition of a d.n.c.d. sequence differs from that given in [64], but is equivalent up to continuous degree. The reason such sequences are referred to as diagonally non-computably diagonalisable is that if there is a Turing functional φ and an x such that $\varphi^R = x$ for any representation R of α, then (uniformly in the index of φ) we can find an e such that $\{x\} = \Pi_e(R)$ for all

representations of α. But then $\alpha(e) = x$. Thus α witnesses the failure of φ to diagonalise α, uniformly in (the index for) φ. The last part of Theorem 7.4.7 can be strengthened to "any Scott ideal is the Turing ideal below some d.n.c.d. sequence."

To prove Theorem 7.4.10 we will show that any d.n.c.d. sequence is above, in the sense of $\geq_r$, a neutral measure that bounds the same total degrees. We will use semimeasures in our construction of this neutral measure. A *semimeasure* is a function $\tau\colon 2^\omega \to [0,1]$ such that $\tau(\sigma) \geq \tau(\sigma 0) + \tau(\sigma 1)$. We will identify a semimeasure τ with the set $S(\tau) = \{\langle \sigma, q\rangle \in 2^{<\omega} \times \mathbb{Q}\colon \tau(\sigma) > q\}$.

Semimeasures have been studied as computably enumerable objects; we call a semimeasure τ *c.e.* if $S(\tau)$ is c.e. Levin proved the existence of a *universal* c.e. semimeasure τ, meaning that for every c.e. semimeasure τ' there is a constant c such that $\tau \geq c\tau'$ [94]. This proof relativizes to show that for any set $A \subseteq \omega$, there is a universal c.e. in A semimeasure. However, what does it mean to enumerate a semimeasure in some sequence $\alpha \in [0,1]^\omega$, if α does not have total degree? A reasonable suggestion would be to define a set to be c.e. in α if it is c.e. in every representation of α. This can be easily expressed in terms of the enumeration degrees.

There is an embedding of the continuous degrees into the enumeration degrees. Given $\alpha \in [0,1]^\omega$ we define $\Xi(\alpha) \subseteq \{0,1\} \times \omega \times \mathbb{Q}$ by

$$\Xi(\alpha) = \{\langle 0, i, q\rangle\colon q < \alpha(i)\} \cup \{\langle 1, i, q\rangle\colon q > \alpha(i)\}. \tag{7.4.1}$$

If R is a representation of α then $\Xi(\alpha)$ is c.e. in R. Further if $\Xi(\alpha)$ is c.e. in some set B, then B computes a representation of α.

Now assume that a set A is c.e. in α (i.e., c.e. in any representation of α). Further, assume that $\Xi(\alpha)$ is c.e. in a set B. Since B computes a representation of α, it must be that A is also c.e. in B. Hence $A \leq_e \Xi(\alpha)$. On the other hand, if $A \leq_e \Xi(\alpha)$, then as $\Xi(\alpha)$ is c.e. in any representation R of α, we have that A is c.e. in any representation of α. Thus the approach suggested above is equivalent to defining a semimeasure τ to be c.e. in α if $S(\tau) \leq_e \Xi(\alpha)$.

We can now provide a characterisation of μ-randomness in terms of the enumeration degrees.

Theorem 7.4.12. *Let $\mu \in \mathcal{P}(2^\omega)$. Then $X \in 2^\omega$ is μ-random if and only if for every semimeasure τ c.e. in μ (i.e., $S(\tau) \leq_e \Xi(\mu)$), there exists $c \in \omega$ such that $\tau(\sigma) \leq c\mu(\sigma)$ for all $\sigma \prec X$.*

Proof. If X is not μ-random, then for some uniform test $t(\mu, X) = \infty$. Define a (semi-)measure $\tau = \hat{t}\mu$. The (semi-)measure τ is c.e. in μ. Fix $c \in \omega$. Since

$t(\mu, X) = \infty$, there is a $\sigma \prec X$ such that if $Y \in [\sigma]$, then $t(\mu, Y) \geq c$. Thus $\tau(\sigma) = \int_{[\sigma]} t(\mu, Y)\, d\mu \geq c\mu(\sigma)$.

For the other direction, assume that there is a semimeasure τ c.e. in μ such that for all c, there exists a $\sigma \prec X$ with $\tau(\sigma) > c\mu(\sigma)$. Given any representation R of μ we can enumerate τ and define a test $\{V_i\}_{i\in\omega}$ such that $V_i = \{[\sigma] \colon \tau(\sigma) > 2^i\mu(\sigma)\}$. This test captures X so X is not R-random. As this is true for any representation of μ, X is not μ-random. □

We claim that, relative to a set B, we can enumerate all semimeasures that are B-c.e. Let $W_e(B)$ be the eth set c.e. in B and let $W_{e,i}(B)$ be an enumeration of $W_e(B)$. Any set X defines a weighting function $f_X(\sigma) = \sup\{q \colon \langle q, \sigma\rangle \in X\}$, where we are viewing X as a subset of $\mathbb{Q} \times 2^{<\omega}$. Define $T_{e,0}(B) = \emptyset$, and

$$T_{e,i+1}(B) = \begin{cases} W_{e,i}(B) & \text{if } f_{W_{e,i}(B)} \text{ is a semimeasure,} \\ T_{e,i}(B) & \text{otherwise.} \end{cases}$$

By passing from $T_e(B)$ to $S_e(B) = \{\langle q', \sigma\rangle \colon \langle q, \sigma\rangle \in T_e(B) \text{ and } q' \leq q\}$, we get an effective list of exactly the semimeasures that are B-c.e.

To prove the following lemma, we use a representation $\rho \colon 2^\omega \to [0,1]^\omega$ with the same properties as the representation of $\mathcal{P}(2^\omega)$ constructed in Section 7.2. The same proof, *mutatis mutandis*, shows that such a representation exists.

Lemma 7.4.13. *If $\alpha \in [0,1]^\omega$, then there is a universal semimeasure τ c.e. in α.*

Proof. Define $\hat{S}_e = \bigcap_{R\in\rho^{-1}(\alpha)} S_e(R)$. For any $R \in \rho^{-1}(\alpha)$, as $\rho^{-1}(\alpha)$ is $\Pi^0_1(R)$ class, it follows that $\hat{S}_e$ is c.e. in R. Let τ_e be $f_{\hat{S}_e}$. Note that this is a semimeasure. Define $\tau = \sum_{e=1}^{\infty} 2^{-e}\tau_e$. Let $S = S(\tau)$, so S is c.e. in R for all $R \in \rho^{-1}(\alpha)$.

Now if τ' is a semimeasure c.e. in α, then there is an index e such that $\tau' = S_e(R)$ for all $R \in \rho^{-1}(\alpha)$ (this holds because any reduction in the enumeration degrees is uniform, which is implicit in Selman [81] and proved independently by Rozinas [76]). Thus $\tau' = \tau_e$, and so τ majorizes τ'. □

Lemma 7.4.14. *Let $\alpha \in [0,1]^\omega$ be a d.n.c.d. sequence. If τ is a semimeasure c.e. in α, then there exists $\mu \in \mathcal{P}(2^\omega)$ such that $\mu \leq_r \alpha$ and $(\forall\sigma)\ \mu(\sigma) \geq \tau(\sigma)$.*

Proof. We will define μ in such a way that any representation of α will (uniformly) be able to determine a representation of μ. First define $\mu(\lambda) = 1$. Hence $\mu(\lambda) \geq \tau(\lambda)$.

Now assume that we have defined $\mu(\sigma)$ with $\mu(\sigma) \geq \tau(\sigma)$. Consider the interval $I_\sigma = [\tau(\sigma 0), \mu(\sigma) - \tau(\sigma 1)]$. Note that I_σ is nonempty because $\mu(\sigma) - \tau(\sigma 1) - \tau(\sigma 0) \geq \mu(\sigma) - \tau(\sigma) \geq 0$. Since $\tau(\sigma 0)$ and $\tau(\sigma 1)$ are left c.e. in R, and $\mu(\sigma)$ is computable in R, we see that I_σ is a $\Pi^0_1(R)$ class. Further, everything is

uniform, so we can actually compute an index e such that $I_\sigma = \Pi_e(R)$, for any representation R of α. Because α is of d.n.c.d. degree, and I_σ is its own convex closure, $\alpha(e) \in I_\sigma$. We define $\mu(\sigma 0) = \alpha(e)$ and $\mu(\sigma 1) = \mu(\sigma) - \mu(\sigma 0)$.

As I_σ is nonempty, we have $\mu(\sigma 0) \leq \mu(\sigma)$, and $\mu(\sigma 0) \geq \tau(\sigma 0)$. Additionally, $\mu(\sigma 1) = \mu(\sigma) - \alpha(e) \geq \mu(\sigma) - \mu(\sigma) + \tau(\sigma 1) = \tau(\sigma 1)$. □

We are finally ready to establish Theorem 7.4.10.

Proof of Theorem 7.4.10. Let $\mathcal{I}$ be a Scott ideal. Let α be a d.n.c.d. sequence such that $\mathcal{I}$ is the Turing ideal below α. Let τ be a universal semimeasure for α. By Lemma 7.4.14, we can take $\mu \leq_r \alpha$ such that μ majorizes τ and $\mu \in \mathcal{P}(2^\omega)$. Let t be a universal test. Since $\hat{t}\mu$ is a (semi-)measure c.e. in μ, hence c.e. in α, there is a b such that $\hat{t}\mu \leq b\tau \leq b\mu$. So by Lemma 7.3.4, μ is neutral for the universal test $\frac{1}{b}t$.

If $A \in \mathcal{I}$, then $A \leq_r \alpha$. Any representation of α can compute A, so some semimeasure τ c.e. in α must concentrate on A. This means A is an atom of μ and so $A \leq_r \mu$. If $A \leq_r \mu$, then $A \leq_r \alpha$ so $A \in \mathcal{I}$. Hence $\mathcal{I}$ is the Turing ideal below μ. □

The previous theorem appears to give a proof of the existence of neutral measures without using a fixed point theorem. However, this is not the case. Miller's construction of a d.n.c.d. sequence makes use of a generalisation of the Kakutani fixed point theorem and Lemma 7.4.14 makes essential use of this underlying fixed point theorem to construct the measure μ.

7.5 Open questions

In this chapter we have worked exclusively in Cantor space. How should we define randomness for non-computable probability measures on other spaces?

Question 7.5.1. Under what assumption on a computable metric space are *μ-randomness* and *μ-randomness for uniform tests* equivalent for an arbitrary probability measure μ?

There are several open questions about the relationship between neutral measures and the continuous degrees. The most basic is:

Question 7.5.2. Does every non-total continuous degree contain a neutral measure?

In the proof of Theorem 7.4.10, we started with a d.n.c.d. sequence α and built a neutral measure $\mu \leq_r \alpha$ that bounds the same Turing degrees as α.

There is no reason to assume that $\mu \equiv_r \alpha$. While μ can list all of the elements of the sequence α, it cannot necessarily determine the order of those elements. Even if we could improve the proof to show that $\mu \equiv_r \alpha$, we would run into another open question (from [64]):

Question 7.5.3. Does every non-total continuous degree contain a diagonally non-computably diagonalisable sequence?

If both questions are answered in the negative, it is natural to ask:

Question 7.5.4. Is there any relationship between the degrees of neutral measures and the degrees of d.n.c.d. sequences?

It is not too difficult to construct a weakly neutral measure that is not a neutral measure. For example, let μ be a neutral measure. Define ν such that:

(i). For all $\sigma \in 2^{<\omega}$, $\nu(0^n 1\sigma) = 2^{-n}\mu(0^n 1\sigma)$.

(ii). ν has an atom at 0^ω.

The measure μ is weakly neutral because there is an atom at 0^ω and every other sequence is in an open neighbourhood where the measure looks neutral. However, there is a uniform test t such that $t(\nu, X) = 2^n$ if $X \in [0^n 1]$ (and of course, $t(\nu, 0^\omega) = 0$). So ν is not a neutral measure.

The fact that these notions are different leads to natural questions:

Question 7.5.5. Is every weakly neutral measure representation equivalent to a neutral measure? Does every non-total continuous degree contain a weakly neutral measure?

Chapter 8

Relative Randomness and PA Degrees

This chapter is joint work with Jan Reimann of the Pennsylvania State University. It is based on research undertaken with Reimann at the Pennsylvania State University in December 2010.

8.1 Independence and relative randomness

The idea of independence is central to probability theory. Given a probability space with measure μ, we call two measurable sets $\mathcal{A}$ and $\mathcal{B}$ independent if

$$\mu\mathcal{A} = \frac{\mu(\mathcal{A} \cap \mathcal{B})}{\mu\mathcal{B}}.$$

The idea behind this definition is that if event $\mathcal{B}$ occurs, it does not make event $\mathcal{A}$ any more or less likely. This chapter considers a similar notion, that of relative randomness. We say that A is Martin-Löf random relative to B, or $A \in \mathrm{MLR}(B)$ if A is not an element of any Martin-Löf test computable in B.

The reason that relative randomness is analogous to independence is that if $A \in \mathrm{MLR}(B)$, then not only is A a random sequence but *even given* the information in B, we cannot capture A in a Martin-Löf test. If we start with the assumption that A and B are both Martin-Löf random, then the following theorem of van Lambalgen establishes that relative randomness is symmetrical.

Theorem 8.1.1 (Van Lambalgen [88])**.** *If $A, B \in \mathrm{MLR}$ then $A \in \mathrm{MLR}(B)$ if and only if $B \in \mathrm{MLR}(A)$ if and only if $A \oplus B \in \mathrm{MLR}$.*

We can extend the notion of relative randomness to any probability measure. We take $\mathcal{P}(2^\omega)$ to be the set of all Borel probability measures on Cantor space. First we need to define what it means to be relatively random in the context of non-computable probability measures. We will relativise Definitions 7.1.1 and 7.1.2.

Definition 8.1.2. Let $A \in 2^\omega$, let $\mu \in \mathcal{P}(2^\omega)$, and let $R \in 2^\omega$ be a representation of μ.

(i). An (R, A)-*test* is a uniform (in $R \oplus A$) sequence $\{V_i\}_{i\in\omega}$ of $\Sigma^0_1(R \oplus A)$ sets such that $\mu(V_i) \leq 2^{-i}$ for all $i \in \omega$.

(ii). $X \in 2^\omega$ *passes* an (R, A)-test if $X \notin \bigcap_i V_i$.

(iii). $X \in 2^\omega$ is (R, A)-*random* if it passes all (R, A)-tests.

Definition 8.1.3. Let $A \in 2^\omega$ and let $\mu \in \mathcal{P}(2^\omega)$. A sequence $X \in 2^\omega$ is (μ, A)-*random* or $X \in \mathrm{MLR}_\mu(A)$, if there exists a representation R of μ such that X is (R, A)-random.

This allows us to give a definition of relative randomness with respect to a probability measure.

Definition 8.1.4. Take $A, B \in 2^\omega$ and $\mu \in \mathcal{P}(2^\omega)$. We say that A and B are *relatively random with respect to* μ if $A \in \mathrm{MLR}_\mu(B)$ and $B \in \mathrm{MLR}_\mu(A)$.

To work with Definition 8.1.3 directly we would need to define a universal test with two oracles, the first for the measure and the second for the relativisation. We can avoid this difficulty by using representations of measures with an upwards closure property in the Turing degrees. The desired property we want is that if R is a representation of a measure μ and $A \geq_T R$ then there is some representation $\hat{R}$ of μ such that $A \equiv_T \hat{R}$. In this chapter we will say that $R = R_0 \oplus R_1$ is a representation of $\mu \in \mathcal{P}(2^\omega)$ if $\rho(R_0) = \mu$ where ρ is the function defined in Section 7.2.

This change means we can regard our representations as encoding both a measure and any potential additional oracle. Therefore the degrees of representations of a measure are closed upwards. The following proposition shows why this is useful. If we want to ask whether $X \in \mathrm{MLR}_\mu(A)$, we can do so by just considering representations of μ in the Turing cone above A.

Lemma 8.1.5. *$X \in \mathrm{MLR}_\mu(A)$ if and only if there exists some representation R of μ such that X is R-random and $R \geq_T A$.*

Proof. Assume that X is R-random and $R \geq_T A$. In this case any (R, A)-test is just an R-test and so X is (R, A)-random and thus $X \in \mathrm{MLR}_\mu(A)$. Conversely assume that $X \in \mathrm{MLR}_\mu(A)$. Then there is some representation R of μ such that X is (R, A)-random. Now there is some representation $\hat{R}$ of μ such that $\hat{R} \equiv_T R \oplus A$ and any $\hat{R}$-test is also a (R, A)-test. Thus X is $\hat{R}$-random and $\hat{R} \geq_T A$. □

If A and B are relatively random with respect to some measure μ, then μ might offer some information about the relationship between A and B. For example, we know that if A and B are relatively random with respect to the uniform measure, then any sequence they both compute must be K-trivial (this follows from results of Nies, and Hirschfeldt, Nies and Stephan which establish that lowness for Martin-Löf randomness, K-triviality and being a base for Martin-Löf randomness coincide [39, 69]). If A and B are both atoms of μ then clearly A and B are relatively random with respect to μ. Given this, perhaps the most obvious question to ask about relative randomness is the following.

Question 8.1.6. For which $A, B \in 2^\omega$ does there exist a measure μ such that A and B are relatively random with respect to μ and neither A nor B is an atom of μ?

This question is closely related to Theorem 7.3.7 of Reimann and Slaman. This theorem states that an element X of Cantor space is non-computable if and only if there exists a measure μ such that X is μ-random and X is not an atom of μ.

Van Lambalgen's theorem states that A and B are relatively random if and only if $A \oplus B \in \mathrm{MLR}$. If we take λ to be the uniform measure, then $A \oplus B \in \mathrm{MLR}$ if and only if the pair $(A, B) \in 2^\omega \times 2^\omega$ is Martin-Löf random with respect to the product measure $\lambda \times \lambda$ or $(A, B) \in \mathrm{MLR}_{\lambda\times\lambda}$. We begin our investigation into relative randomness by showing that van Lambalgen's theorem holds for any Borel probability measure on Cantor space.

In the discussion following Definition 7.1.1 we noted that there exists a uniform sequence of c.e. sets U_n such that if $U_n^R = \{[\tau] \colon \langle \tau, \sigma \rangle \in U_n \text{ and } \sigma \prec R\}$, then $\{U_n^R\}_{n\in\omega}$ is a universal R-test. We called $\{U_n\}_{n\in\omega}$ a universal oracle Martin-Löf test. Now we can define a *uniform oracle Martin-Löf test* by

$$\{U_n^R\}_{n\in\omega} = \bigcap_{\hat{R}\in\rho^{-1}(R)} V_n^{\hat{R}}$$

where $\{V_n\}_{n\in\omega}$ is a universal oracle Martin-Löf test. The sequence $\{U_n^R\}_{n\in\omega}$ is uniformly c.e. in R because $\pi^{-1}(R)$ is a $\Pi_1^0(R)$ class and so a compact subset of 2^ω. We term this a uniform oracle Martin-Löf test, as it is similar to the uniform test constructed in the proof of Theorem 7.1.6. In particular if R is a representation of a measure μ, then $\mathrm{MLR}_\mu = 2^\omega \setminus (\bigcap_n U_n^R)$.

Note that uniform oracle Martin-Löf tests are not special cases of universal oracle Martin-Löf tests, for example if R is a representation of a neutral measure then $\bigcap_n U_n^R$ is empty if $\{U_n\}_{n\in\omega}$ is a uniform test, but not empty if $\{U_n\}_{n\in\omega}$

is a universal test (by Lemma 7.4.1). We will use both universal and uniform oracle Martin-Löf tests to generalise Theorem 8.1.1.

Theorem 8.1.7. *Let $\mu \in \mathcal{P}(2^\omega)$ and let $A, B \in 2^\omega$. Then $(A, B) \in \mathrm{MLR}_{\mu\times\mu}$ if and only if $A \in \mathrm{MLR}_\mu$ and $B \in \mathrm{MLR}_\mu(A)$.*

Proof. Let $\{U_n^X\}_{n\in\omega}$ be a universal oracle Martin-Löf test on 2^ω. Assume $B \notin \mathrm{MLR}_\mu(A)$. Let R be any representation of μ. Let $\hat{R}$ be a representation of μ such that $\hat{R} \equiv_T A \oplus R$ so $\hat{R} = \Phi(A \oplus R)$ for some Turing functional Φ. Now because $B \notin \mathrm{MLR}_\mu(A)$ we have that $B \in \bigcap_n U_n^{\hat{R}}$ by Lemma 8.1.5. We define an R-test for $2^\omega \times 2^\omega$ by letting $V_n^R = \{\{X\} \times [\sigma] : \sigma \in U_n^{\Phi(X\oplus R)}\}$. This ensures that $(A, B) \in \bigcap_n V_n^R$. By applying Fubini's theorem we can establish that:

$$\begin{aligned}(\mu \times \mu)V_n^R &= \int\int_{2^\omega\times 2^\omega} \chi_{V_n^R}(X, Y)d\mu \times d\mu \\ &= \int_{2^\omega}\int_{2^\omega} \chi_{U_n^{\Phi(X\oplus R)}}(Y)d\mu(Y)d\mu(X) \\ &\leq \int_{2^\omega} 2^{-n}d\mu(X) = 2^{-n}.\end{aligned}$$

Hence (A, B) is not R-random. As this is true for any representation R of μ we have that $(A, B) \notin \mathrm{MLR}_{\mu\times\mu}$. The same argument shows *a fortiori* that if $A \notin \mathrm{MLR}_\mu$ then $(A, B) \notin \mathrm{MLR}_{\mu\times\mu}$.

To establish the other direction assume that $(A, B) \notin \mathrm{MLR}_{\mu\times\mu}$. This time let $\{U_n^X\}_{n\in\omega}$ be a *uniform* oracle Martin-Löf test on $2^\omega \times 2^\omega$. Let $\{V_n^X\}_{n\in\omega}$ be a *universal* oracle Martin-Löf test on $2^\omega \times 2^\omega$.

Let R be any representation of μ such that $R \geq_T A$. We can define $T_n^R = \{Y \in 2^\omega : (A, Y) \in U_{2n}^R\}$. If for almost all n, $\mu T_n^R \leq 2^{-n}$, then we can turn $\{T_n^R\}_{n\in\omega}$ into an R-test and so B is not R-random because $B \in \bigcap_n T_n^R$.

Now assume that $B \in \mathrm{MLR}_\mu(A)$. Then for some fixed representation $\hat{R}$ of μ, with $\hat{R} \geq_T A$, and for infinitely many n, $\mu T_n^{\hat{R}} > 2^{-n}$. Now because $\{U_n^X\}_{n\in\omega}$ is a uniform oracle Martin-Löf test, then if R is any representation of μ, we have that $U_n^{\hat{R}} \subseteq V_n^R$. Now we will define a test by $S_n^R = \{X \in 2^\omega : \mu\{Y \in 2^\omega : (X, Y) \in V_{2n}^R\} \geq 2^{-n}\}$. First, $\mu S_n^R \leq 2^{-n}$ because

$$\begin{aligned}2^{-2n} \geq (\mu \times \mu)V_{2n}^R &\geq \int_{S_n(R)}\int_{2^\omega} \chi_{V_{2n}^R}(X, Y)d\mu(Y)d\mu(X) \\ &\geq \int_{S_n(R)} 2^{-n}d\mu(X) = 2^{-n}\mu(S_n(R)).\end{aligned}$$

Secondly we know that for infinitely many n, $\mu T_n^{\hat{R}} > 2^{-n}$. Hence for these n, we have that $A \in S_n^R$. This implies that A is not R-random because if we define $\hat{S}_n^R = \bigcup_{i>n} S_i^R$, then $\{\hat{S}_n^R\}_{n\in\omega}$ is an R-test that captures A. As this is true for any representation of μ, we have that $A \notin \mathrm{MLR}_\mu$. □

Corollary 8.1.8. *If $A, B \in 2^\omega$ and $\mu \in \mathcal{P}(2^\omega)$, then A and B are relatively random with respect to μ if and only if $(A, B) \in \mathrm{MLR}_{\mu\times\mu}$.*

This theorem extends the results in van Lambalgen's thesis by adapting some of the techniques in Nies's proof of van Lambalgen's theorem [70, 88].[1]

Corollary 8.1.9. *If $A \geq_T B$ and $(A, B) \in \mathrm{MLR}_{\mu\times\mu}$ then B must be an atom of μ.*

Proof. If $A \geq_T B$, then $B \in \mathrm{MLR}_\mu(A)$ implies $B \in \mathrm{MLR}_\mu(B)$ by Lemma 8.1.5 which in turn implies that B is an atom of μ. □

We note that we cannot extend one direction of van Lambalgen's theorem to product measures of the form $\mu \times \nu$. In particular it is not true that if $A \in \mathrm{MLR}_\mu$ and $B \in \mathrm{MLR}_\nu(A)$ then $(A, B) \in \mathrm{MLR}_{\mu\times\nu}$. For example, let λ be the uniform measure and let $A, B \in 2^\omega$ be relatively random with respect to λ. We define $\nu = (\lambda + \delta_B)/2$ where δ_B is the measure that concentrates on B. We have $B \in \mathrm{MLR}_\lambda(A)$ by assumption. Now we will establish that $A \in \mathrm{MLR}_\nu$. We know that there is a representation R of ν such that $R \equiv_T B$. Let $\{U_n^R\}_{n\in\omega}$ be the universal R-test. Now $\lambda(U_{n+1}) \leq 2\nu(U_n + 1) \leq 2^{-n}$ hence $\{U_{n+1}^R\}_{n\in\omega}$ is a Martin-Löf test relative to B. Thus $A \notin \bigcap_n U_n^R$. However any representation of $\nu \times \lambda$ must be able to compute B because B is an atom of ν. This implies that $(A, B) \notin \mathrm{MLR}_{\nu\times\lambda}$ because the test $\{2^\omega \times [B \upharpoonright n]\}_{n\in\omega}$ is an R-test for any representation R of $\nu \times \lambda$.

Given any $A \in 2^\omega$, we will use $\mathcal{R}(A)$ to denote the set of sequences B such that A and B are relatively random with respect to some measure μ and neither A nor B are atoms of μ i.e.

$$\mathcal{R}(A) = \{B \in 2^\omega : (\exists \mu \in \mathcal{P}(2^\omega))((A, B) \in \mathrm{MLR}_{\mu\times\mu} \wedge \mu\{A, B\} = 0)\}.$$

The following lemma will be useful when examining the basic properties of the set $\mathcal{R}(A)$.

Lemma 8.1.10. *If λ is the uniform measure, $\mu \in \mathcal{P}(2^\omega)$, and $\nu = \frac{1}{2}(\lambda + \mu)$ then $\mathrm{MLR}_\mu \subseteq \mathrm{MLR}_\nu$.*

Proof. Take any $X \notin \mathrm{MLR}_\nu$ and let R be a representation of μ. There exists some representation S of ν such that $S \leq_T R$. Because $X \notin \mathrm{MLR}_\nu$, there is an S-test $\{U_n\}_{n\in\omega}$ such that $X \in \bigcap_n U_n$. Now $\{U_{n+1}\}_{n\in\omega}$ is an R-test because $\mu(U_{n+1}) \leq 2 \cdot \nu(U_{n+1}) \leq 2^{-n}$. Therefore $X \notin \mathrm{MLR}_\mu$. □

[1] Essentially some of the tests constructed by van Lambalgen were tests for weak 2-randomness.

Proposition 8.1.11. *For all $A, B \in 2^\omega$ the following hold:*

(i). $A \in \mathcal{R}(B)$ if and only if $B \in \mathcal{R}(A)$.

(ii). $B \in \mathcal{R}(A)$ implies that $A \mid_T B$.

(iii). If A is non-computable, then $\mathcal{R}(A)$ has uniform measure 1.

(iv). If $A \in \mathrm{MLR}$ then $\mathrm{MLR}(A) \subsetneq \mathcal{R}(A)$.

Proof. (i) This holds because for all $\mu \in \mathcal{P}(2^\omega)$, $(A, B) \in \mathrm{MLR}_{\mu\times\mu}$ if and only if $(B, A) \in \mathrm{MLR}_{\mu\times\mu}$.
(ii) This holds by Corollary 8.1.9.
(iii) If A is non-computable then there is a measure μ such that A is not an atom of μ and $A \in \mathrm{MLR}_\mu$. Let $\nu = (\mu + \lambda)/2$ where λ is uniform measure. We have that $A \in \mathrm{MLR}_\nu$ (by the previous lemma) and $\nu\{A\} = 0$. Hence $(\mathrm{MLR}_\nu(A) \setminus \{B : \nu\{B\} \neq 0\}) \subseteq \mathcal{R}(A)$. Now $\lambda(\mathrm{MLR}_\nu(A)) = 1$ because the complement of $\mathrm{MLR}_\nu(A)$ is a ν null set and hence a λ null set. The set of atoms of ν is countable and so has uniform measure 0. This gives us that $\lambda(\mathrm{MLR}_\nu(A) \setminus \{B : \nu\{B\} \neq 0\}) = 1$.
(iv) Let $A = A_0 \oplus A_1 \in \mathrm{MLR}$. By the definition of $\mathcal{R}(A)$ we have that $\mathrm{MLR}(A) \subseteq \mathcal{R}(A)$. To show that the reverse inclusion does not hold, take $B \in \mathrm{MLR}(A)$. Let $\mu = \lambda \times \delta_{A_1}$ where λ is the uniform measure and δ_{A_1} is the measure that concentrates on A_1. Let us regard μ as a measure on 2^ω rather than on $2^\omega \times 2^\omega$. Clearly $B \oplus A_1 \notin \mathrm{MLR}(A)$. If $B \oplus A_1 \notin \mathrm{MLR}_\mu(A)$ then fix representation R of μ such that $\deg(R) = \deg(A_1)$. There is an R-test $\{U_n\}_{n\in\omega}$ that captures $B \oplus A_1$. Given access to A we define a new test $\{V_n\}_{n\in\omega}$ by $V_n = \{X \in 2^\omega : X \oplus A_1 \in U_{n+1}\}$. Now $\lambda(V_n) \cdot \delta_{A_1}(\{A_1\}) \leq \mu(U_n)$ which implies that $B \notin \mathrm{MLR}(A)$. This contradicts our choice of B and hence $B \oplus A_1 \in \mathrm{MLR}_\mu(A)$. This shows that $B \oplus A_1 \in \mathcal{R}(A)$. □

The above properties leave open the possibility that $\mathcal{R}(A)$ is just the class of sets that are Turing incomparable with A. We will now establish that this is not necessarily the case.

Proposition 8.1.12. *Let R be a representation of a measure μ, and let A be a c.e. set. Then $R \oplus A \geq_T R'$ if*

(i). A is R-random.

(ii). A is not an atom of μ.

Proof. Given such an R and an A, let A_s be a computable enumeration of A. We define the function $f \leq_T A \oplus R$ by:

$$f(x) = \min\{s : (\exists m \leq s)(A_s \upharpoonright m = A \upharpoonright m \wedge \mu_s[A \upharpoonright m] < 2^{-x})\}.$$

In this definition we take $\mu_s[\sigma]$ to be an R-computable approximation to $\mu[\sigma]$ from above. Note that f is well defined because A is not an atom of μ. We claim that if g is any partial function computable in R, then for all but finitely many $x \in \text{dom}(g)$, we have that $f(x) > g(x)$. To establish this claim, let g be an R-computable partial function. We will build an R-test $\{U_n\}_{n\in\omega}$ by defining U_n to be:

$$\{X \in 2^\omega : (\exists x > n, m)(g(x)\downarrow \wedge \mu[A_{g(x)} \upharpoonright m] < 2^{-x} \wedge X \succ (A_{g(x)} \upharpoonright m))\}.$$

Because any $x \in \text{dom}(g)$ adds a single open set ($[A_{g(x)} \upharpoonright m]$ for some m) of measure less than 2^{-x} to those U_n with $n < x$, we have constructed a valid test. Now if $g(x)\downarrow \geq f(x)$, then by definition of f, there is some $m \leq f(x)$ such that $\mu[A \upharpoonright m] < 2^{-x}$ and $A \upharpoonright m = A_{f(x)} \upharpoonright m = A_{g(x)} \upharpoonright m$. Thus for all $n < x$, $A \in U_n$. Because A is R-random we have that $f(x) > g(x)$ for all but finitely many x in $\text{dom}(g)$.

Let $g(x)$ be the R-computable partial function with domain R' such that $g(x)$ is the unique number s such that $x \in R'_{s+1} \setminus R'_s$. For almost all x, we have that $x \in R'$ if and only if $x \in R'_{f(x)}$ and so $R' \leq_T A \oplus R$.

□

Theorem 8.1.13. *Let R be a representation of a measure μ, and let A be a c.e. set. Then $R \oplus A \geq_T \emptyset'$ if:*

(i). A is μ-random.

(ii). A is not an atom of μ.

Proof. Note the following characteristics of the previous proof. First the totality of f does not depend on the fact that A is R-random, it only depends on the fact that A is not an atom of μ. The construction is uniform so there is a single index e such that $\Phi_e(A \oplus \hat{R})$ is total if $\hat{R}$ is any representation of μ. Additionally if A is $\hat{R}$-random then for all but finitely many x, $\Phi_e(A \oplus \hat{R}; x) > g(x)$ where g is any fixed $\hat{R}$-computable partial computable function.

Let R be any representation of μ. The set $\{A \oplus \hat{R} : \hat{R} \text{ is a representation of } \mu\}$ is a $\Pi^0_1(A \oplus R)$ class (Lemma 7.2.2) and Φ_e is total on this class. From $A \oplus R$ we can compute a function f that dominates $\Phi_e(A \oplus \hat{R})$ where A is $\hat{R}$-random. As f dominates any $\hat{R}$-computable partial function we have that $A \oplus R \geq_T \emptyset'$. □

Corollary 8.1.14. *If A is c.e. and $B \leq_T \emptyset'$ then $B \notin \mathcal{R}(A)$.*

Proof. Let $\mu \in \mathcal{P}(2^\omega)$ be such that neither A nor B is an atom of μ. If A is a μ-random c.e. set and R is a representation of μ, then $A \oplus R \geq_T B$. Hence $B \notin \mathrm{MLR}_\mu(A)$ and so $(A, B) \notin \mathrm{MLR}_{\mu\times\mu}$. □

8.2 Computably enumerable sets and PA degrees

We will now show two interesting applications of Theorem 8.1.13 to the study of the interaction between computably enumerable sets and sets of PA degree. First we review some basic facts. A function $f : \omega \to \omega$ is a called a *diagonally non-computable function* or *DNC function* if for all e such that $\varphi_e(e)\downarrow$, $f(e) \neq \varphi_e(e)$. Any set of PA degree computes a $\{0, 1\}$ valued DNC function [44]. Finally Arslanov's completeness criterion states that any c.e. set that computes a DNC function is complete [3].

Corollary 8.2.1 (to Theorem 8.1.13)**.** *If A is a c.e. set and P a set of PA degree such that $P \not\geq_T A$ then $P \oplus A \geq_T \emptyset'$.*

Proof. In Chapter 7 we noted that any set of PA degree computes a neutral measure. Hence P can compute a neutral measure μ and $A \in \mathrm{MLR}_\mu$. Now because $P \not\geq_T A$ we have that A is not an atom of μ. Thus $P \oplus A \geq_T \emptyset'$. □

In the slides of a talk to the 2004 Córdoba conference in Logic, Computability and Randomness, Kučera asked the following question.

Question 8.2.2. For every incomplete c.e. set A, does there exist a set P of PA degree such that $A <_T P <_T \emptyset'$?

Kučera gave a negative answer to this question based on an unpublished result of Kučera and Slaman.[2] We provide a complete answer to this question by showing that the c.e. sets with this property are precisely the low c.e. sets.

Theorem 8.2.3. *If A is a c.e. set then the following are equivalent:*

(i). A is low.

(ii). There exists P such that, $P \gg A$ and P is low.

(iii). There exists P of PA degree such that $\emptyset' >_T P >_T A$.

[2]Their result is that there is a c.e. set A with $A'' \equiv_T \emptyset''$ such that $A \oplus f \equiv_T \emptyset'$ for any diagonally non-computable function $f \leq_T \emptyset'$.

Proof. (i) implies (ii). Relativize the low basis theorem to find $P \gg A$ and $P' \equiv_T A'$. As A is low so is P.
(ii) implies (iii). This is a trivial implication.
(iii) implies (i). Take any Q of PA degree such that $P \gg Q$ (see Simpson [82]). Now $Q \geq_T A$ because otherwise $Q \oplus A \geq_T \emptyset'$ by Corollary 8.2.1, but this is impossible because $P \geq_T Q \oplus A$ and $P \ngeq_T \emptyset'$. Hence $P \gg A$. But now we have that $\emptyset'$ is c.e. in A and also $\emptyset'$ computes a DNC function relative to A. Hence by relativizing Arslanov's completeness criterion we have that $A' \equiv_T \emptyset'$. □

Using the same ideas we can also strengthen a theorem of Kučera. Kučera pointed out the following behaviour of sets of PA degree.

Theorem 8.2.4 (Kučera [52])**.** *There exists $A, B \in 2^\omega$ with A of PA degree and $A >_T B >_T 0$ such that for all $C \in 2^\omega$ such that C is of PA degree and $A \geq_T C$ we have that $C \geq_T B$.*

In fact we can take A and B to be any such sets with the additional properties that A is incomplete and B is c.e. Let C be of PA degree such that $A \geq_T C$. Now if $C \ngeq_T B$, then by Corollary 8.2.1 we must have $C \oplus B \geq_T \emptyset'$. However $A \geq_T C \oplus B$ and A is incomplete. Therefore it must be that $C \geq_T B$.

Chapter 9

Computable Lipschitz Reducibility

(The results of this chapter appeared in the paper: The computable Lipschitz degrees of computably enumerable sets are not dense, Annals of Pure and Applied Logic, Vol 161 (2010), no. 12, pp. 1588 – 1602.)

9.1 Overview

If $A, B \in 2^\omega$, what does it mean for A to be more random than B? We have seen in Chapter 2 that the randomness of a sequence is linked to the descriptive complexity of its initial segments. This suggests that A will be more random than B if the descriptive complexity of any initial segment of A is greater than the descriptive complexity of the corresponding initial segment of B. This can be formalised using plain Kolmogorov complexity by saying A is more random than B if:

$$C(A \upharpoonright n) \geq C(B \upharpoonright n) - O(1).$$

Elements of 2^ω can also be ordered by computational power, typically using Turing reducibility. What is the relationship between relative randomness and relative computational power? It is certainly easy to construct a set A and a set B such that A is more random than B but B has greater computational power than A. For example, A could be Chaitin's Ω and the nth bit of B could be 0 unless for some m, $n = 2^m$ and $A'(m) = 1$. In this chapter we will examine computable Lipschitz or cl reducibility. This reducibility measures both relative randomness and computational power. This chapter establishes that the ordering this reducibility induces on the computably enumerable sets is not dense.

Definition 9.1.1. (Downey, Hirschfeldt and LaForte [31]) Let $A, B \subseteq \omega$. We say that B is *computable Lipschitz* reducible to A, written $B \leq_{cl} A$, if there exists a

Turing functional Γ and a constant c such that for all x, $\Gamma^A(x) = B(x)$ and the use of this computation is bounded by $x + c$.

In essence, the Turing functionals used in Definition 9.1.1 are effective versions of Lipschitz continuous operators (for details see [59]).

If we require the constant c in Definition 9.1.1 to be 0, then we get an even stronger reducibility known as *identity bounded Turing* or ibT reducibility. Both ibT reducibility and cl reducibility maintain some sense of relative randomness. If $A \geq_{cl} B$, then $C(A \upharpoonright n) \geq C(B \upharpoonright n) - O(1)$ and $K(A \upharpoonright n) \geq K(B \upharpoonright n) - O(1)$. There are applications of these reducibilities beyond randomness. For example, the ibT reducibility has been used in differential geometry [21].

As is the case with Turing reducibility, we can define a degree structure on the subsets of ω using either cl or ibT reducibility. We say $A \equiv_{cl} B$ if $A \leq_{cl} B$ and $B \leq_{cl} A$. We can then define the cl degree of A to be $\deg(A) = \{B : A \equiv_{cl} B\}$. The class of all cl degrees is a partially ordered set under $\leq_{cl}$ where $\deg(A) \leq_{cl} \deg(B)$ is defined to hold if $A \leq_{cl} B$. If we only consider those degrees that have a computably enumerable member, then we can talk about the cl degrees of c.e. sets. Similarly we can define the ibT degrees.

Given such a structure, a fundamental question is whether or not it is dense. The Turing degrees of c.e. sets, and the wtt degrees of c.e. sets are both dense. The first of these results is due to Sacks and the second to Ladner and Sasso [54, 77]. Barmpalias and Lewis showed that the ibT degrees of c.e. sets are not dense [7]. For the cl degrees of c.e. sets, the main related results in this area are the following.

Theorem 9.1.2. *(Downey, Hirschfeldt and LaForte [31]) There is no cl-minimal c.e. set. That is for every c.e. set A, there exists a c.e. set W such that $A >_{cl} W >_{cl} \emptyset$.*

Theorem 9.1.3. *(Barmpalias [4]) There are no cl-maximal c.e. sets. That is for every c.e. set A, there exists a c.e. set W such that $A <_{cl} W$.*

Hence the cl degrees are downwards dense, but as there is no maximal element, it does not make sense to talk about upwards density. This chapter establishes the following theorem.

Theorem 9.1.4. *The cl degrees of c.e. sets are not dense.*

The proof of this theorem uses a construction that is loosely based on Barmpalias and Lewis's proof that the ibT degrees of c.e. sets are not dense [7]. However, the construction used for the cl degrees is more difficult. New techniques are developed in this chapter that may well find wider applicability.

This theorem has an application to the Solovay degrees.

Definition 9.1.5. (Solovay [30]) If $x, y \in \mathbb{R}$, then y is *Solovay reducible* to x, written $y \leq_S x$, if there are a constant c and a partial computable function $f : \mathbb{Q} \to \mathbb{Q}$ such that if $q \in \mathbb{Q}$, and $q < x$, then $f(q) < y$ and $y - f(q) < c(x - q)$ (where $\mathbb{Q}$ is the set of all rationals).

We call $x \in \mathbb{R}$ a *strongly c.e.* real if $x = 0.A(0)A(1)A(2)\ldots$ for some c.e. set A. Hence Theorem 9.1.4 says that the cl degrees of strongly c.e. reals are not dense. Solovay reducibility agrees on the strongly c.e. reals with cl reducibility on the c.e. sets that define those reals, i.e. if $x = 0.A(0)A(1)A(2)\ldots$ and $y = 0.B(0)B(1)B(2)\ldots$ for c.e. sets A and B, then $y \leq_S x$ if and only if $B \leq_{cl} A$ [31]. Hence we get the following corollary.

Corollary 9.1.6. *The Solovay degrees of strongly c.e. reals are not dense.*

The c.e. reals are an important object of study in algorithmic randomness. A real x is c.e. if $\{q \in \mathbb{Q} : q < x\}$ is computably enumerable. Recently, there has been significant interest in the cl degrees of c.e. reals [6, 31, 93]. The question whether the cl degrees of c.e. reals are dense remains open. The techniques developed in this chapter to prove Theorem 9.1.4 may be useful in answering this question.

In this chapter we will use the notation $A \upharpoonright n$ for $A \upharpoonright (n+1)$. Given a set $A \subseteq \omega$, and integers a, b we define: $A[a, b] = \{x \in A : a \leq x \leq b\}$ and $A(a, b) = \{x \in A : a < x < b\}$.

9.2 Proof strategy

Our goal is to prove that the cl degrees of c.e. sets are not dense. To do this, we will construct c.e. sets A and B such that $B <_{cl} A$, and for any c.e. set W such that $B \leq_{cl} W \leq_{cl} A$ we have that $A \equiv_{cl} W$ or $B \equiv_{cl} W$. Another way of describing this situation is to say that A is a minimal cover of B.

We will achieve this goal by developing a construction of A and B that meets a number of requirements. First we need to ensure that $B \leq_{cl} A$. To keep $B \leq_{cl} A$, we will code any change to B into A by changing A before the change in B. This ensures that if $A_s \upharpoonright x = A \upharpoonright x$, then $B_s \upharpoonright x = B \upharpoonright x$. Hence $B(x)$ is computable from A with use x.

Secondly, we need to ensure that $A \not\leq_{cl} B$. This can be achieved by diagonalising against all cl functionals. We can take an enumeration of all Turing functionals $\{\Gamma_p\}_{p<\omega}$. We can turn this into an enumeration of cl functionals

$\{\Delta_p\}_{p<\omega}$, by defining, for any oracle Z:

$$\Delta_p^Z(x) = \begin{cases} \Gamma_p^Z(x) & \text{if } \gamma_p^Z(x) \leq x + p \\ \uparrow & \text{otherwise,} \end{cases}$$

where $\gamma_p^Z(x)$ is the use function of Γ_p with oracle Z. This works because any Turing functional has an infinite number of indices. Assume that we have some Turing functional Ψ, and an oracle Z, such that for some constant k, $\psi^Z(x) \leq x+k$. This means that Ψ with oracle Z is a cl functional. Now there is some index $p \geq k$ such that $\Gamma_p = \Psi$ and Γ_p^Z has use function $\gamma_p^Z(x) = \psi^Z(x) \leq x + k \leq x + p$, thus $\Delta_p^Z = \Gamma_p^Z = \Psi^Z$. So we can ensure that $A \not\leq_{cl} B$ by meeting for all $p \in \omega$, the requirement:

P_p: $\quad \Delta_p^B \neq A$.

Given any p, if at some stage s, for some k, we find that $\Delta_p^B[s] \upharpoonright k = A_s \upharpoonright k$, then we want to find some $x < k$ with $x \notin A_s$, add x to A_{s+1} and preserve B on $x + p \geq \delta_p^{B_s}(x)$. This is the Friedberg-Muchnik strategy for dealing with such requirements. If we do this for all $p \in \omega$, then $A \not\leq_{cl} B$. For these requirements, we do not need to add anything to B. However, it will be advantageous to do so. We will change B but only above $\delta_p^{B_s}(x)$. In fact our construction must make some change to B as we know by Theorem 9.1.2 that B cannot be computable. Thus we will depart slightly from the classic Friedberg-Muchnik approach; if we add something to A, we will *always* add something to B as well.

To understand why it is useful to add something to B, we need to consider our other requirements to make A a minimal cover of B. What we will do is enumerate all triples $r = \langle a, b, c\rangle$. Now for all such r we define W_r to be the ath c.e. set (we could define r as a function of a but this is cumbersome). Also we define:

$$\Phi_r^Z(x) = \begin{cases} \Gamma_b^Z(x) & \text{if } \gamma_b^Z(x) \leq x + r \\ \uparrow & \text{otherwise;} \end{cases}$$

$$\Psi_r^Z(x) = \begin{cases} \Gamma_c^Z(x) & \text{if } \gamma_c^Z(x) \leq x + r \\ \uparrow & \text{otherwise.} \end{cases}$$

If $B \leq_{cl} W \leq_{cl} A$ then there will be some c.e. functionals Γ_b, Γ_c and constant d such that $\Gamma_b^A = W$, and $\Gamma_c^W = B$ with $\gamma_b^A(x)$, $\gamma_c^W(x) \leq x + d$. Now there is some $r \geq d$ with $r = \langle a, b, c\rangle$ such that $W_r = W$. So we have that $\Phi_r^A = \Gamma_b^A = W_r$ and $\Psi_r^{W_r} = \Gamma_c^{W_r} = B$.

It is simpler to write $r = \langle W, \Phi, \Psi\rangle$ with the understanding that $W = W_r$, $\Phi = \Phi_r$ and $\Psi = \Psi_r$. Hence we see that if $B \leq_{cl} A$ but A is not a minimal cover of B then for some triple $r = \langle W, \Phi, \Psi\rangle$ we have that:

(i). $\Phi^A = W$.

(ii). $\Psi^W = B$.

(iii). $A \not\leq_{cl} W$.

(iv). $W \not\leq_{cl} B$.

What we will do is ensure that for all such triples $\langle W, \Phi, \Psi \rangle$, if properties (i) and (ii) above hold, then either property (iii) or (iv) does not hold. Suppose that we want to ensure that (iii) does not hold. To show this, we need to build a Turing functional Γ such that $\Gamma^W = A$ with $\gamma^W(x) \leq x + c$ for some constant c. Hence we would like to somehow get W to permit every A change. The problem is W is not under our control. We can regard W as being controlled by an opponent who would like to see us fail. However, if (ii) holds then we can force a W-change by adding an element to B at a stage $s + 1$ within the length of agreement of $\Psi^W[s]$ and B_s. This is why we always add something to B. We add to A to meet some requirement P_p. At the same time we add something to B to force a change in W (the change in B is above the use of the change in A). We will use this W-change to ensure that either (iii) or (iv) is false i.e. that $W \equiv_{cl} A$ or $W \equiv_{cl} B$. This gives us another set of requirements R_r for all $r \in \omega$.

R_r: If $W_r = \Phi_r^A$ and $B = \Psi_r^{W_r}$, then there exists a Γ such that: $W_r = \Gamma^B$ with $\gamma^B(x) \leq x + r$; or $A = \Gamma^{W_r}$ with $\gamma^{W_r}(x) \leq x + r$.

In our proof we will need to monitor various length of agreements to determine when we need to act on a particular requirement.

Definition 9.2.1. Given two sequences A, B we define:

$$d_s(A, B) = \min(\{n < s : A(n) \neq B(n)\} \cup \{s\}).$$

If $r = \langle W, \Phi, \Psi \rangle$ and $s \in \omega$, we define the length of agreement for requirement R_r by:

$$l_r(s) = \min(d_s(B_s, \Psi^W[s]), d_s(W_s, \Phi^A[s])).$$

If $p \in \omega$ and $s \in \omega$, we define the length of agreement for requirement P_p by:

$$m_p(s) = d_s(A_s, \Delta_p^B[s]).$$

Note that l_r and m_p are computable. We will say that l_r is unbounded if $\{l_r(s) : s \in \omega\}$ is infinite. Because we know the bound on the use, we can make use of the following lemma.

Lemma 9.2.2. *Let $r = \langle W, \Phi, \Psi \rangle$. The following are equivalent:*

(i). l_r is unbounded.

(ii). $\Phi^A = W$ and $\Psi^W = B$.

(iii). $\liminf l_r(s) = \infty$.

Proof. (i) implies (ii). If l_r is unbounded then given any x there is a stage s such that $l_r(s) > x$, and $x + r < \min\{d_s(A_s, A), d_s(B_s, B), d_s(W_s, W)\}$. At this stage we have that $\Phi^A(x) = \Phi^A(x)[s] = W_s(x) = W(x)$ and $\Psi^W(x) = \Psi^W(x)[s] = B_s(x) = B(x)$.
(ii) implies (iii). Take any $x \in \omega$. Let s_0 be a stage such that:

$$x + r < \min\{d_{s_0}(A_{s_0}, A), d_{s_0}(B_{s_0}, B), d_{s_0}(W_{s_0}, W)\}.$$

Let $s_1 > s_0$ be a stage such that for all $y \leq x$, we have that $\Phi^A(y)[s_1] \downarrow = \Phi^A(y)$ and $\Psi^W(y)[s_1] \downarrow = \Psi^W(y)$. So for all $s > s_1$, for all $y \leq x$, $\Phi^A(y)[s] = \Phi^A(y) = W(y) = W_s(y)$ and $\Psi^W(y)[s] = \Psi^W(y) = B(y) = B_s(y)$. Thus for all $s \geq s_1$, $l_r(s) \geq x$. Hence $\liminf l_r(s) = \infty$
(iii) implies (i). This holds trivially. □

In the discussion to follow we will assume that for a certain $r = \langle W, \Phi, \Psi \rangle$, l_r is unbounded. We will use this knowledge to develop a strategy for meeting a requirement P_p that tightly controls what the set W can do.

Given that l_r is unbounded, suppose that at some stage s, $y < l_r(s)$ and $y \notin B_s$. At stage $s + 1$, we set $B_{s+1} = \{y\} \cup B_s$. We need to code the change to B into A, so we choose some $x \leq y$ with $x \notin A_s$ and set $A_{s+1} = \{x\} \cup A_s$. Our change to B has broken the computation of Ψ, i.e. $\Psi^W(y)[s] = B_s(y) = 0 \neq B_{s+1}(y)$. In order to fix the computation, the opponent must add something to W within the use of the computation of y. The use is bounded by $y + r$, so the opponent must add some number to W less than or equal to $y + r$. However, the opponent also wants to ensure that $\Phi^A = W$. Now because we coded the B change into A by adding x to A, we have given the opponent the opportunity to redefine the Φ functional. This allows the opponent to add something to W_{s+1} and also have that $\Phi^A[s+1] \upharpoonright l_r(s) = W_{s+1} \upharpoonright l_r(s)$. However, $x < l_r(s) \leq d_s(W_s, \Phi^A[s])$, hence if $z < x - r$, $\Phi^A(z)[s+1] = \Phi^A(z)[s] = W_s(z)$ because A has not changed within the use of z. Hence if $W_{s+1}(z) \neq W_s(z)$ the Φ computation will be broken. If we preserve A on $z + r$ then this computation can never recover. Consequently in order to fix Ψ without breaking Φ, W must change between $x - r$ and $y + r$.

To meet a requirement P_p, we select an $x \notin A_s$ and $y \notin B_s$ with $x + p < y$ at a stage s at which $m_p(s) > x$ and $l_r(s) > y$. Then we add x to A_{s+1} and y to B_{s+1}. So $\Delta_p^B(x)[s+1] = \Delta_p^B(x)[s] = 0 \neq A_{s+1}(x)$ as the change to B is outside the use of the computation of x. Additionally, we know that in order to fix the computation Ψ, W must change between $x - r$ and $y + r$. We will refer to this process of adding x to A and y to B as *diagonalising against* P_p.

Of course this still gives the opponent a number of choices for where W can change. We will show that we can construct an interval where the opponent has only two choices. Let us assume that for some $p > 2r$, we want to diagonalise against P_p and restrict the places where W can change in response. As we control A and B, we can find an integer interval $[b, c]$, where $c \geq 3p + b$ and a stage s, such that:

$$[b, c] \setminus A_s = [b, c] \setminus B_s = \{b + p, c - p\}.$$

We would like to have $|[b, c] \setminus W_s| = 2$. Assume this is true for some interval $\mathcal{I} = [b, c]$ at stage s. To maintain consistency with future notation, let $x(\mathcal{I}) = b + p, y(\mathcal{I}) = c - p$ be the positions of the two zeros in $A_s[b, c]$ and $B_s[b, c]$. Because $c \geq 3p + b$, it follows that $x(\mathcal{I}) < y(\mathcal{I})$. Let $x(\mathcal{I}, r), y(\mathcal{I}, r)$ be the positions of the two zeros in $W_s[b, c]$ with $x(\mathcal{I}, r) < y(\mathcal{I}, r)$. We will also assume that $|x(\mathcal{I}) - x(\mathcal{I}, r)| \leq r$ and that $|y(\mathcal{I}) - y(\mathcal{I}, r)| \leq r$, i.e. the zeros in W_s are within r places of the zeros in A_s and B_s.

If we add $x(\mathcal{I})$ to A and $y(\mathcal{I})$ to B, then we know that some element of $[x(\mathcal{I}) - r, y(\mathcal{I}) + r]$ must be added to W. The only elements left in this interval are $x(\mathcal{I}, r)$ and $y(\mathcal{I}, r)$. If W responds by adding $x(\mathcal{I}, r)$, then as we know that $x(\mathcal{I}, r) \leq x(\mathcal{I}) + r$, we can use this change in W to permit the change we have already made to A. Otherwise, if W responds by adding $y(\mathcal{I}, r)$, then as we know that $y(\mathcal{I}, r) \leq y(\mathcal{I}) + r$, we can regard our change in B as coding the change in W. In the first case, we will say that W *follows* A during the diagonalisation of P_p. In the second case we will say that W *follows* B.

Even assuming that we can create such intervals, there is more to be done. There is an infinite number of diagonalisation requirements that need to be met. It is possible that for infinitely many of these, W could follow A and for infinitely many of these, W could follow B. First, note that if W almost always follows B then, after some point, every W change made during some diagonalisation is coded by a change to B within r positions of the change to W. We can use this fact to show that $W \leq_{cl} B$.

Now if W does not almost always follow B, then infinitely often, W must follow A. Our construction will deal with this outcome by ensuring that $A \leq_{cl} W$. To outline how this will work, suppose that by some stage s we construct

two diagonalisation intervals $\mathcal{I}^p = [b_p, c_p]$ and $\mathcal{I}^q = [b_q, c_q]$ such that $b_p < c_p < b_q < c_q$.

$\mathcal{I}^p$ and $\mathcal{I}^q$ are used to meet requirements P_p and P_q respectively with $p < q$. Further suppose that at stage s we can ensure that $|W_s(c_p, b_q)| \geq |A_s(c_p, b_q)|$ and $A_s(c_p, b_q) = B_s(c_p, b_q)$.

Now what happens if W has followed A on interval $\mathcal{I}^q$, and W has followed B on interval $\mathcal{I}^p$? In this case we have that $y(\mathcal{I}^p, r), x(\mathcal{I}^q, r) \in W_s$. Note that $y(\mathcal{I}^p) \notin A_s$ and $x(\mathcal{I}^q) \notin B_s$. Thus we can define $A_{s+1} = A_s \cup \{y(\mathcal{I}^p)\}$ and $B_{s+1} = B_s \cup \{x(\mathcal{I}^q)\}$. How can the opponent respond? Well, the opponent must respond by adding some element of $[y(\mathcal{I}^p) - r, x(\mathcal{I}^q) + r]$ to W. However, $[y(\mathcal{I}^p) - r, c_p] \cup [b_q, x(\mathcal{I}^q) + r] \subseteq W_s$. Hence the opponent can only respond by adding some element of (c_p, b_q) to W_{s+1}. But this means that

$$|W_{s+1}(c_p, b_q)| > |W_s(c_p, b_q)| \geq |A_s(c_p, b_q)| = |A_{s+1}(c_p, b_q)|.$$

In this case we have a simple strategy to ensure that l_r is bounded. As $A_s(c_p, b_q) = B_s(c_p, b_q)$ we can add an element of (c_p, b_q) to A and to B each time l_r exceeds b_q. To make l_r exceed b_q again, some element of (c_p, b_q) must be added to W. As $|W_{s+1}(c_p, b_q)| > |A_{s+1}(c_p, b_q)|$ at some point the opponent must run out of responses.

So if W follows A on the interval $\mathcal{I}^q$, and l_r is unbounded, then W must follow A on the interval $\mathcal{I}^p$ as well. We will use this idea to develop a construction during which if infinitely often W follows A, then $W \equiv_{cl} A$. The basic idea is that we prefer our requirements P_p to be assigned diagonalisation intervals like $\mathcal{I}^p$ that precede an interval $\mathcal{I}^q$ where W has followed A. Hence if a requirement P_p has the opportunity to obtain an interval with this property, it will do so (provided it does not injure any higher priority requirements). Now to compute $A(x)$ from $W \upharpoonright\upharpoonright (x + r)$ we do the following. Run the construction of A and B until a stage s such that:

(i). $W_s \upharpoonright\upharpoonright (x + r) = W \upharpoonright\upharpoonright (x + r)$.

(ii). All diagonalisation requirements assigned intervals before x have either already diagonalised, or have been assigned a diagonalisation interval before one in which W has followed A.

Now if any of these requirements do diagonalise, W must follow A on their interval. Thus there must be some change to W_s within r places of the change to A. However, as $W_s \upharpoonright\upharpoonright (x + r) = W \upharpoonright\upharpoonright (x + r)$, no such requirement can diagonalise and so $A_s(x) = A(x)$.

9.3 Diagonalisation intervals and blocks

We need to formalise the concept of a diagonalisation interval introduced in the previous section.

Definition 9.3.1. A *diagonalisation interval* is a quadruple $\mathcal{I} = \langle b, c, s, k\rangle$ such that:

(i). $c = b + 6k + 1$.

(ii). $[b, c] \setminus A_s = \{b + 2k, c - 2k\}$.

(iii). $[b, c] \setminus B_s = \{b + 2k, c - 2k\}$.

The elements of $[b, c]$ that are not in A_s and B_s are important. We will define $x(\mathcal{I}) = b + 2k$ and $y(\mathcal{I}) = c - 2k$.

A diagonalisation interval $\mathcal{I} = \langle b, c, s, k\rangle$ is *suitable* for some $r \in \omega$ if :

(i). $r \leq k$.

(ii). $|[b, c] \setminus W_{r,s}| = 2$.

(iii). $|[x(\mathcal{I}) - r, x(\mathcal{I}) + r] \setminus W_{r,s}| = 1$.

(iv). $|[y(\mathcal{I}) - r, y(\mathcal{I}) + r] \setminus W_{r,s}| = 1$.

Definition 9.3.2. Given $\mathcal{I} = \langle b, c, s, k\rangle$, we define the following functions:

(i). $R(\mathcal{I}) = \{r \in \omega : r$ is suitable for $\mathcal{I}\}$.

(ii). $x(\mathcal{I}, r) =$ the unique element of $[x(\mathcal{I}) - r, x(\mathcal{I}) + r] \setminus W_{r,s}$ if $r \in R(\mathcal{I})$, otherwise undefined.

(iii). $y(\mathcal{I}, r) =$ the unique element of $[y(\mathcal{I}) - r, y(\mathcal{I}) + r] \setminus W_{r,s}$ if $r \in R(\mathcal{I})$, otherwise undefined.

Suppose that we have a diagonalisation interval $\mathcal{I}$ and that $r \in R(\mathcal{I})$. We need to know how W_r responds to a diagonalisation on $\mathcal{I}$. The function g^A defined below records those elements of $R(\mathcal{I})$ that follow A, and the function g^B those elements that follow B.

Definition 9.3.3. (i). $g^A(\mathcal{I}, t) = \{r \in R(\mathcal{I}) : x(\mathcal{I}, r) \in W_{r,t}\}$.

(ii). $g^B(\mathcal{I}, t) = \{r \in R(\mathcal{I}) : y(\mathcal{I}, r) \in W_{r,t}\}$.

It is possible for the sets $g^A(\mathcal{I}, t)$ and $g^B(\mathcal{I}, t)$ to have non-empty intersection.

The following lemma provides our strategy for meeting a requirement P_p using an appropriate diagonalisation interval.

Lemma 9.3.4. *If $\mathcal{I} = \langle b, c, s_0, k\rangle$ is a diagonalisation interval, $R \subseteq R(\mathcal{I})$ and $p \leq 2k$ then there is a strategy starting at s_0 to ensure that either:*

(i). For some $r \in R$, l_r is bounded; or

(ii). Requirement P_p is met, and if $x(\mathcal{I}) \in A$ then $R \subseteq g^A(\mathcal{I}, s_1) \cup g^B(\mathcal{I}, s_1)$ for some $s_1 > s_0$.

Proof. Assume that for all $r \in R$, l_r is unbounded. We only act at stages $s \geq s_0$ at which all lengths of agreement l_r exceed c. The set of such stages must be cofinite by Lemma 9.2.2. Now if m_p never exceeds c during such a stage s, then m_p is bounded so requirement P_p is met. Otherwise if $m_p(s) > c$ for some such stage s, then we set $A_{s+1} = A_s \cup \{x(\mathcal{I})\}$ and $B_{s+1} = B_s \cup \{y(\mathcal{I})\}$. As $y(\mathcal{I}) - x(\mathcal{I}) > 2k \geq p$, P_p is met.

We then wait until a stage s_1 at which for all $r \in R$, $l_r(s_1) > c$. If this happens, then for any $r \in R$, there must be some element of $[b, c]$ in $W_{r,s_1} \setminus W_{r,s}$. But this means that either $x(\mathcal{I}, r)$ or $y(\mathcal{I}, r)$ are in W_{r,s_1} and so $r \in g^A(\mathcal{I}, s_1) \cup g^B(\mathcal{I}, s_1)$. □

Now we will prove that it is possible to construct diagonalisation intervals.

Lemma 9.3.5. *Given a finite subset R of ω, $k \geq \max(R)$, $s_0 \in \omega$, and an interval $[a, d]$ where $d = a + (2k^2 + 1)(6k + 2) - 1$ such that $A_{s_0}[a, d] = B_{s_0}[a, d] = \emptyset$, there is an strategy that either:*

(i). Ensures, for some $r \in R$, that l_r is bounded; or

(ii). Ensures that for some stage $s_1 > s_0$, there is a diagonalisation interval $\mathcal{I} = \langle b, c, s_1, k\rangle$ such that $a \leq b < c \leq d$ and $\mathcal{I}$ is suitable for all $r \in R$, i.e. $R \subseteq R(\mathcal{I})$.

Proof. Our approach is to build a large enough number of diagonalisation intervals so that we can argue, by the pigeonhole principle, that one of them must be suitable for all $r \in R$. As each diagonalisation interval has length $6k + 2$, there is enough space in the interval $[a, d]$ for $2k^2 + 1$ diagonalisation intervals. For $i \in \omega$ such that $0 \leq i < 2k^2 + 1$, let $b_i = a + i(6k + 2)$ and $c_i = a + (i + 1)(6k + 2) - 1$. The ith diagonalisation interval that we will build will be on the interval $[b_i, c_i]$. On each of these diagonalisation intervals we want to add all but two elements to A and B. The only elements we will not add will be the $b_i + 2k$ and $c_i - 2k = b_i + 4k + 1$ positions in each diagonalisation interval. We call these elements the *zeros* and define:

$$Z = \{b_i + z : 0 \leq i < 2k^2 + 1, z \in \{2k, 4k + 1\}\}$$

to be the set of all zeros. Hence the set of elements we want to add to A and B is: $X = [a, d] \setminus Z$. The construction proceeds in two phases, the build phase and the review phase.

Build phase. If $l_r(s) < d$ for some $r \in R$, then set $A_{s+1} = A_s$ and $B_{s+1} = B_s$. Repeat the build phase. If for all $r \in R$, $l_r(s) \geq d$ and $A_s[a, d] = X$, go to the review phase. Otherwise choose some element x in $X \setminus A_s[a, d]$, set $A_{s+1} = A_s \cup \{x\}$ and $B_{s+1} = B_s \cup \{x\}$ and repeat the build phase.

Review phase. Let s_1 be the current stage. If for some i, $\langle b_i, c_i, s_1, k\rangle$ is a diagonalisation interval that is suitable for all $r \in R$ then finish. Otherwise, if there is some $z \in Z$ and some $r \in R$ such that, $[z - r, z + r] \subseteq W_{r,s_1}$, set $A_{s_1+1} = A_{s_1} \cup \{z\}$ and $B_{s_1+1} = B_{s_1} \cup \{z\}$ and terminate the algorithm.

To verify the algorithm, first note that if the review phase is never reached, then by Lemma 9.2.2, this must be because for some $r \in R$, l_r is bounded. Now let us consider what occurs if the review phase is reached. Let s_1 be the stage when the review phase is reached. Take any $r \in R$. We know that each change to B has forced a change to W_r within r places of the change to B. As $k \geq r$, we have that:

$$|W_{r,s_1}[a, d]| \geq |B_{s_1}[a, d]| - 2k. \tag{9.3.1}$$

Now assume that for all $z \in Z$, for all $r \in R$, $[z - r, z + r] \not\subseteq W_{r,s_1}$. Then as there are two zeros in each diagonalisation interval $[b_i, c_i]$, and as $k \geq \max(R)$, we have that: $|W_{r,s_1}[b_i, c_i]| \leq |[b_i, c_i]| - 2 = |B_s[b_i, c_i]|$.

If there are more than $2k$ intervals i where $|W_{r,s_1}[b_i, c_i]| < |B_s[c_i, d_i]|$ then:

$$\begin{aligned} |W_{r,s_1}[a, d]| &= \sum_{i=0}^{2k^2} |W_{r,s_1}[b_i, c_i]| \\ &< \sum_{i=0}^{2k^2} |B_{s_1}[b_i, c_i]| - 2k \\ &= |B_{s_1}[a, d]| - 2k. \end{aligned}$$

This would contradict (9.3.1), hence there can be at most $2k$ intervals i where:

$$|W_{r,s_1}[b_i, c_i]| < |B_s[b_i, c_i]|.$$

Thus there can be at most $2k|R| \leq 2k^2$ diagonalisation intervals where for some $r \in R$, $|W_{r,s}[b_i, c_i]| < |B_s[b_i, c_i]|$. As there are $2k^2 + 1$ intervals, there is some interval $[b_i, c_i]$ such that for all $r \in R$, $|W_{r,s}[b_i, c_i]| = |B_s[b_i, c_i]|$. Hence if we let $\mathcal{I} = \langle b_i, c_i, k, s\rangle$ then we have that $R \subseteq R(\mathcal{I})$.

The contrapositive of the proceeding argument gives us that if there are no diagonalisation intervals that include R, then there must be some $z \in Z$ and $r \in R$ such that $[z-r, z+r] \subseteq W_{r,s_1}$. In this case the construction adds this z to A and B and as there is no position where W_r can respond, l_r is bounded. $\square$

The approach outlined in Section 9.2 to meeting one requirement R_r where l_r is unbounded can be extended to two requirements r_0 and r_1 *provided* that we know whether or not both l_{r_0} and l_{r_1} are unbounded. The interesting case is if they are both unbounded. For this situation we build a pair of diagonalisation intervals $\mathcal{I}^0 = \langle b_0, c_0, s, p\rangle$ and $\mathcal{I}^1 = \langle b_1, c_1, s, p\rangle$ for each requirement P_p. The intervals would have the property that $\{r_0, r_1\} \subseteq R(\mathcal{I}^0) \cap R(\mathcal{I}^1)$ and $c_0 < b_1$. We would assign the requirement P_p to the second interval $\mathcal{I}^1$. Now if we diagonalise against P_p, then there are four possible ways that these two R requirements could respond: both could follow B, only r_1 could follow B, only r_0 could follow B, or both could follow A. We assign these outcomes the strings 00, 01, 10 and 11 respectively (later we will refer to these as e-states). Interpreting the strings as binary values, if the outcome is greater than 0 (i.e. at least one requirement follows A) then we look for any higher priority P requirements that have not diagonalised. We consider those requirements currently assigned the second in a pair of intervals. We also consider those requirements assigned the first in a pair in intervals such that the second interval, in the same pair, has a lower value than $\mathcal{I}^1$. We take the highest priority such requirement, assign it the interval $\mathcal{I}^0$, and injure all lower priority requirements. As we move a requirement at most three times (the number of times that the e-state can be improved), we can still meet all our P requirements through a finite injury argument.

Requirements r_0 and r_1 are met through a similar argument to that given in Section 9.2. However, if r_1 follows A infinitely often, then to show that $A \equiv_{cl} W_{r_1}$ we need to know whether requirement r_0 follows A infinitely often or not.

The problem with this approach is that in order to generalise this argument to an infinite number of R requirements, it is not possible for r_0 to know whether or not l_{r_1} is bounded. Our opponent could pretend that l_{r_1} is bounded until we have built the intervals $\mathcal{I}^0$ and $\mathcal{I}^1$. This would mean that neither of these intervals could be suitable for r_1. After we diagonalise on $\mathcal{I}^1$, then l_{r_1} could recover. However, we have already built interval $\mathcal{I}^0$. If we attempt to diagonalise on this interval now, then we have placed no real restriction on how the set W_{r_1} can change in response.

To solve this problem, we need to *delay* the construction of the interval $\mathcal{I}^0$

until after the diagonalisation on $\mathcal{I}^1$ has occurred. To achieve this, we need a more elaborate strategy and we will introduce the idea of blocks. A block is a way of dividing the integers into separate areas that we can allocate to different requirements.

Definition 9.3.6. A *block* is a quadruple $\langle b, c, s, k\rangle$ such that:

(i). $c = b + k - 1$.

(ii). $[b, c] \subseteq A_s$.

(iii). $[b, c] \subseteq B_s$.

A block $\mathcal{B} = \langle b, c, s, k\rangle$ *encompasses* some $r \in \omega$ if :

(i). $r \leq k$.

(ii). $[b, c] \subseteq W_{r,s}$.

From this definition we define the function:

$$R(\mathcal{B}) = \{r \in \omega : \mathcal{B} \text{ encompasses } r\}.$$

If $\mathcal{B}^0 = \langle b_0, c_0, s_0, k\rangle$ and $\mathcal{B}^1 = \langle b_1, c_1, s_1, k\rangle$, and $c_0 < b_1$, then we will write $\mathcal{B}^0 < \mathcal{B}^1$. This indicates that $\mathcal{B}^0$ occurs before $\mathcal{B}^1$. We will use $<$ in a similar way to compare diagonalisation intervals, or diagonalisation intervals and blocks.

Blocks are useful because they segment the sets A, B and W_r into what can be regarded as separate games. If we have two blocks $\mathcal{B}^0 < \mathcal{B}^1$, that both include r, then any change to A and B between two blocks must be met by a change to W_r between the same two blocks. Let c_0 denote the right end point of $\mathcal{B}^0$ and let b_1 be the left end point of $\mathcal{B}^1$. If we keep A and B identical on the interval (c_0, b_1) then the opponent is in trouble if at any stage s, $|W_{r,s}(c_0, b_1)| > |A_s(c_0, b_1)|$. The following lemma explains why.

Lemma 9.3.7. *Assume that $\mathcal{B}^0 = \langle b_0, c_0, s_0, k\rangle$ and $\mathcal{B}^1 = \langle b_1, c_1, s_0, k\rangle$ are two blocks with $\mathcal{B}^0 < \mathcal{B}^1$ and $A_{s_0}(c_0, b_1) = B_{s_0}(c_0, b_1)$. If for some $r \leq k$, $|W_{r,s_0}(c_0, b_1)| > |A_{s_0}(c_0, b_1)|$ then there is a strategy to ensure that l_r is bounded.*

Proof. The strategy is as follows. At each stage $s \geq s_0$, such that $l_r(s) \geq b_1$, we choose an element x of $(c_0, b_1) \setminus A_s$. We set $A_{s+1} = A_s \cup \{x\}$ and $B_{s+1} = B_s \cup \{x\}$.

This strategy works because each time we add an element x to A and B, W_r must respond by making adding some element from $[x - r, x + r]$. But as $[b_0, c_0] \cup [b_1, c_1] \subseteq W_{r,s}$ it must be an element from (c_0, b_1) that is added in order for the length of agreement l_r to exceed b_1 again. Hence at all stages s where $l_r(s) \geq b_1$ we have that $|W_{r,s}(c_0, b_1)| > |A_s(c_0, b_1)|$. It follows that at some point W_r will run out of possible responses and l_r will be bounded. □

Let us establish a strategy for constructing blocks.

Lemma 9.3.8. *Given any finite subset of R of ω, $k \geq \max(R)$, $s_0 \in \omega$ and interval $[a, d]$ where $d = a + (2k^2 + 1)k - 1$, there is a strategy starting at s_0 to ensure that either:*

(i). For some $r \in R$, l_r is bounded; or

(ii). At some stage $s_1 > s_0$ there is a block $\mathcal{B} = \langle b, c, s_1, k\rangle$ such that $a \leq b < c \leq d$ and $\mathcal{B}$ encompasses R.

Proof. The proof is similar to the proof of Lemma 9.3.5. At each stage s at which $l_r(s) > d$ for all $r \in R$, we take some element $x \in [a, d] \setminus A_s$ and set $A_{s+1} = A_s \cup \{x\}$ and $B_{s+1} = B_s \cup \{x\}$. We stop at some stage s_1 if $[a, d] \subseteq A_{s_1}$ and $l_r(s_1) > d$ for all $r \in R$.

Now $|W_{r,s_1}[a, d]| \geq |A_{s_1}[a, d]| - 2r \geq |A_{s_1}[a, d]| - 2k = d - a + 1 - 2k$. For all $i \in \omega$, such that $0 \leq i < 2k^2 + 1$, let $b_i = a + ik$ and let $c_i = a + (i + 1)k - 1$. There are at most $2k$ possible values for i such that $|W_r[b_i, c_i]| \neq c_i - b_i + 1$. Hence as $2k|R| \leq 2k^2$ there must be some i such that for all $r \in R$, $|W_r[b_i, c_i]| = c_i - b_i + 1 = |A_s[b_i, c_i]|$. □

We want to be able to build diagonalisation intervals $\mathcal{I}^0 < \mathcal{I}^1$ such that if $\mathcal{I}^0$ and $\mathcal{I}^1$ are both suitable for some r, then r cannot both follow A on $\mathcal{I}^1$ and follow B on $\mathcal{I}^0$. The following lemma describes a situation in which this holds.

Lemma 9.3.9. *Let $\mathcal{I}^0 = \langle b_0, c_0, s_0, k_0\rangle$ and $\mathcal{I}^1 = \langle b_1, c_1, s_1, k_0\rangle$ be two diagonalisation intervals such that $\mathcal{I}^0 < \mathcal{I}^1$. Suppose that at stage $s_2 \geq \max(s_0, s_1)$ the following conditions are met for some $r \in R(\mathcal{I}_0) \cap R(\mathcal{I}_1)$:*

(i). $y(\mathcal{I}^0) \notin A_{s_2}$.

(ii). $x(\mathcal{I}^1) \notin B_{s_2}$.

(iii). $A_{s_2}(c_0, b_1) = B_{s_2}(c_0, b_1)$.

(iv). $|W_{r,s_2}(c_0, b_1)| \geq |A_{s_2}(c_0, b_1)|$.

If $r \in g^A(\mathcal{I}^1, s_2) \cap g^B(\mathcal{I}^0, s_2)$, then there is a strategy to ensure that l_r is bounded.

Proof. Adopt the following strategy. Wait until some stage $s > s_2$ at which $l_r(s) > c$. Define $A_{s_2+1} = A_{s_2} \cup \{y(\mathcal{I}^0)\}$ and $B_{s_2+1} = B_{s_2} \cup \{x(\mathcal{I}^1)\}$. If at some stage $s' > s$, $l_r(s') > c$ again then some element of $[y(\mathcal{I}^0) - r, x(\mathcal{I}^1) + r]$ must have been added to $W_{r,s'}$. As $r \in g^A(\mathcal{I}^1, s_2) \cap g^B(\mathcal{I}^0, s_2)$, $x(\mathcal{I}^1, r) \in W_{r,s_2}$ and

$y(\mathcal{I}^0, r) \in W_{r,s_2}$. This means that only elements of (c_0, b_1) could have been added to $W_{r,s'}$. Thus $|W_{r,s'}(c_0, b_1)| > |A_{s'}(c_0, b_1)|$ and so by Lemma 9.3.7 we have a strategy to ensure that l_r is bounded. □

If we consider the conditions of the previous lemma, items (i), (ii) and (iii) are under our control. To be able to make use of this lemma we need a means of ensuring (iv) occurs as well. Because of the importance of this item we will introduce the following definition.

Definition 9.3.10. Let $\mathcal{I}_0 = \langle b_0, c_0, s_0, k_0\rangle$ and $\mathcal{I}_1 = \langle b_1, c_1, s_1, k_1\rangle$ be two diagonalisation intervals with $\mathcal{I}_0 < \mathcal{I}_1$, and $s \geq \max(s_0, s_1)$. If $R \subseteq R(\mathcal{I}^0) \cap R(\mathcal{I}^1)$, then we will say that $\mathcal{I}_0$ and $\mathcal{I}_1$ are *R-linked at stage* s if for all $r \in R$, $|W_{r,s}(c_0, b_1)| \geq |A_s(c_0, b_1)|$.

We will use a similar definition for blocks.

Definition 9.3.11. Let $\mathcal{B}^0 = \langle b_0, c_0, s_0, k_0\rangle$ and $\mathcal{B}^1 = \langle b_1, c_1, s_1, k_0\rangle$ be two blocks with $\mathcal{B}^0 < \mathcal{B}^1$. If $R \subseteq R(\mathcal{B}^0) \cap R(\mathcal{B}^1)$, then we will say that $\mathcal{B}_0$ and $\mathcal{B}_1$ are *R-linked at stage* s if for all $r \in R$, $|W_{r,s}(c_0, b_1)| \geq |A_s(c_0, b_1)|$.

In both cases we will say that blocks or diagonalisation intervals are r-linked if they are $\{r\}$-linked.

We will make extensive use of blocks and diagonalisation intervals. The following functions are useful because they represent the width of interval required by Lemma 9.3.5 to build a diagonalisation interval, and the width of interval required by Lemma 9.3.8 to build a block. We define:

$$w_{\mathcal{I}}(k) = (2k^2 + 1)(6k + 2) \text{ and } w_{\mathcal{B}}(k) = (2k^2 + 1)k.$$

9.4 Basic algorithm

Now we are ready to explain the basic algorithm that we will use in our main construction. This algorithm uses a combination of Lemmas 9.3.5 and 9.3.8. First we will outline the algorithm. We are given some finite $R \subset \omega$, and some $k > \max(R)$. We will assume that for all $r \in R$, l_r is unbounded. First by some stage s, we build a whole sequence of blocks $\mathcal{B}_0 = \langle b_0, c_0, s, k\rangle$, $\mathcal{B}_1 = \langle b_1, c_1, s, k\rangle, \ldots, \mathcal{B}_n = \langle b_n, c_n, s, k\rangle$ such that $\mathcal{B}_1 < \mathcal{B}_2 < \ldots < \mathcal{B}_n$. Each block will encompass R. We will show that by making n large enough, for some $i \in \{0, 1 \ldots, n-1\}$, for all $r \in R$, $|W_{r,s}(c_i, b_{i+1})| \geq |A_s(c_i, b_{i+1})|$. Now we will let $\mathcal{B}^0 = \mathcal{B}_i$ and let $\mathcal{B}^1 = \mathcal{B}_{i+1}$. Hence $\mathcal{B}^0$ and $\mathcal{B}^1$ are R-linked at stage s.

Between any two adjacent blocks we will have built a diagonalisation interval. We let $\mathcal{I} = \langle b, c, s, k\rangle$ be the diagonalisation interval between $\mathcal{B}^0$ and

$\mathcal{B}^1$. We will also make sure that there is space between $\mathcal{B}^0$ and $\mathcal{I}$ to run the basic algorithm again for any smaller k. To do this we need to know the space required by the algorithm. We will define this inductively.

$$w_A(k) = \begin{cases} 2 & \text{if } k = 0 \\ (2k^2+1)(w_{\mathcal{B}}(k) + w_A(k-1) + w_{\mathcal{I}}(k)) + w_{\mathcal{B}}(k) & \text{otherwise.} \end{cases}$$

Lemma 9.4.1 (Basic Algorithm)**.** *Given any finite $R \subset \omega$, and $k > \max(R)$, if at some stage s_0 for some $[a, d]$ with $d = a + w_A(k) - 1$, $A_{s_0}[a, d] = B_{s_0}[a, d] = \emptyset$, then there is a strategy to ensure that either:*

(1) For some $r \in R$, l_r is bounded; or

(2a) Case $k = 0$: At some stage $s_1 > s_0$ there is a diagonalisation interval $\mathcal{I} = \langle b, c, s_1, 0\rangle$ such that:

(a) $a \leq b$, and $c \leq d$.

(b) $A_{s_1}[a, d] = B_{s_1}[a, d]$.

(2b) Case $k > 0$: At some stage $s_1 > s_0$ there are two blocks $\mathcal{B}^0 = \langle b_0, c_0, s_1, k\rangle$, $\mathcal{B}^1 = \langle b_1, c_1, s_1, k\rangle$ and a diagonalisation interval $\mathcal{I} = \langle b, c, s_1, k\rangle$ such that:

(a) $a \leq b_0$, $\mathcal{B}^0 < \mathcal{I} < \mathcal{B}^1$, and $c_1 \leq d$.

(b) $A_{s_1}[a, d] = B_{s_1}[a, d]$.

(c) $R \subseteq R(\mathcal{B}^0) \cap R(\mathcal{B}^1) \cap R(\mathcal{I})$.

(d) $\mathcal{B}^0$ and $\mathcal{B}^1$ are R-linked at stage s_1.

(e) There exists $h \in (c_0, b_1)$ such that $A_{s_1}[h, h + w_A(k-1)] = \emptyset$.

Proof. First if $k = 0$, then $d = a + 1$ and $R = \emptyset$. So we can set $\mathcal{I} = \langle a, d, s_0, 0\rangle$ and the conditions are met.

For $k \geq 1$, assume that for all $r \in R$, l_r is unbounded. First we use Lemma 9.3.8, to build $n = 2k^2 + 2$ blocks $\mathcal{B}_0, \ldots, \mathcal{B}_{n-1}$ that all encompass R. Set $j = w_{\mathcal{B}}(k) + w_A(k-1) + w_{\mathcal{I}}(k)$. For all $i, 0 \leq i < n$, we use the interval $[a + ij, a + ij + w_{\mathcal{B}}(k) - 1]$ to build block $\mathcal{B}_i$. Secondly, we use Lemma 9.3.5 to build $n - 1$ diagonalisation intervals. We use the interval $[a + ij + w_{\mathcal{B}}(k) + w_A(k-1), a + ij + w_{\mathcal{B}}(k) + w_A(k-1) + w_{\mathcal{I}}(k) - 1]$ to build the diagonalisation interval $\mathcal{I}_i$.

Let s' be the stage at which these blocks and diagonalisation intervals are complete. Let $\langle b_i, c_i, s', k\rangle = \mathcal{B}_i$. Note that there are $2k^2 + 1$ intervals between the blocks we have created. These intervals are (c_i, b_{i+1}) for $0 \leq i \leq n - 1$. Now if for any $r \in R$ and i we have that $|W_{r,s'}(c_i, b_{i+1})| > |A_{s'}(c_i, b_{i+1})|$ then as $A_{s'}(c_i, b_{i+1}) = B_{s'}(c_i, b_{i+1})$ we can use Lemma 9.3.7 to ensure that l_r is bounded.

Assume this is not the case. We can ensure that we have only operated at stages s at which $l_r(s) \geq d$ for all $r \in R$. Hence it must be that $|W_{r,s'}[a,d]| \geq |A_{s'}[a,d]| - 2k$. This means there can only be $2k$ intervals with $|W_{r,s'}(c_i, b_{i+1})| < |A_{s'}(c_i, b_{i+1})|$. As $2k|R| \leq 2k^2$ this means that for some i we have that

$$|W_{r,s'}(c_i, b_{i+1})| = |A_{s'}(c_i, b_{i+1})|$$

for all $r \in R$. We let $\mathcal{B}^0 = \mathcal{B}_i$ and $\mathcal{B}^1 = \mathcal{B}_{i+1}$. Now we have that $\mathcal{B}^0$ and $\mathcal{B}^1$ are R-linked at stage s'. We set $\mathcal{I} = \mathcal{I}_i$. Finally if we take $h = a + ij + w_{\mathcal{B}}(k)$, then $A_{s_1}[h, h + w_A(k-1)] = B_{s_1}[h, h + w_A(k-1)] = \emptyset$. □

This basic algorithm is useful because we can use it to create linked intervals.

Lemma 9.4.2. *If the following hold:*

(i). We apply the basic algorithm for some k, $R \subset \omega$ with $k > \max(R)$ to some suitable interval $[a,d]$. At stage s_1 the algorithm halts giving us a diagonalisation interval $\mathcal{I}$ and parameter h.

(ii). At stage $s_2 > s_1$ we diagonalise a P requirement by adding $x(\mathcal{I})$ to A_{s_2} and $y(\mathcal{I})$ to B_{s_2}.

(iii). At stage $s_3 > s_2$ we are given $k' < k$ and $R' \subset \omega$ with $k' > \max(R')$ such that $R \cap R' \subseteq g^A(\mathcal{I}, s_3) \cup g^B(\mathcal{I}, s_3)$.

(iv). We apply the basic algorithm again on the interval $[h, h + w_A(k-1) - 1]$ for k' and R'. At stage s_4 the algorithm halts giving us an interval $\mathcal{I}'$.

Then for all $r \in R \cap R'$, $\mathcal{I}$ and $\mathcal{I}'$ are r-linked at stage s_4 or there is a strategy to ensure that l_r is bounded above.

Proof. Let blocks $\mathcal{B}^0 = \langle b_0, c_0, s_1, k\rangle$, $\mathcal{B}^1 = \langle b_1, c_1, s_1, k\rangle$, diagonalisation interval $\mathcal{I} = \langle b, c, s_1, k\rangle$ and parameter h be the results of the running the basic algorithm for the first time with $\mathcal{B}^0 < \mathcal{I} < \mathcal{B}^1$. Let $\mathcal{I}' = \langle b', c', s_4, k'\rangle$ be the diagonalisation interval produced by running the basic algorithm a second time. We have that: $\mathcal{B}^0 < \mathcal{I}' < \mathcal{I} < \mathcal{B}^1$ so that:

$$b_0 < c_0 < b' < c' < b < c < b_1 < c_1.$$

Take any $r \in R \cap R'$. The basic algorithm guarantees that $\mathcal{B}^0$ and $\mathcal{B}^1$ are R-linked at stage s_1. Hence:

$$|W_{r,s_1}(c_0, b_1)| \geq |A_{s_1}(c_0, b_1)|. \tag{9.4.1}$$

Now if $|A_{s_4}(c, b_1)| < |W_{r,s_4}(c, b_1)|$ we can force l_r to be bounded by adopting the strategy of Lemma (9.3.7). This is because the diagonalisation interval $\mathcal{I}$ contains a block at the end (i.e. $\langle c - k + 1, c, s_1, k\rangle$ is a block that encompasses r). So let us assume that $|A_{s_4}(c, b_1)| \geq |W_{r,s_4}(c, b_1)|$. As we make no changes to A on the interval (c, b_1) between s_1 and s_4, we have that:

$$|A_{s_1}(c, b_1)| = |A_{s_4}(c, b_1)| \geq |W_{r,s_4}(c, b_1)| \geq |W_{r,s_1}(c, b_1)|.$$

Further, as $|W_{r,s_1}[b, c]| = |A_{s_1}[b, c]|$, combining this with (9.4.1) gives that:

$$|W_{r,s_1}(c_0, b)| \geq |A_{s_1}(c_0, b)|.$$

Now as $r \in R'$, any change to A made during the second running of the algorithm is met by a change in W so we have that:

$$|W_{r,s_4}(c_0, b)| \geq |A_{s_4}(c_0, b)|.$$

We know that at stage s_4, $|W_{r,s_4}[b', c']| = |A_{s_4}[b', c']|$. Hence

$$|W_{r,s_4}(c_0, b')| + |W_{r,s_4}(c', b)| \geq |A_{s_4}(c_0, b')| + |A_{s_4}(c', b)|.$$

Now if $|W_{r,s_4}(c_0, b')| > |A_{s_4}(c_0, b')|$ then we can adopt the strategy of Lemma 9.3.7 and force l_r to be bounded. Otherwise we would have that $|W_{r,s_4}(c', b)| \geq |A_{s_4}(c', b)|$ and hence $\mathcal{I}$ and $\mathcal{I}'$ are r-linked at stage s_4. □

9.5 The priority tree

We have now developed the basic tools we need to prove the main theorem. We have an infinite number of R requirements, and an infinite number of P requirements. These requirements will be ordered by priority as follows:

$$P_0 > R_0 > P_1 > R_1 > P_2 > R_2 \ldots.$$

We use this ordering so that when we run a strategy to meet a requirement P_n, we only need to worry about the responses of those c.e. sets W_r with $r < n$.

There are two possible outcomes for an R requirement. Firstly, that l_r is bounded. This is the finite outcome f. Secondly, that l_r is unbounded. This is the infinite outcome i. In order to successfully run a strategy to meet a requirement P_n, we need to know, for all $r < n$, the outcome for requirement R_r. As we cannot know this in advance, we form a priority tree $\mathcal{T} = \{i, f\}^{<\omega}$. If $\alpha \in \mathcal{T}$ is a node of this tree, then α represents a guess as to the outcome of all requirements R_r with $r < |\alpha|$. The outcome node α is guessing for R_r is $\alpha(r)$.

We will order the tree with the i branch off any node to the left of the f branch. Given two nodes α and β, then α has higher priority than β, if α is to the left of β, or α is an initial segment of β.

If $\alpha \in \mathcal{T}$, then α will be assigned a status at all stages s. This status can be one of: *unassigned*, *building*, *waiting*, *diagonalised*, or *incorrect response*.

In a standard tree argument, at each stage in the construction of A and B, we would start at the top of the tree, with $\alpha = \lambda$, visit this node, then check to see if $l_{|\alpha|}$ has increased since our last visit to αi (αi is the string formed by appending i to the end of α). If it has, the next node we visit is αi. Otherwise we visit αf. We would continue this process until we get to some specified depth in the tree or some action on a node requires us to end the stage. The construction that we use will follow this idea with some modifications.

The basic idea is the following. The first time we visit a node α, or the first time we visit it after it has been injured, we assign α a work space $[a, a + w_A(|\alpha|)]$. At this point we change the status of α from *unassigned* to *building*. Each subsequent time we visit α, we apply a step of the basic algorithm with $k = |\alpha|$ and $R = \{r < |\alpha| : \alpha(r) = i\}$. Now if α's guess is correct then at some stage the basic algorithm will terminate. At this point we set the status of α to *waiting*.

We will define two functions that allow us to track the outputs of this basic algorithm. These are: $i : \mathcal{T} \times \omega \to \omega$ and $h : \mathcal{T} \times \omega \to \omega$. The value $i(\alpha, s)$ takes is equal to the diagonalisation interval assigned to α at stage s if such an interval exists. Otherwise $i(\alpha, s)$ is undefined. The function $h(\alpha, s)$ is defined if and only if $i(\alpha, s)$ is defined. If defined, the value of $h(\alpha, s)$ is the position in the work space at which the algorithm can be run again (the h parameter in the outcome of the basic algorithm).

If α is visited at stage s, at this stage α has status *waiting*, and $m_{|\alpha|}$ exceeds the diagonalisation interval built for α, then we use the interval to meet requirement $P_{|\alpha|}$ through the diagonalisation strategy of Lemma 9.3.4. We set the status of α to *diagonalised*.

However, this alone is not enough. As it stands, if we assume that α guesses correctly, then for all $r < \alpha$ with $\alpha(r) = i$ there are two possible ways that W_r could respond. W_r could follow A or follow B. In order to ensure that $W_r \equiv_{cl} A$, or $W_r \equiv_{cl} B$ we need to make use of linked intervals and Lemma 9.3.9.

Consider the following. We have two nodes α and β. Node α has higher priority than β and $|\beta| > |\alpha|$. Node β is in the *waiting* stage and for β we have constructed diagonalisation blocks $\mathcal{B}^0$, $\mathcal{B}^1$ and a diagonalisation interval $\mathcal{I}_\beta$. Now assume that at some stage, β diagonalises, and W_r has responded for all

$r < |\alpha|$ such that $\beta(r) = i$. A new work space can be assigned to α between $\mathcal{B}^0$ and $\mathcal{I}_\beta$. We apply the basic algorithm again and construct a diagonalisation interval $\mathcal{I}_\alpha$ in this work space. By Lemma 9.4.2, either $\mathcal{I}_\alpha$ and $\mathcal{I}_\beta$ are r-linked at this stage for all r such that $\alpha(r) = \beta(r) = i$; or we can ensure that l_r is bounded and hence the guess made by α is incorrect. Now because these two intervals are linked, if W_r followed A during the diagonalisation on $\mathcal{I}_\beta$, then W_r must follow A again on any subsequent diagonalisation on $\mathcal{I}_\alpha$. If not then there is a strategy to ensure that l_r is bounded (see Lemma 9.3.9) .

To understand how the diagonalisation has been responded to, we will define a function $f : \mathcal{T} \times \omega \to 2^{<\omega}$ that describes how sets W_r respond to a diagonalisation. The function $f(\gamma, s)$ is defined if and only if γ has status at stage s of *diagonalised*. In this case, if $\mathcal{I} = i(\gamma, s)$ then:

$$|f(\gamma, s)| = \max\{x < |\gamma| : \forall r \leq x, (\gamma(r) = i) \to (r \in g^A(\mathcal{I}, s) \cup g^B(\mathcal{I}, s))\} + 1.$$

We define $|f(\gamma, s)|$ this way to make sure that all sets W_r where $\gamma(r) = i$ and $r < |f(\gamma, s)|$ have responded. The rth bit of $f(\gamma, s)$ is then defined by:

$$f(\gamma, s)(r) = \begin{cases} 1 & \text{if } \gamma(r) = i \text{ and } r \in g^A(\mathcal{I}, s) \\ 0 & \text{otherwise.} \end{cases}$$

We will gain more control of how W_r responds to a diagonalisation by attempting to give each node α the best possible work space. We will define a function $e : \mathcal{T} \times \omega \to 2^{<\omega}$ that specifies the type of work space assigned to α at stage s. The function $e(\alpha, s)$ will be defined if α has a work space at stage s. We will ensure that if it is defined, then $|e(\alpha, s)| = |\alpha|$. We call $e(\alpha, s)$ the e-state of α at stage s. We will try to maximise the e-state of a node (we can consider the e-state as a binary value and try to maximise this value). We will ensure that if at any stage s, for any $r < |\alpha|$, we have that $e(\alpha, s)(r) = 1$, then α has been assigned a work space inside the work space that was assigned to some lower priority node β. Further, at some point β diagonalised on this work space and after the diagonalisation, W_r followed A. Hence if α diagonalises, then either W_r will follow A or we have a strategy to ensure that l_r is bounded by Lemma 9.4.2.

Now assume that W_r does not diagonalise as we would like. We have a strategy to ensure that l_r is bounded, but which node should we get to implement the strategy? We cannot give it to α because there is no guarantee that α will be visited again. We let $\gamma = \alpha \upharpoonright\!\uparrow r$ and we get γ to implement the strategy. We do this by assigning γ a status of *incorrect response*. Of course γ may not be visited again but this would be caused by γ's guess being wrong for an even higher priority requirement.

If for some node α at some stage s we have that $e(\alpha, s)(r) = 1$ for some $r < |\alpha|$, then α has been assigned a diagonalisation interval $\mathcal{I}_\alpha$ that is linked to the diagonalisation interval $\mathcal{I}_\beta$ originally created for a lower priority node β. It is useful to be able to reference this interval. Hence we will define a function $n : \mathcal{T} \times \omega \to \omega$ where $n(\alpha, s)$ is defined if and only if for some $r < |\alpha|$, $e(\alpha, s)(r) = 1$. In this situation, we set $n(\alpha, s)$ to be the interval $\mathcal{I}_\beta$.

Following is a summary of the functions we are using to track different aspects of the construction:

(i). $f(\alpha, s)$ – the responses to α's diagonalisation.

(ii). $e(\alpha, s)$ – e-state of current work space assigned to α.

(iii). $i(\alpha, s)$ – the diagonalisation interval assigned to α at stage s.

(iv). $h(\alpha, s)$ – the position at which some a new work space can be built within the current work space of α.

(v). $n(\alpha, s)$ – the diagonalisation interval that $i(\alpha, s)$ is linked to.

9.6 Construction

Our objective is to construct c.e. sets A, B such that:

(i). $B \leq_{cl} A$.

(ii). For all $p \in \omega$, $P_p : A \neq \Delta_p^B$.

(iii). For all $r \in \omega$, R_r : If $W_r = \Phi_r^A$ and $B = \Psi_r^{W_r}$, then there exists a Γ such that $W_r = \Gamma^B$ or $A = \Gamma^{W_r}$ and $\gamma(n) \leq n + r$.

The requirement that $B \leq_{cl} A$ will be achieved by coding any B change into A. Injuring a node α at stage s is equivalent to:

(i). Setting the status of α to *unassigned*.

(ii). Setting $i(\alpha, s+1)$, $f(\alpha, s+1)$, $h(\alpha, s+1)$, $e(\alpha, s+1)$, and $n(\alpha, s+1)$ to be undefined.

We construct A and B as follows. At stage 0, set $A_0 = B_0 = \emptyset$, set all nodes to have status *unassigned*. Set $e(\alpha, 0)$, $i(\alpha, 0)$, $h(\alpha, 0)$, $n(\alpha, 0)$ to be undefined for all α.

We will perform one of three tasks at stage $s + 1$. First we look to see if we can force some l_r to be bounded because W_r has not responded as required by

the e-state during some diagonalisation. If we cannot do this, then we look to see if we can improve the e-state of any node. Finally, if we cannot do either of the first two tasks, we access the requirement tree until we find some node that needs attention. After completing one of these three tasks, we undertake the steps listed in close stage.

Task One. Check if there exists a node γ such that for some $r < |f(\gamma, s)|$,

(i). $f(\gamma, s)(r) = 0$.

(ii). $\gamma(r) = i$.

(iii). $e(\gamma, s)(r) = 1$.

If such a node exists, then take γ to be the highest priority such node and let r be the least such r. Let $\alpha = \gamma \upharpoonright r$. Set the status of α to *incorrect response*.

Task Two. Check if there exist any nodes α and β such that:

(i). The current status of α is *waiting*.

(ii). α has higher priority than β.

(iii). $|\alpha| < |\beta|$.

(iv). $e(\alpha, s) < f(\beta, s) \upharpoonright |\alpha|$.

If such an α and β exist, then find the highest priority such α, and any such β. We set $e(\alpha, s+1) = f(\beta, s) \upharpoonright |\alpha|$. As $f(\beta, s)$ is defined, β must have the status *diagonalised* and so $h(\beta, s)$ is defined. We reassign the interval $[h(\beta, s), h(\beta, s) + w_A(|\alpha|)]$ to be the work space for α. As α is now nested inside the interval once assigned to β we set $n(\alpha, s+1) = i(\beta, s)$.

Task Three. We access the tree. Let $\alpha_{0,s} = \lambda$. We use substages t for all $0 \leq t \leq s$. At substage t, run the basic module on node $\alpha_{t,s}$. If instructed to end the stage, then do so. Otherwise, let $r = |\alpha_{t,s}|$. If:

$$l_r(s) > \max\{l_r(s') : s' < s \text{ and node } \alpha i \text{ was visited at stage } s'\}$$

then set $\alpha_{t+1,s} = \alpha_{t,s} i$ and injure all nodes extending $\alpha_{t,s} f$. Otherwise set $\alpha_{t+1,s} = \alpha_{t,s} f$.

Close stage. If A_{s+1} and B_{s+1} have not been defined, then set $A_{s+1} = A_s$ and $B_{s+1} = B_s$. For all nodes δ that have not been injured at stage s, if $e(\delta, s+1)$ has not yet been defined and $e(\delta, s)$ is defined, then set $e(\delta, s+1) = e(\delta, s)$. Define the functions $i(\delta, s+1)$, $h(\delta, s+1)$ and $n(\delta, s+1)$ similarly.

The basic module. Act according to the current status of α.

Status: unassigned. We take a larger than any value seen so far and $d = a + w_A(|\alpha|)$. We assign the work space $[a, d]$ to α. We change the status of α to *building*. We set $e(\alpha, s) = 0^{|\alpha|}$. We end the current task.

Status: building. We use the strategy of Lemma 9.4.1 with $R' = \{r < |\alpha| : \alpha(r) = i\}$ and $k' = |\alpha|$ on the interval assigned to α. We run a step of this strategy each time the node α is visited. If this strategy has finished then we let $\mathcal{I}'$ be the diagonalisation interval built and set $i(\alpha, s+1) = \mathcal{I}'$.

Now if $n(\alpha, s)$ is defined, then we let $\mathcal{I} = n(\alpha, s)$, and let β be the node that $\mathcal{I}$ was built for. If we take $k = |\beta|$ and $R = \{r < |\beta| : \beta(r) = i\}$, then the conditions of Lemma 9.4.2 are met so if for any $r \in R \cap R'$, $\mathcal{I}$ and $\mathcal{I}'$ are not r-linked at the current stage we adopt a strategy to ensure that l_r is bounded above for the least such r. If $\mathcal{I}$ and $\mathcal{I}'$ are $(R \cap R')$-linked then we change the status of α to *waiting*.

Each time this node is visited we injure all lower priority nodes and end the current task.

Status: waiting. Let $\langle b, c, s, k\rangle = i(\alpha, s)$. If $m_{|\alpha|} < c$, then finish the current substage. Otherwise, if $m_{|\alpha|} \geq c$ we need to diagonalise. We set $A_{s+1} = A_s \cup \{x(i(\alpha, s))\}$ and $B_{s+1} = B_s \cup \{y(i(\alpha, s))\}$. We set the status of α to *diagonalised*, injure all lower priority nodes and end the current task.

Status: diagonalised. Finish the current substage.

Status: incorrect response. A node α is only assigned this status if some diagonalisation has not occurred as expected for some $\gamma \succeq \alpha$. Let s' be the stage at which α was assigned this status. Since stage s', α cannot have been injured. Set $r = |\alpha| - 1$. Because of the way we choose α in task 1, it must be that $\alpha(r) = \gamma(r) = i$, $f(\gamma, s')(r) = 0$ and $e(\gamma, s')(r) = 1$. Let $\mathcal{I}^0 = i(\gamma, s')$.

As $e(\gamma, s')(r) = 1$ it must be that $n(\gamma, s')$ is defined. Let $\mathcal{I}^1 = n(\gamma, s')$ and let β be the node that $\mathcal{I}^1$ was assigned to. As $e(\gamma, s')(r) = 1$, it must be that $\beta(r) = i$ and $r \in g^A(\mathcal{I}^1, s)$.

As $f(\gamma, s')(r) = 0$ and $\gamma(r) = i$ it follows from the definition of f that $r \in g^B(\mathcal{I}^0, s') \subseteq g^B(\mathcal{I}^0, s)$.

Hence $r \in g^A(\mathcal{I}^1, s) \cap g^B(\mathcal{I}^0, s)$. Further $y(\mathcal{I}^0) \notin A_s$, $x(\mathcal{I}^1) \notin B_s$. If $\mathcal{I}^0 = \langle b_0, c_0, s_0, k_0\rangle$ and $\mathcal{I}^1 = \langle b_1, c_1, s_1, k_1\rangle$ then $A_s(c_0, b_1) = B_s(c_0, b_1)$ as A and B only ever differ on the diagonalisation blocks.

Now finally let s'' be the stage at which γ was last assigned the status *waiting* so $s'' < s' < s$. The strategy adopted in the building phase ensured that $\mathcal{I}^0$ and $\mathcal{I}^1$ were r-linked at stage s''. Since then there have been no changes to A or B on the interval (c_0, b_1). Hence $\mathcal{I}^0$ and $\mathcal{I}^1$ are still r-linked at stage s. Thus we can apply the strategy of Lemma 9.3.9 to ensure that l_r is bounded. Each time this node is visited we apply another step of this strategy, injure all lower priority nodes and end the current task.

9.7 Verification

Note that the construction is careful about when nodes are injured. We only injure a node β if a node α to the left of β is visited during task three, or if some initial segment α of β makes a change to A and B. The disadvantage of this approach is that it is possible to have a stage s at which a higher priority node α is assigned a work space that occurs after the work space assigned to a lower priority node β. Potentially, β could change A and B before the work space of α. However, this can only happen if α has just been assigned the *building* or *incorrect response* status. Any change to A and B before the work space will not affect the strategies α might use, as these are localised to the work space. If α is visited then β will be injured at this point. The reason we take this approach is that we know that if β is injured by α, then as α must have been visited, for all r such that $\alpha(r) = i$, l_r has increased. We will use this fact in the proof of Lemma 9.7.8.

Lemma 9.7.1. *During the construction, the tree is accessed infinitely often.*

Proof. If not then there is only a finite number of nodes whose status is ever changed from *unassigned*. As the actions of task one and two do not act on nodes whose status is *unassigned*, one of these tasks must run an infinite number of times on one node. Take the highest priority node α for which this is true. After tasks one and two have finished acting on all higher priority nodes, task one can only act once on α, and task two can only act $2^{|\alpha|}$ times. This is a contradiction and hence task three must run infinitely often. □

Let TP, the true path, be the leftmost path of the tree accessed infinitely often.

Lemma 9.7.2. *If $TP(r) = i$ then l_r is unbounded.*

Proof. Let $\gamma = TP \upharpoonright\upharpoonright r$. Let S be set of stages at which the construction visits γ. If $TP(r) = i$ then for all $s \in S$ we have that $l_r(s) > \max\{l_r(s') : s' \in S \text{ and } s' < s\}$. S is infinite so l_r is unbounded. □

Lemma 9.7.3. *If $\alpha \preceq TP$ then requirement $P_{|\alpha|}$ is met.*

Proof. There is some stage s after which no node to the left of α is visited during task three. After stage s no node to left of α will injure α, so α can only be injured by nodes γ that are initial segments of α. We argue inductively by assuming that all nodes γ where γ is an initial segment of α have finished acting (that is to say they have status *diagonalised* or they have status *waiting* on the last interval that is assigned to them and will not diagonalise). First we claim that α is only assigned a work space a finite number of times. This holds because α will only get a work space if it can improve its e-state. However, as there are only $2^{|\alpha|}$ possible e-states for α it can be assigned at most $2^{|\alpha|}$ work spaces.

If α is on the true path and $\alpha(r) = i$ then l_r is unbounded (Lemma 9.7.2). Hence at some stage the building component of the basic module is completed. Hence α must remain in the *waiting* or *diagonalised* states from some point onwards. If α reaches the *diagonalised* stage, then $m_{|\alpha|}$ is bounded so the node $|\alpha|$ is meet. Similarly, if α remains in the *waiting* stage. Note that once all higher priority nodes have finished acting, α cannot be given the status of *incorrect response*. If this did happen then l_r would be bounded for some r with $\alpha(r) = i$. Hence α would not be on the true path. □

This lemma also proves that the true path is infinite because if α stays in the *waiting* or *diagonalised* stage, then the next node is visited. Thus we have met all requirements P_p and so we can conclude that $A \not\leq_{cl} B$.

Now we also need to show that if l_r is unbounded then either $W_r \leq_{cl} B$, or $A \leq_{cl} W_r$. First let us show that in this case $TP(i) = r$.

Lemma 9.7.4. *If l_r is unbounded, then $TP(r) = i$.*

Proof. Let $\gamma = TP \upharpoonright r$. Let S be the set of stages when the construction visits γ. As l_r is unbounded, $\liminf l_r = \infty$ by Lemma 9.2.2. Hence given any x, for all but finitely many $s \in S$, $l_r(s) > x$. Hence there must be infinitely many $s \in S$ such that:

$$l_r(s) > \max\{l_r(s') : s' < s \text{ and } \gamma i \text{ was visited at stage } s'\}.$$

Thus the node γi is visited infinitely often and so $\gamma i \prec TP$ and $TP(r) = i$. □

Now take any r with l_r unbounded. Let $\alpha = TP \upharpoonright\upharpoonright r$ (so $\alpha(r) = i$). Let s_1 be a stage at which all nodes of priority greater than or equal to α have finished acting. Let σ be the maximum $r+1$-bit binary value such that $\{\gamma \in 2^{<\omega} : \exists s, f(\gamma, s) \upharpoonright |\alpha| = \sigma\}$ is infinite.

Lemma 9.7.5. *If for any $q \leq r$, $TP(q) = f$, then $\sigma(q) = 0$.*

Proof. If $\sigma(q) = 1$, then for infinitely many γ, there exists an s for which we have that $f(\gamma, s)(q) = 1$. This implies that for infinitely many γ with $\gamma(q) = i$, for some stage s' a diagonalisation interval $i(\gamma, s')$ is constructed. However, the construction of such a diagonalisation interval requires that l_q is at some stage at least the right end point of the interval. Hence l_q is unbounded and by Lemma 9.7.4, $TP(q) = i$. □

We will say that at stage s a node γ *will not diagonalise* if the status of γ will never change to *diagonalised* after stage s. Note that it is possible for the status of γ at stage s to be *diagonalised*. However, in this case, if the status of γ changes, it can never again be set to *diagonalised*.

Lemma 9.7.6. *There exists a stage s_2 such that for all $\gamma \in \{\pi \in 2^{<\omega} : \exists s, f(\pi, s) \upharpoonright |\alpha| > \sigma\}$, γ will not diagonalise.*

Proof. There is only a finite number of such nodes γ. The lemma holds for those nodes to the left of the true path as these are visited finitely often. The lemma holds for those nodes on the true path as these are only injured finitely often. Assume some such node γ is to the right of the true path. Then once $TP \upharpoonright |\gamma|$ reaches the *waiting* stage, and will no longer be injured, it follows that if γ does change its status to *diagonalised*, then the length of agreement of $m_{|\gamma|}$ has increased sufficiently so that $TP \upharpoonright |\gamma|$ will change its status to *diagonalised* as well. This will injure γ. Now γ will never get the opportunity to diagonalise again as $m_{|\gamma|}$ will not exceed any new diagonalisation interval assigned to γ. □

Let $s_\alpha = \max(s_1, s_2)$.

Lemma 9.7.7. *Given α, s_α and σ, for any x we can compute a stage s such that for all nodes β assigned diagonalisation intervals that overlap with $[0, x]$ and may act on that interval after stage s:*

(i). $\beta \succ \alpha$; and

(ii). Either:

(a) $e(\beta, s) \upharpoonright |\alpha| = \sigma$; or

(b) β will not diagonalise after stage s.

Proof. We know that α is on the true path so we can run the construction until a stage $s_3 > s_\alpha$ at which all nodes to the right of α are not allocated any interval overlapping with $[0, x]$. As $s_3 > s_\alpha > s_1$ all nodes of priority equal to or higher than α have finished acting. Thus the only nodes that can act on the interval $[0, x]$ are those that extend α. As $s_3 > s_\alpha > s_2$ any node γ with $e(\gamma, t) > \sigma$ for some t will not diagonalise.

Assume that there is some node β that extends α, such that $e(\beta, s) \upharpoonright |\alpha| < \sigma$ and β is assigned a diagonalisation interval that overlaps with $[0, x]$. Take β to be of highest priority of such nodes. Then we continue running the construction until some stage s_4 at which for some γ such that $|\gamma| > |\beta|$, and the status of γ at stage s_3 is *unassigned* and $f(\gamma, s_4) \upharpoonright |\alpha| = \sigma$. This must occur by our choice of σ. If γ has higher priority than β, then β has been injured (because γ must have changed its status to *building* at some point). Hence β must have been assigned an interval beyond x. Now if β has a higher priority than γ, and β still has not diagonalised, then β will be moved to the new interval. By repeating this process, we can continue until a stage s at which there are no such nodes β. □

Lemma 9.7.8. *If $TP(r) = i$, then $W_r \equiv_{cl} A$ or $W_r \equiv_{cl} B$.*

Proof. Let $r = \langle W, \Phi, \Psi \rangle$. Let $\alpha = TP \Vdash r$. To show that either $W \equiv_{cl} A$ or $W \equiv_{cl} B$, we will construct a Turing functional whose use is bounded by $x + r$. We need the following finite amount of information. Let $\alpha = TP \Vdash r$. Define σ and s_α as above.

If $\sigma(r) = 1$, then we will show that $W \equiv_{cl} A$. Given $W \Vdash x + r$, we run the construction until a stage s when the conditions of Lemma 9.7.7 are met on the interval $[0, x]$. Further we can assume that s is a stage at which $W_s \Vdash (x + r) = W \Vdash (x + r)$.

We claim that $A_s \Vdash x = A \Vdash x$. If this is not the case then it must be that some node β adds a number to $A \Vdash x$ without changing $W \Vdash (x + r)$. However each change to A occurs with a change to B. Any node that will act after the stage s on the interval $[0, x]$ must extend α. So if β makes a change to A and B, no other node will act during task three until α is visited again. Visiting α requires that the length of agreement l_r recovers. The length of agreement l_r can only recover with a change to W. Thus *any* change to A must have a corresponding change to W. Given this, the only possibility for $A_s \Vdash x \neq A \Vdash x$ is if a node β is assigned a diagonalisation interval $\mathcal{I}$ that overlaps with $[0, x]$, and $x(\mathcal{I}) \leq x < x + r < y(\mathcal{I}, r)$ and $x(\mathcal{I}) \in A \setminus A_s$ and $y(\mathcal{I}, r) \in W \setminus W_s$. Note that by Lemma 9.7.7, $e(\beta, s) \upharpoonright |\alpha| = \sigma$.

If this occurs, there must be some stage s' at which α is visited again such that $x(\mathcal{I}) \in A_{s'}$ and $y(\mathcal{I}, r) \in W_{s'}$. Further, for all $q \leq r$ where $\alpha(q) = i$ we must that l_q has recovered. Thus, provided β has not been injured, $|f(\beta, s')| \geq |\alpha|$. But in this case $f(\beta, s')(r) = 0$ (as $y(\mathcal{I}, r) \in W_{s'}$) but $e(\beta, s')(r) = \sigma(r) = 1$. Hence some initial segment γ of α will have its status changed to *incorrect response* and will act the next time it is visited. This is a contradiction as we assumed $s > s_1$ and all nodes with priority greater than or equal to α had finished acting by stage s_1.

Further, β cannot be injured between stage s and s'. This is because β can only be injured by a higher priority node during task three. But no node to the left of α can be visited and no initial segment of α will act. Thus the only higher priority nodes that can injure β are those that extend α. But these nodes cannot be visited until at least stage s', at which the lengths of agreement recover.

If $\sigma(r) = 0$, then we will show that $W \equiv_{cl} B$. Given $B \Vdash x + r$, run the construction until a stage s when the conditions of Lemma 9.7.7 are met on the interval $[0, x]$. Further we can assume that s is a stage at which $B_s \Vdash (x + r) = B \Vdash (x + r)$.

We claim that $W_s \Vdash x = W \Vdash x$. Again if β is a node that is assigned to an interval that overlaps with $[0, x]$, then $\beta \succ \alpha$ so $\beta(r) = i$. If $W_s \upharpoonright x \neq W \upharpoonright x$ then there must be some node β such that if $\mathcal{I} = i(\beta, s)$ then $x(\mathcal{I}, r) \leq x < x + r < y(\mathcal{I})$ and $x(\mathcal{I}, r) \in W \setminus W_s$ and $y(\mathcal{I}) \in B \setminus B_s$.

For this to occur there must be some stage s' at which α is visited again such that $x(\mathcal{I}, r) \in W_{s'}$ and $y(\mathcal{I}) \in B_{s'}$.

Again, β cannot be injured between stages s and s'. Hence it must be that $|f(\beta, s')| \geq |\alpha|$ but also, $f(\beta, s')(r) = 1 \neq \sigma(r) = 0$. So we know that $f(\beta, s') \upharpoonright |\alpha| \neq \sigma$. Further, $f(\beta, s') \upharpoonright |\alpha| \not\succ \sigma$ as such β will not diagonalise after stage $s > s_2$. Hence it must be that $f(\beta, s') \upharpoonright |\alpha| < \sigma = e(\beta, s')$. If this is true, there must be some $r' < |\alpha|$ such that $f(\beta, s')(r') = 0$ and $e(\beta, s')(r') = 1$. Thus some initial segment γ of α will have its status changed to *incorrect response*. Again this is contradiction. □

Proof of Theorem 9.1.4. By construction we have that $B \leq_{cl} A$. By Lemma 9.7.3 requirement P_p is met for all $p \in P$. Now suppose that for some requirement $r = \langle W, \Phi, \Psi \rangle$ we have that $\Phi^A = W$ and $\Psi^W = B$. In this case l_r is unbounded by Lemma 9.2.2 and so $TP(r) = i$ by Lemma 9.7.4. Thus by Lemma 9.7.8 either $W \equiv_{cl} A$ or $W \equiv_{cl} B$. So we have met requirement R_r for all $r \in \omega$. □

Chapter 10

Indifferent Sets for Comeager Classes

10.1 Overview

In this chapter we will apply concepts developed in the study of randomness to classical computability theory. Any Martin-Löf random sequence has a certain robustness; it can be changed at finitely many places and the resulting sequence is also Martin-Löf random. Figueira, Miller and Nies showed that this robustness could be extended in the following manner. Given a Martin-Löf random sequence A, there is some infinite set of locations I such that no matter how A is changed on the locations specified by I, the resulting sequence is also Martin-Löf random. They termed the set I an *indifferent set* for A with respect to Martin-Löf randomness [34].

Results about randomness can often be interpreted as results about forcing with closed sets of positive measure. A central notion of forcing used in computability theory is Cohen forcing. In Cohen forcing, comeager classes play an analogous role to closed sets of full measure. Our objective for this chapter is to investigate indifferent sets for comeager classes in Cantor space and particularly for the important comeager classes of 1-generic sets and weakly 1-generic sets.[1]

In order to study indifferent sets for some notion it is essential that the class of sets with this property has cardinality $2^{\aleph_0}$. This is certainly true for Martin-Löf randomness because any closed set of positive uniform measure has size $2^{\aleph_0}$. Meager and comeager sets were introduced by Baire and while they have a wider definition, we will restrict our attention to Cantor space. A subset of Cantor space is *comeager* if it contains the intersection of a countable family of open dense sets. A subset is *meager* if its complement is comeager. Any comeager set has cardinality $2^{\aleph_0}$.

In Section 10.3, we will consider universal indifferent sets for comeager

[1] We will define the notion of 1-genericity and other key concepts in Section 10.2.

classes in Cantor space. If $\mathcal{A}$ is a comeager class in Cantor space, then a universal indifferent set for $\mathcal{A}$ is a set I such that for all $A \in \mathcal{A}$, no matter how A is changed on the bits specified by I, the resulting sequence remains in $\mathcal{A}$. We will show in Theorem 10.3.3 that any comeager class in Cantor space contains a comeager class with a universal indifferent set. The classes of weakly n-generic sets are natural examples of comeager classes in computability theory. We will establish in Theorem 10.3.4, that for any n, the class of weakly n-generic sets has a universal indifferent set.

The class of all 1-generic sets is arguably the most important example of a comeager class in computability theory. Constructions based on Cohen forcing can often be adapted to construct a 1-generic set. Though they did not use this terminology, Jockusch and Posner showed that there exist 1-generic sets that have indifferent sets [43]. Figueira, Miller and Nies were aware that all 1-generic sets have indifferent sets, though this result was not published.[2] Fitzgerald established a few preliminary results on indifferent sets for 1-generic sets while he was a student at Victoria University of Wellington. These results were never published and we will present two of them here. We cannot investigate the question of indifferent sets for 1-generic sets using the approach of Section 10.3 because Miller has established that there is no universal indifferent set for the class of all 1-generic sets. We present his result in Theorem 10.3.6. Instead we will look for an indifferent set for a given 1-generic set. In Theorem 10.4.5, we establish a strong existence result for indifferent sets for 1-generic sets. All 1-generic sets G have an indifferent set I that is also 1-generic. Further, such a set I can be found below any set $A \in \overline{\mathrm{GL}_2}$ that bounds G. An easy corollary to Theorem 10.4.5 is that given any countable class of 1-generic sets, there is a set which is indifferent for all elements of this class. Fitzgerald established that any Δ^0_2 1-generic set has a co-c.e. indifferent set. We present his result in Theorem 10.4.8.

One use of indifferent sets is as coding locations. In Corollary 10.4.7, a corollary to Theorem 10.4.5, we use encoding with indifferent sets to show that if $X \in \overline{\mathrm{GL}_2}$, then for every 1-generic set G such that $X \geq_T G$, there is another 1-generic set $\hat{G}$ such that $X \equiv_T G \oplus \hat{G}$.

In Section 10.5 we examine the relationship between sparseness and indifferent sets. Fitzgerald showed that any indifferent set for a 1-generic set must be hyperimmune. In Theorem 10.5.3 we establish a spareness condition that is sufficient for a set to be the indifferent set for some 1-generic set.

An implication of Theorem 10.4.5 is that any 1-generic set has a GL_1 indif-

[2]This was observed by the anonymous referee of their paper.

ferent set. This contrasts strongly with indifferent sets for Martin-Löf randomness which must be complete [34]. The fact that indifferent sets for 1-generic sets can be computationally weak raises the possibility that a 1-generic set might be able to compute its own indifferent set. We investigate this question in Section 10.6 for Δ^0_2 1-generic sets. We establish that some but not all Δ^0_2 1-generic sets have this property. We consider which c.e. sets bound a 1-generic set with this property. We show that any c.e. set which is not of totally ω^ω-c.a. degree bounds such a 1-generic set. On the other hand no c.e. set of totally ω-c.a. degree bounds such a 1-generic set. These results are presented in Theorems 10.6.8 and 10.6.16 respectively.

In Section 10.7, we consider similar questions for weakly 1-generic sets. These results offer interesting contrasts with those for 1-generic sets. First in Theorem 10.7.1 we show that any hyperimmune set I computes a weakly 1-generic set G that I is an indifferent set for. This tells us that I is an indifferent set for some weakly 1-generic set if and only if I is hyperimmune (Corollary 10.7.2). In Theorem 10.7.4 we show that if a set G is weakly 1-generic and I is a set whose principal function escapes domination by any G-computable function then I is an indifferent set for G. Further, we show in Theorem 10.7.5, just as is the case for 1-generic sets, that if $A \in \overline{\mathrm{GL}_2}$ then A computes an indifferent set for any weakly 1-generic set it bounds. A difference to the case of 1-generic sets is provided in Theorem 10.7.6. In this theorem we show that any Δ^0_2 weakly 1-generic set computes a set it is indifferent to.

We conclude in Section 10.8 with some open questions.

10.2 Background and notation

Given A and $X \subseteq \omega$, we will write both $A_{[X]}$ and $A\Delta X$ to denote the symmetric difference of A and X, i.e. the set $(A \setminus X) \cup (X \setminus A)$. This is the set which differs from A at *precisely* the elements of X. If $\sigma, \tau \in 2^{<\omega}$, then by $\sigma\Delta\tau$ we mean $\sigma 0^\omega \Delta \tau 0^\omega$. Our central definition is the following.

Definition 10.2.1. Let $\mathcal{A} \subseteq 2^\omega$ and $I \subseteq \omega$.

(i). Take $A \in \mathcal{A}$. If for all $X \subseteq I$ we have that $A_{[X]} \in \mathcal{A}$ then we call I an *indifferent set for A with respect to $\mathcal{A}$.*

(ii). We call I a *universal indifferent set for $\mathcal{A}$* if I is an indifferent set for all $A \in \mathcal{A}$ with respect to $\mathcal{A}$.

We will be interested in the case in which $\mathcal{A}$ is comeager, i.e. contains the intersection of countably many open dense subsets. We denote the eth c.e. set

of finite strings by S_e.

Let $A \subseteq \omega$ and $S \subseteq 2^{<\omega}$. We will say that A *meets* S if $(\exists \sigma \prec A)(\sigma \in S)$, and that A *avoids* S if $(\exists \sigma \prec A)(\forall \tau \in S)\,(\sigma \not\preceq \tau)$. We will also say that a string σ meets S if for some $\tau \preceq \sigma$, $\tau \in S$, and σ avoids S if no string comparable with σ is in S. We call a set of strings S *dense* if for all $\sigma \in 2^{<\omega}$ there exists $\tau \in S$ such that $\tau \succeq \sigma$.

Definition 10.2.2. If $G \subseteq \omega$ meets or avoids all sets of finite strings computably enumerable in $\emptyset^{n-1}$, then G is *n-generic*. If G meets all dense sets computably enumerable in $\emptyset^{n-1}$, then G is *weakly n-generic*.

For an introduction to n-generic sets see survey papers by Jockusch and Kumabe [42, 49] and Kumabe's thesis [48]. Weakly n-generic sets were introduced by Kurtz [51]. In this chapter we will focus our study on 1-generic and weakly 1-generic sets.

A function f is *ω-c.a.* if $f(x) = \lim_s g(x,s)$ for some computable g and there is some computable h such that for all x,

$$|\{s : g(x,s+1) \neq g(x,s)\}| \leq h(x).$$

A Turing degree is a is *array non-computable*, or ANC if for any ω-c.a. function g, there is a function $f \leq_T \mathrm{a}$ such that f escapes domination by g. This class of degrees was introduced by Downey, Jockusch and Stob [32, 33]. A degree a is in $\mathrm{GL_n}$ if $\mathrm{a^n} = (\mathrm{a} \vee \emptyset')^{\mathrm{n-1}}$. We write $\overline{\mathrm{GL_n}}$ for the complement of $\mathrm{GL_n}$. A well-known fact is that the degree of any 1-generic set is $\mathrm{GL_1}$.

The use of $\overline{\mathrm{GL_2}}$ degrees and ANC degrees to perform Cohen forcing constructions was noted by Jockusch and Posner [43], and by Downey, Jockusch and Stob [33] respectively. Forcing using $\overline{\mathrm{GL_2}}$ degrees makes use of the following characterisation of Martin: $\mathrm{a} \in \overline{\mathrm{GL_2}}$ if and only if for any function $g \leq_T \mathrm{a} \vee \emptyset'$ there is a function $f \leq_T \mathrm{a}$ such that f escapes domination by g [62]. The following theorem of Cai and Shore extends these ideas and helps us understand the computational power required to undertake the forcing constructions used in this chapter [10]. An *A-computable notion of forcing $\mathcal{P}$*, is a set P of forcing conditions with a partial order $\leq_{\mathcal{P}}$ on P which contains a greatest element 1, such that $\mathcal{P}$ is computable in A. Let $\mathcal{C}$ be a sequence of dense subsets of $\mathcal{P}$. A sequence $\langle p_i \rangle$ is *$\mathcal{C}$-generic* if it meets each element of $\mathcal{C}$ and for all i, $p_i \geq_{\mathcal{P}} p_{i+1}$.

Theorem 10.2.3 (Cai, Shore)**.** *Suppose that $\mathcal{P}$ is an A-computable notion of forcing, that $\mathcal{C} = \langle D_n \rangle$ is a sequence of sets dense in $\mathcal{P}$, and that there is a function $d(x,y) = \Phi(A \oplus \emptyset'; x, y)$ witnessing their density, i.e. $\forall p \in \mathcal{P}\ \forall n (d(p,n) \leq_{\mathcal{P}} p \wedge d(p,n) \in D_n)$.*

(i). If $A \in \overline{\mathrm{GL}_2}$ then there is a $\mathcal{C}$-generic sequence computable in A.

(ii). If $A \in \mathrm{ANC}$ and the use from $\emptyset'$ in the computation of $\Phi(A \oplus \emptyset'; x, y)$ is bounded by a function computable in A, then there is also a $\mathcal{C}$-generic sequence computable in A.

A function computable in $\emptyset'$ with use bounded by a computable function is called wtt-reducible to $\emptyset'$. A c.e. degree a is of *totally ω-c.a.* degree if for all $f \leq_T \mathrm{a}$, f is ω-c.a. The class of totally ω-c.a. sets was introduced by Downey, Greenberg and Weber [27] and also studied by Barmpalias, Downey and Greenberg [5]. A forthcoming monograph of Downey and Greenberg generalises this concept [26]. The terminology below follows that monograph.

Let $\mathcal{R} = (R, \leq_R)$ be a computable well-ordering of a computable set R. An *$\mathcal{R}$-computable approximation* of a function f is a computable approximation $\langle f_s \rangle_{s<\omega}$ of f, equipped with a uniformly computable sequence $\langle o_s \rangle_{s<\omega}$ of functions from ω to R such that for all x and s:

- $o_{s+1}(x) \leq_{\mathcal{R}} o_s(x)$.
- If $f_{s+1}(x) \neq f_s(x)$, then $o_{s+1}(x) <_{\mathcal{R}} o_s(x)$.

The sequence $\langle o_s \rangle_{s<\omega}$, together with the well-foundedness of $\mathcal{R}$, witnesses the fact that the approximation $\langle f_s \rangle_{s<\omega}$ indeed reaches a limit.

Definition 10.2.4. A function $f : \omega \to \omega$ is *$\mathcal{R}$-computably approximable* (or *$\mathcal{R}$-c.a.*) if it has an $\mathcal{R}$-computable approximation.

It is possible to use this definition to establish a hierarchy in the Turing degrees by restricting ourselves to certain well-orderings. Every ordinal α has a unique expression as the sum

$$\omega^{\alpha_1} n_1 + \omega^{\alpha_2} n_2 + \cdots + \omega^{\alpha_k} n_k$$

where $n_i < \omega$ are nonzero and $\alpha_1 > \alpha_2 > \cdots > \alpha_k$ are ordinals. This is called the *Cantor normal form* of α. Further,

$$\varepsilon_0 = \sup \left\{ \omega, \omega^{\omega}, \omega^{\omega^{\omega}}, \omega^{\omega^{\omega^{\omega}}}, \dots \right\}$$

is the least ordinal γ such that $\omega^\gamma = \gamma$, so for all $\alpha < \varepsilon_0$, every ordinal appearing in the Cantor normal form of α is strictly smaller than α.

Let $\mathcal{R} = (R, <_{\mathcal{R}})$ be a computable well-ordering, and let $|\cdot| \colon R \to \mathrm{otp}(\mathcal{R})$ be the unique isomorphism between $\mathcal{R}$ and its order-type. The pullback to $\mathcal{R}$ of the Cantor normal form function is the function $\mathtt{nf}_{\mathcal{R}}$ whose domain is R and is defined by letting

$$\mathtt{nf}_{\mathcal{R}}(z) = \langle (z_1, n_1), (z_2, n_2), \dots, (z_k, n_k) \rangle$$

where $n_i < \omega$ are nonzero, $z_i \in R$, $z_1 >_{\mathcal{R}} z_2 >_{\mathcal{R}} \cdots >_{\mathcal{R}} z_k$, and

$$|z| = \omega^{|z_1|}n_1 + \omega^{|z_2|}n_2 + \cdots + \omega^{|z_k|}n_k.$$

Definition 10.2.5. A computable well-ordering $\mathcal{R}$ is *canonical* if its associated Cantor normal form function $\mathtt{nf}_{\mathcal{R}}$ is also computable.

Downey and Greenberg have established that for every ordinal $\alpha \leq \varepsilon_0$ there is a canonical well-ordering of order-type α and further that any two canonical well-orderings of order-type α are computably isomorphic. Hence we can define a function f as being *α-c.a.* if it is $\mathcal{R}$-c.a. for some canonical well-ordering of order-type α.

Definition 10.2.6. If $\alpha \leq \varepsilon_0$, then a Turing degree a is *totally α-c.a.* if every function $f \in \mathrm{a}$ is α-c.a.

It is not difficult to show that a is totally α-c.a. if and only if every function $f \leq_T \mathrm{a}$ is α-c.a. The following theorem establishes that the α-c.a. degrees do indeed form a hierarchy.

Theorem 10.2.7 (Downey, Greenberg [26])**.** *Let $\alpha \leq \varepsilon_0$. There is a totally α-c.a. degree which is not totally γ-c.a. for any $\gamma < \alpha$ if and only if α is a power of ω. If α is a power of ω then in fact there is a c.e. degree which is totally α-c.a. but not totally γ-c.a. for any $\gamma < \alpha$.*

10.3 Universal indifferent sets

We will begin by looking at indifferent sets for comeager subsets of Cantor space. Not all comeager subsets of Cantor space have a universal indifferent set. A trivial example is the set comprised of all 1-generic sets and the empty set. If this comeager set had a universal indifferent set I, then we could add the least element of I to the empty set and obtain a finite 1-generic set.

However, we will show that any comeager subset of Cantor space contains a comeager subset with a universal indifferent set. In order to prove this, we will establish a property of a countable class of dense sets of strings $\mathcal{S}$, that will ensure that the class of sets that meet all elements of $\mathcal{S}$ has a universal indifferent set.

Let S be a dense set of strings. A typical approach to constructing a sequence A that meets S is to find some string $\sigma \in S$ and then fix σ to be an initial segment of A. However, as the following lemma shows, the segment that we fix does not need to be an initial segment of A.

Lemma 10.3.1. *If S is a dense set of strings, then for all $n \in \omega$, there some $\tau_n \in 2^{<\omega}$ such that for all $X \in 2^\omega$, if $(X \upharpoonright n)\tau_n \prec X$ then X meets S.*

Proof. Fix some enumeration of S. To find τ_n, let $\sigma_1, \ldots, \sigma_k$ be a list of all finite strings of length n. Define ρ_1 such that $\sigma_1\rho_1 \in S$ and ρ_1 is the first string observed with this property. Once ρ_i is defined, if $i < k$, define ρ_{i+1} such that $\sigma_{i+1}\rho_1\rho_2 \ldots \rho_i\rho_{i+1} \in S$ and ρ_{i+1} is the first string observed with this property. Define $\tau_n = \rho_1 \ldots \rho_k$.

Take any $X \subseteq \omega$ such that $(X \upharpoonright n)\tau_n \prec X$. Fix the i such $(X \upharpoonright n) = \sigma_i$. Thus $\sigma_i\rho_1 \ldots \rho_i \prec X$ and hence $X \in S$. □

Given a dense set of strings S, let τ_i witness the truth of Lemma 10.3.1 for each $i \in \omega$. We can inductively define a pair of functions $t : \omega \to 2^{<\omega}$ and $l : \omega \to \omega$ by

$$l(0) = 0, \quad t(0) = \tau_0, \quad l(n+1) = l(n) + |t(n)|, \quad t(n+1) = \tau_{l(n+1)}.$$

If for any n we have that $(X \upharpoonright l(n))t(n) \prec X$, then X meets S. This section will make use of pairs of function of this type so we will introduce the following term. We call a pair of functions $(t, l) \in (2^{<\omega})^\omega \times \omega^\omega$ *suitable* for S if for all n:

(i). $l(n+1) = l(n) + |t(n)|$.

(ii). For all $X \in 2^\omega$ if $(X \upharpoonright l(n))t(n) \prec X$, then X meets S.

Let $\mathcal{S}$ be a class of dense sets of strings. We say that $\mathcal{S}$ has property $(\star)$ if for all $S \in \mathcal{S}$ there is a pair of functions (t, l) suitable for S such that for all n, the following set is also in $\mathcal{S}$:

$$\{\sigma\tau\rho : \exists m > n[|\sigma| = l(2m) \wedge \tau = t(2m) \wedge \rho = t(2m+1)]\}. \tag{10.3.1}$$

Lemma 10.3.2. *If $\mathcal{S} = \{S_i\}_{i\in\omega}$ is a countable sequence of dense sets of strings with property $(\star)$, then there is a universal indifferent set for $\mathcal{A} = \{X \subseteq \omega : (\forall i)X$ meets $S_i\}$.*

Proof. For each $i \in \omega$, we can take (t_i, l_i) to be a pair of suitable functions for S_i that witness property $(\star)$. Let I be an infinite set such that:

$$(\forall i)(\forall^\infty n)\ |[l_i(2n), l_i(2n+2) - 1] \cap I| \leq 1.$$

We claim that I is a universal indifferent set for $\mathcal{A}$. Let $A \in \mathcal{A}$ and $X \subseteq I$. Consider any dense set S_i. Now because of property $(\star)$ there are infinitely many n such that $(X \upharpoonright l_i(2n))t_i(2n)t_i(2n+1) \prec A$. Now for almost all of these n we have that $|I \cap [l_i(2n), l_i(2n+2) - 1]| \leq 1$. For such n either

(i). $A_{[X]} \succ (A_{[X]} \upharpoonright l_i(2n))t_i(2n)$; or

(ii). $A_{[X]} \succ (A_{[X]} \upharpoonright l_i(2n+1))t_i(2n+1)$.

In either case we have that $A_{[X]}$ meets S_i. Thus $A_{[X]} \in \mathcal{A}$. □

Theorem 10.3.3. *Let $\mathcal{A} \subseteq 2^\omega$ be comeager. There exists $\mathcal{B} \subseteq \mathcal{A}$ and $I \subseteq \omega$ such that $\mathcal{B}$ is comeager and I is a universal indifferent set for $\mathcal{B}$.*

Proof. Let $\mathcal{S} = \{S_i\}_{i\in\omega}$ be a countable sequence of dense sets of strings such that $\{X \subseteq \omega : (\forall i)X \text{ meets } S_i\} \subseteq \mathcal{A}$.

We extend $\mathcal{S}$ as follows. Let $\mathcal{S}_0 = \mathcal{S}$. We build $\mathcal{S}_{n+1}$ from $\mathcal{S}_n$ as follows. For each $S \in \mathcal{S}_n$ we choose a pair of functions (t, l) suitable for S and add S along with the countably many dense sets of strings defined by (10.3.1) to $\mathcal{S}_{n+1}$. Then $\mathcal{S}_\omega = \bigcup_n \mathcal{S}_n$ has property $(\star)$ and is a countable set of dense sets of strings. Thus there is a universal indifferent set for $\mathcal{B} = \{X \subseteq \omega : (\forall S \in \mathcal{S}_\omega)X$ meets $S\}$. □

If a set X can enumerate an open dense subset S, i.e. $S = W_e(X)$ for some e, then using the procedure of Lemma 10.3.1, X can uniformly in e compute a pair of functions (t, l) that are suitable for S.

Theorem 10.3.4. *Fix $n \geq 1$. If $B \geq_T \emptyset^{n-1}$ and $B' \geq_T \emptyset^{n+2}$ then B computes a universal indifferent set for the class of all weakly n-generic sets.*

Proof. Fix $n \geq 1$ and let $A = \emptyset^{n-1}$, let $\{S_i\}_{i\in\omega}$ be an A computable enumeration of all c.e. sets of strings enumerable in A. Using A as an oracle, we can in a uniform manner define (t_i, l_i) such that if S_i is a dense set of strings, then

(i). (t_i, l_i) is suitable for S_i.

(ii). All sets defined from (t_i, l_i) as in (10.3.1) are also enumerable in A.

Hence the collection of open dense sets enumerable in A has property $(\star)$. The set $D = \{i : S_i(A) \text{ is dense }\}$ is $\Delta^0_2(B)$ because $B' \geq_T A''$. Let D_s be a B-computable approximation to D.

From B we define $x_0 = 0$. Once x_s is defined, we find some $t \geq s$ such that for all $e \leq s$, one of the following is true:

(i). $D_t(e) = 0$; or

(ii). For some n_i, we have $l_i(2n_i)[t] \downarrow > x_s$.

We let x_{s+1} be the least element of ω greater than both x_s and $\max\{l_i(2n_i) : l_i(2n_i)[t] \downarrow$ and $e \leq s\}$. We let $I = \{x_n : n \in \omega\}$. Now because our approximation to D must converge, I is well defined and has the desired property. □

Corollary 10.3.5. *If $A, B \in 2^\omega$ with $A \leq_T B$ and $A'' \leq_T B'$ then B computes a universal indifferent set for the class of open dense sets enumerable in A.*

Proof. The restriction of A to $\emptyset^n$ for some n is not necessary. □

Theorem 10.3.3 tells us that the class of 1-generic sets contains a comeager subset with a universal indifferent set. Theorem 10.3.4 provides a specific example, namely the class of all weakly 2-generic sets. However, is there a universal indifferent set for class of 1-generic sets itself? Miller, in unpublished work, has proved that there is not.

Theorem 10.3.6 (Miller)**.** *Let $I \subseteq \omega$ be infinite. There are sets $G, A \subseteq \omega$ such that G is 1-generic, A is not 1-generic, and $G \Delta A \subseteq I$. In other words, I is not universally indifferent for the class of 1-generic sets.*

Proof. We define the characteristic function of G in stages. At stage 0 we pick some element $i_0 \in I$ greater than 0. We define $\sigma_0 = 0^n 10 \emptyset'(0)$ where n is chosen so that $|0^n 1| = i_0$ i.e. i_0 is the location of the 0 in σ_0 before the coding location of $\emptyset'(0)$.

At stage $s+1$, we define σ_{s+1} extending σ_s as follows. If σ_s avoids S_s, then define $\tau_s = \lambda$. If σ_s does not avoid S_s then let τ_s be the first string enumerated into S_s that extends σ_s. We take $i_{s+1} \in I$ such that $i_{s+1} > |\tau_s|$ and we now define $\sigma_{s+1} = \tau_s 0^n 10 \emptyset'(s+1)$ where n is chosen so that $|\tau_s 0^n 1| = i_{s+1}$. Let $G = \bigcup_n \sigma_n$. The construction ensures that the set G is 1-generic. Define $X = \{i_n : \sigma_n$ avoids $S_n\}$. Define $A = G_{[X]}$. From A and σ_n, we can determine whether or not σ_n meets or avoids S_n. Thus we can determine σ_{n+1} and so we can determine $\emptyset'(n+1)$. This shows that $A \geq_T \emptyset'$ and consequently A cannot be 1-generic. □

10.4 Indifference and 1-genericity

We know that there is no universal indifferent set for the class of all 1-generic sets. However we will establish that all 1-generic sets have an indifferent set with respect to the class of all 1-generic sets. It is known that there exists 1-generic sets with indifferent sets.

Theorem 10.4.1 (Jockusch and Posner [43])**.** *If $A \in \overline{\mathrm{GL}_2}$, then there exists G, I Turing below A such that G is 1-generic and I is an indifferent set for G.*

The terminology used by Jockusch and Posner is different. They constructed a function $f : \omega \to \{0, 1, 2\}$ such that f is 1-generic and any characteristic function obtained from f by replacing 2's with 0's and 1's is also 1-generic. Given such an f, let $G = \{x : f(x) = 1\}$ and $I = \{x : f(x) = 2\}$.

Clearly I is an indifferent set for G. This result can be strengthened by the work of Cai and Shore.

Theorem 10.4.2 (Jockusch and Posner; Cai and Shore [10, 43])**.** *If $A \in \mathrm{ANC}$ then there exists G, I Turing below A such that G is 1-generic and I is an indifferent set for G.*

We will strengthen Jockusch and Posner's result in a different direction. Theorem 10.4.5 establishes that any $X \in \overline{\mathrm{GL}_2}$ computes an indifferent set for *any* 1-generic set it bounds. We make use of the following well-known lemma.

Lemma 10.4.3. *If G is a 1-generic set and $X \subseteq \omega$ is finite, then $G_{[X]}$ is 1-generic.*

Given a 1-generic set G and $e, n \in \omega$ we can find some point m such that no matter how we change G on the first n bits, the resulting set meets or avoids S_e after m bits.

Lemma 10.4.4. *Let G be a 1-generic set. There is a function $f_G : \omega^2 \to \omega$ with $f_G \leq_T G \oplus \emptyset'$ such that for all $n, e \in \omega$, for all $X \subseteq \{0, 1, \ldots, n-1\}$, we have that $G_{[X]} \upharpoonright f_G(n, e)$ meets or avoids S_e.*

Proof. Take any n and e and let $X_1, X_2, \ldots, X_{2^n}$ be a list of all subsets of the set $\{0, 1, \ldots, n-1\}$. For all i, $1 \leq i \leq 2^n$, we have that $G_{[X_i]}$ is 1-generic as X_i is finite. Hence there is some m_i such that $G_{[X_i]} \upharpoonright m_i$ meets or avoids S_e. This m_i is computable in $G \oplus \emptyset'$ because we can query whether successive initial segments of $G_{[X_i]}$ meet or avoid S_e until we find one that does. Define $f_G(n, e) = \max\{m_i : 1 \leq i \leq 2^n\}$. □

We are now ready to give our basic existence result for indifferent sets for 1-generic sets.

Theorem 10.4.5. *Given $A \geq_T G$ where $A \in \overline{\mathrm{GL}_2}$ and G is a 1-generic set, there exists $I \leq_T A$ such that I is an indifferent set for G with respect to 1-genericity and I is 1-generic.*

Proof. Let G be 1-generic. We can make I both 1-generic and an indifferent set for G by satisfying the following requirements for all $e \in \omega$.

Q_e: I meets or avoids S_e.

R_e: For all $X \subseteq I$ we have that $G_{[X]}$ meets or avoids S_e.

These requirements can be met by constructing an I that meets the following dense sets:

$$C_e = \{\sigma \in 2^{<\omega} : \sigma \text{ meets or avoids } S_e\}, \text{ and}$$

$$D_e = \{\sigma \in 2^{<\omega} : \forall X \subseteq \sigma, G_{[X]} \upharpoonright |\sigma| \text{ meets or avoids } S_e\}.$$

We say $X \subseteq \sigma$ if $X \subseteq \{i : i < |\sigma| \wedge \sigma(i) = 1\}$.

In this proof the notion of forcing that we use is just Cohen forcing so $P = 2^{<\omega}$ and $\sigma \leq_{\mathcal{P}} \tau$ if $\sigma \succeq \tau$. There is a function computable in $\emptyset'$ that uniformly witnesses the density of the sequence of sets $\langle C_n \rangle$: let $c(\sigma, e)$ be the first extension of σ to enter S_e if such an extension exists, or σ otherwise. There is also a function computable in $G \oplus \emptyset'$ that witnesses the density of the sequence of sets $\langle D_n \rangle$. We define $d(\sigma, e) = \sigma 0^{f_G(|\sigma|,e)-|\sigma|}$.

If $I \succ d(\sigma, e)$ and $X \subseteq I$, then $G_{[X]} \upharpoonright |d(\sigma, e)|$ can only differ from G on the first $|\sigma|$ bits. As $f_G(|\sigma|, e) = |d(\sigma, e)|$ by Lemma 10.4.4, $G_{[X]} \upharpoonright |d(\sigma, e)|$ meets or avoids S_e.

By applying Theorem 10.2.3, there is an A-computable $\leq_{\mathcal{P}}$ decreasing sequence $\langle \sigma_i \rangle$ that meets all sets in the sequences $\langle C_e \rangle$ and $\langle D_e \rangle$. Hence taking $I = \lim_i \sigma_i$, we have that I is 1-generic and I is an indifferent set for G. □

Corollary 10.4.6. *If $\{G_i\}_{i\in\omega}$ is a countable family of 1-generic sets and $A \geq_T \oplus_{i\in\omega} G_i$ with $A \in \overline{\mathrm{GL}_2}$, then there is an $I \leq_T A$ such that for all $i \in \omega$, I is an indifferent set for G_i.*

Proof. We simply replace the sequence of sets $\langle D_e \rangle$ with

$$D_{e,i} = \{\sigma \in 2^{<\omega} : \forall X \subseteq \sigma, G_{i,[X]} \upharpoonright |\sigma| \text{ meets or avoids } S_e\}.$$

Now because the sequence $\{G_i\}_{i\in\omega}$ is uniformly computable in A there is a function computable in $A \oplus \emptyset'$ that witnesses the density of this sequence. □

One possible use of indifferent sets for 1-generic sets is as coding locations. We can take a 1-generic G, and then form another 1-generic $\hat{G}$ by changing G on some bits of an indifferent set. We can recover these changes from the join of G and $\hat{G}$. The following theorem is an application of this idea.

Corollary 10.4.7. *Given $A \geq_T G$ where $A \in \overline{\mathrm{GL}_2}$ and G is a 1-generic set, there exists a 1-generic $\hat{G}$ such that $G \oplus \hat{G} \equiv_T A$.*

Proof. G has an A-computable indifferent set I that is also 1-generic. As I is 1-generic there are infinitely many even numbers in I and infinitely many odd numbers in I. Define the following A computable subset of I. If $0 \in A$, then let x_0 be the first even element of I, otherwise let x_0 be the first odd element of I. We inductively define x_{i+1} to be the first even element of I greater than x_i if $i + 1 \in A$ and the first odd element of I greater than x_i otherwise. Let $X = \{x_i : i \in \omega\}$. As $G, X \leq_T A$ we have that $G_{[X]} \leq_T A$. Further, $A \leq_T G \oplus G_{[X]}$ because $x \in A$ if and only if the ith position where G and $G_{[X]}$ differ is even. Thus $A \equiv_T G \oplus G_{[X]}$. □

In our final result for this section, we will show that any Δ^0_2 1-generic set has a co-c.e. indifferent set. This was originally shown by Fitzgerald using a full approximation argument. We give an alternative proof.

Theorem 10.4.8 (Fitzgerald)**.** *If G is a Δ^0_2 1-generic set, then there exists a co-c.e. set I such that I is an indifferent set for G.*

Proof. Let $G \in \Delta^0_2$ be a 1-generic set. We will show that there is a function $f \leq_T \emptyset'$ such that for any set I such that p_I, the principal function of I, majorizes f we have that I is an indifferent set for G. Because for any Δ^0_2 function f, there is a co-c.e. set whose principal function majorizes f we are done.

Define $f(e)$ as follows. Set $n_{e,0} = 0$ and then $n_{e,i+1} = f_G(n_{e,i}, e) + 1$. Now set $f(e) = n_{e,e+1}$. The function f is computable in $\emptyset'$ because $f_G \leq_T G \oplus \emptyset' = \emptyset'$. Let I be a co-c.e. set such that for all x, $p_I(x) \geq f(x)$. Take any e. Now consider the pairwise disjoint sets $[n_{e,i}, n_{e,i+1} - 1]$ for i such that $0 \leq i < e + 1$. Because $n_{e,e+1} = f(e) \leq p_I(e)$, There must be some i such that $[n_{e,i}, n_{e,i+1} - 1] \cap I = \emptyset$. Now because $n_{e,i+1} - 1 = f_G(n_{e,i})$, for any $X \subseteq I$, $G_{[X]} \upharpoonright f_G(n_{e,i}, e)$ only differs from G on the first $n_{e,i}$ many bits. Thus $G_{[X]}$ meets or avoids S_e and so I is a co-c.e. indifferent set for G. □

10.5 Sparsity of indifferent sets

In the previous section, we used a forcing construction to build an indifferent set. That construction succeeded by placing large intervals of zeros into the indifferent set. This indicates that indifferent sets for 1-generic sets may need to be sparse. The following theorem of Fitzgerald confirms this intuition by showing that they must be hyperimmune.

Theorem 10.5.1 (Fitzgerald)**.** *If I is an indifferent set for some 1-generic set G, then I is hyperimmune.*

Proof. Let G be 1-generic and $X \subseteq \omega$ be a set that is not hyperimmune Let f be a computable and strictly increasing function such that for all j, $[f(j), f(j + 1) - 1] \cap X \neq \emptyset$. Then there exists some $\hat{X} \subseteq X$ such that for all j there is a unique $n \in [f(j), f(j+1)-1] \cap \hat{X}$ if and only if $|[f(j), f(j+1)-1] \cap G|$ is even.

Consider the set $S \subseteq 2^{<\omega}$, defined by $\sigma\tau \in S$ if for some j, $|\sigma| = f(j)$, $|\tau| = f(j+1)-1-f(j)$ and τ has an even number of bits. The set S is a c.e. dense set of strings. Now $G_{[\hat{X}]}$ cannot meet S because for all j, $|[f(j), f(j+1)-1] \cap G_{[\hat{X}]}|$ must be odd. Thus $G_{[\hat{X}]}$ is not 1-generic and so X is not an indifferent set for G. □

Thus an indifferent set needs to be sparse. In fact if a set is sufficiently sparse, then it must be the indifferent set for some 1-generic set. The following lemma is a version of Lemma 10.3.1 for sets that are not necessarily dense.

Lemma 10.5.2. *There exists an ω-c.a. function $g : \omega^2 \to 2^{<\omega}$ such that for all $n, e \in \omega$, for all $X \subseteq \omega$, if $(X \upharpoonright n)g(n, e) \prec X$ then X meets or avoids S_e.*

Proof. Fix some enumeration of the sets S_e such that at most one string enters S_e at any stage s. Given n and e we use the following process to determine $g(n, e)$. We let $\tau_0 = \lambda$. Now we inductively define τ_{s+1} as follows. If we have some string $\sigma\rho$ enumerated into S_e at stage s with $|\sigma| = n$, $\rho \succeq \tau_s$ and such that σ has not been satisfied, then we define $\tau_{s+1} = \rho$ and we regard σ as being satisfied. Otherwise, we define $\tau_{s+1} = \tau_s$. We let $g(n) = \lim_s \tau_s$. As τ_s can change at most 2^n times we have that g is ω-c.a.

Assume that for some $n, e \in \omega$ and $X \subseteq \omega$ we have that $(X \upharpoonright n)g(n, e) \prec X$. If during the construction of $g(n, e)$ the string $X \upharpoonright n$ was satisfied then for some $\tau \preceq g(n, e)$ we have that $(X \upharpoonright n)\tau \in S_e$ and so X meets S_e. If not, then there is no $\tau \succeq g(n, e)$ such that $(X \upharpoonright n)\tau \in S_e$. Thus $(X \upharpoonright n)g(n)$ avoids S_e. □

Theorem 10.5.3. *Take I to be an infinite subset of ω. There is an ω-c.a. function f, such that if*

$$\exists^\infty n\, [n, f(n)] \cap I = \emptyset,$$

then I is an indifferent set for some 1-generic G.

Proof. Take $f(n) = n + \max\{|g(n, e)| : e \leq n\}$ where g is the function defined in Lemma 10.5.2. As g is ω-c.a., so is f. Now given I such that $\exists^\infty n\, [n, f(n)] \cap I = \emptyset$ we can take a sequence $n_0, n_1, \ldots$ such that for all i, $[n_i, f(n_i)] \cap I = \emptyset$, and $f(n_i) < n_{i+1}$. We define G as $\lim_i \sigma_i$ where $\sigma_0 = 0^{n_0}$ and with inductive assumption that $|\sigma_i| = n_i$, we define $\sigma_{i+1} = \sigma_i g(n_i, i)0^{k_i}$ where $k_i = n_{i+1} - |\sigma_i g(n_i, i)|$ (k_i ensures that $|\sigma_{i+1}| = n_{i+1}$ and $k_i > 0$ because $n_{i+1} > f(n_i) \geq n_i + g(n_i, i)$).

Now let $X \subseteq I$ and take any $e \in \omega$. We have that $(G_{[X]} \upharpoonright n_e)g(n_e, e) \prec G_{[X]}$ because $I \cap [n_e, f(n_e)] = \emptyset$ and so by Lemma 10.5.2, $G_{[X]}$ meets or avoids S_e. □

10.6 1-generic sets that compute their own indifferent sets

We know that every 1-generic set has as indifferent set which is GL_1. This indicates that it may be possible for a 1-generic set to compute an indifferent set for itself. Such a 1-generic set would have the following interesting property. For all $A \geq_T G$ there exists a 1-generic $\hat{G}$ such that A is the join of G and $\hat{G}$.

To show this, let G compute an indifferent set for itself with principal function p_I. If $A >_T G$, then let $X = \{p_I(x) : x \in A\}$, and so $A \equiv_T G \oplus G_{[X]}$. In this section we will show that such 1-generic sets exist. In fact we will show that any c.e. set that is not of totally ω^ω-c.a. degree computes a 1-generic set with this property. First, by appealing to known results, we will show that not all 1-generic sets have this property.

Proposition 10.6.1. *Any 1-generic set bounded by a set with a strong minimal cover does not compute an indifferent set for itself.*

Proof. If $A >_T B \geq_T G \geq_T I$, where G is 1-generic and I is an indifferent set for G, then by the discussion above, there is a 1-generic $\hat{G}$ such that $A = G \oplus \hat{G}$. If $\hat{G} <_T A$, then as $\hat{G} \not\leq_T B$ we have that A is not a strong minimal cover of B. If $\hat{G} \equiv_T A$, then A splits into two incomparable degrees one of which cannot be below B and so again A cannot be a strong minimal cover of B. □

Corollary 10.6.2. *Any 1-generic set bounded by an array computable c.e. set does not compute an indifferent set for itself.*

Proof. This corollary follows from Ishmukhametov's theorem that any array computable c.e. set has a strong minimal cover [41]. □

Corollary 10.6.3. *There exists a Δ^0_2 1-generic set G that fails to compute an indifferent set for itself.*

Proof. Any c.e. set bounds a 1-generic set and there exist array computable c.e. sets. □

We will now work towards proving that every c.e. set that is not of totally ω^ω-c.a. degree computes a 1-generic set which computes an indifferent set for itself. We first start by proving that $\emptyset'$ computes such a 1-generic set. Following this, we will examine the proof and turn it into a permissions argument.

Theorem 10.6.4. *There exists a Δ^0_2 1-generic set G that computes an indifferent set for itself.*

Proof. For this construction we will build a set G and a reduction Γ such that $\Gamma(G)$ is the principal function of I, an indifferent set for G. We have the following requirements:

I_e: $\Gamma(G; e) \downarrow$ and if $e > 0$, $\Gamma(G; e) > \Gamma(G; e-1)$.

R_e: $\exists w$ such that $\forall X \subseteq \text{rng}\Gamma(G)$, $G_{[X]} \upharpoonright w$ meets or avoids S_e.

If we meet all these requirements, then G is 1-generic and $\text{rng}\Gamma(G)$ is an infinite indifferent set for G. Our requirements will be prioritised as follows:

$R_0 > I_0 > R_1 > I_1 > \ldots$. The construction of G is a finite injury argument. At stage 0 we set $G_0 = 0^\omega$. At stage $s+1$ we find the highest priority requirement that needs attention and act on it.

Requirement I_e needs attention at stage s if $\Gamma(G, e)[s] \uparrow$. Our action is to choose a new large number n such that $G_s(n) = 0$ (this will always be possible because there will only be finitely many places where G_s is 1). We then set $\Gamma(G_s; e) = n$ with use n, define $G_{s+1} = G_s$ and restrain $G_{s+1} \upharpoonright (n+1)$ with priority e.

Each requirement R_e will have a restraint, r, imposed on it by higher priority requirements. There will also be a value $w(e, s) \geq r$ (this is our candidate for the witness w). We start with $w(e, 0) = 0$ for all e. A requirement R_e needs attention at stage s if there exists $\sigma \in S_{e,s}$ with $|\sigma| > w(e, s)$, such that:

(i). $(\sigma \upharpoonright w(e, s))\Delta(G_s \upharpoonright w(e, s)) \subseteq \text{rng}\Gamma(G)[s]$.

(ii). No initial segment of $\sigma \upharpoonright w(e, s)$ is in $S_{e,s}$.

The reason we need to pay attention to R_e at this stage is because $\sigma \upharpoonright w(e, s)$ differs from G_s only on elements of $\text{rng}\Gamma(G)[s]$ but $\sigma \upharpoonright w(e, s)$ does not meet or avoid S_e. This means that $w(e, s)$ is not a correct witness for requirement R_e.

Our action is like that of a greedy algorithm. We change G_s to look like σ anywhere we can while maintaining our constraints. Specifically we set:

$$G_{s+1}(x) = \begin{cases} G_s(x) & x \leq r \\ \sigma(x) & r < x < |\sigma| \\ 0 & |\sigma| \leq x. \end{cases}$$

We set $w(e, s+1) = |\sigma|$, and we restrain $G_{s+1} \upharpoonright w(e, s+1)$ with priority e. For all $e' < e$ we set $w(e', s+1) = w(e', s)$. For all $e' > e$ we set $w(e', s+1) = w(e, s)$. This ends the construction.

Verification. The key step to verifying this construction is to show that each requirement needs attention finitely often. Take any requirement, and assume that at stage s_0, all higher priority requirements have stopped acting and a final restraint r is placed on this requirement. First consider a requirement of the form I_e. If $\Gamma(G; e)[s_0]$ is not defined, then it will be defined at the next stage and its use preserved.

Now consider a requirement of the form R_e. Let $n = 2^r$. We want to show that R_e acts finitely often. To prove this, for each stage $s \geq s_0$, we will choose an element of c_s of the linear order $(\omega + 1)^n$ (if $a, b \in (\omega + 1)^n$ then $a < b$ if on the first coordinate that a and b differ the a value on this coordinate is smaller

than the b value). We will ensure that for all $s \geq s_0$, $c_{s+1} \leq c_s$ and if R_e acts at stage $s+1$, then $c_{s+1} < c_s$. Hence as $(\omega+1)^n$ is a well-order, we have that R_e can only act finitely often.

To determine c_s we first let $T \subseteq S_e$ be the set of all strings that cause R_e to act after stage s_0. Because any element of T causes R_e to change, the next element of T to enter S_e must be of strictly greater length. Let $\sigma_{0,s}, \sigma_{1,s}, \ldots, \sigma_{m_s,s}$ be a list of all strings in T_s that agree with G_s after r. We assume that the strings are numbered so that $|\sigma_{0,s}| < |\sigma_{1,s}| < \ldots < |\sigma_{m_s,s}|$. For any $s \geq s_0$, we have that $m_s < n$. This is because if $m_s \geq n$, then because there are only n strings of length r, there would be $i < j \leq m_s$ with $\sigma_{i,s} \upharpoonright r = \sigma_{j,s} \upharpoonright r$. As both strings agree with G_s after r, this implies that $\sigma_{i,s} \prec \sigma_{j,s}$. However, it is not possible for two such strings to cause requirement R_e to act (by condition (ii) for R_e to need attention and the fact that strings enter T in increasing order).

Fix some $i \leq m_s$. Provided that for all $t \geq s$ and all $j < i$ we have that $\sigma_{j,t+1} = \sigma_{j,t}$, our objective is to bound the size of the set $\{t : \sigma_{i,t+1} \neq \sigma_{i,t}\}$. First we will find a set of strings that includes all possible values that $G_t \upharpoonright |\sigma_{i,t}|$ could take for $t \geq s$. We define a string ρ as being an (i,s)-candidate if:

(i). $\rho \not\succ G_s \upharpoonright |\sigma_{i,s}|$.

(ii). $\rho \geq_{lex} G_s \upharpoonright |\rho|$.

If $\sigma_{i,s} \neq \sigma_{i,s+1}$ then $\sigma_{i,s+1}$ must differ from $\sigma_{i,s}$ at some point in $\mathrm{rng}\Gamma(G_s)[s]$. We define:

$$C_s(i) = \{x \in \omega : x \in \mathrm{rng}\Gamma(\rho)[s] \text{ and } \rho \text{ is an } (i,s)\text{-candidate}\}.$$

We will show that the size of this set bounds the possible number of times that $\sigma_{i,s}$ changes. Thus we define:

$$c_s(i) = \begin{cases} |C_s(i)| & \text{if } i \leq m_s \\ \omega & \text{otherwise.} \end{cases}$$

This ends our definition of c_s.

Lemma 10.6.5. *If $i \leq \min\{m_s, m_{s+1}\}$, and $\sigma_{i,s+1} = \sigma_{i,s}$ then $c_{s+1}(i) = c_s(i)$.*

Proof. We have that $\sigma_{i,s}$ agrees with $G_s \upharpoonright |\sigma_{i,s}|$ after r, and $\sigma_{i,s+1}$ agrees with $G_{s+1} \upharpoonright |\sigma_{i,s+1}|$ after r, hence $G_s \upharpoonright |\sigma_{i,s}| = G_{s+1} \upharpoonright |\sigma_{i,s+1}|$. Thus if ρ is an (i,s)-candidate, then it is also an $(i,s+1)$-candidate.

Now at stage $s+1$, if a new axiom is enumerated for Γ, it must be on some extension of $G_s \upharpoonright w(e,s)$. As $w(e,s) \geq |\sigma_{i,s+1}|$, any new axiom must be on a string which is not an $(i,s+1)$-candidate. Hence we have that $C_{s+1}(i) = C_s(i)$ and consequently $c_{s+1}(i) = c_s(i)$. □

Lemma 10.6.6. *If R_e does not act at stage $s+1$, then $c_{s+1} = c_s$.*

Proof. If R_e does not act at stage $s+1$, then $T_{s+1} = T_s$ and $G_{s+1} \upharpoonright w(e, s+1) = G_{s+1} \upharpoonright w(e,s)$ so $m_{s+1} = m_s$. Thus for all $i \leq m_s$, $\sigma_{i,s} = \sigma_{i,s+1}$ and so be the previous lemma $c_s(i) = c_{s+1}(i)$. For i with $m_s < i < n$ we have that $c_{s+1}(i) = \omega = c_s(i)$. □

Lemma 10.6.7. *If R_e acts at stage $s+1$, then $c_{s+1} < c_s$.*

Proof. If R_e acts at stage $s+1$, then some new string enters T_{s+1}. This means that either there is some $i \leq \min\{m_s, m_{s+1}\}$ with $\sigma_{i,s+1} \neq \sigma_{i,s}$, or $m_{s+1} = m_s + 1$. For the second case we have that $c_{s+1}(i) = c_s(i)$ for all $i \leq m_s$ and $c_{s+1}(m_{s+1}) < \omega = c_s(m_{s+1})$. Thus $c_{s+1} < c_s$.

For the first case, let i be least such that $\sigma_{i,s+1} \neq \sigma_{i,s}$. We have already established that $c_{s+1}(j) = c_s(j)$ for all $j < i$.

Our first claim is that $G_{s+1} \upharpoonright |\sigma_{i,s}| >_{lex} G_s \upharpoonright |\sigma_{i,s}|$. This claim holds because first $G_{s+1} \upharpoonright |\sigma_{i,s+1}|$ cannot be an extension of $G_s \upharpoonright |\sigma_{i,s}|$ (if so it would not be the ith string in order of size in T_{s+1} that agrees with G_{s+1} after r). Hence $\sigma_{i,s+1}$ does not agree with G_s at some point between $|\sigma_{i-1,s}|$ (or r if $i = 0$) and $|\sigma_{i,s}|$. The only places that $\sigma_{i,s+1}$ can disagree with G_s are those elements of $\mathrm{rng}\Gamma(G)[s]$. This is a subset of $C_s(i)$. Now if $\Gamma(G; m)[s] \downarrow = n$ then $G_s(n) = 0$. This implies that we must form $G_{s+1} \upharpoonright |\sigma_s|$ from $G_s \upharpoonright |\sigma_s|$ by changing some elements to 1 and so $G_{s+1} \upharpoonright |\sigma_{i,s}| >_{lex} G_s \upharpoonright |\sigma_{i,s}|$. Consequently the $(i, s+1)$-suitable strings are a strict subset of the (i,s)-suitable strings. Further there must have been some element of $C_s(i)$ that is now an element of G_{s+1} and hence $C_{s+1}(i) \subsetneq C_s(i)$ and so $c_{s+1}(i) < c_s(i)$.

□

Hence the stages at which R_e acts can be mapped to a decreasing sequence of a well-order and so requirement R_e needs attention finitely often. As each requirement needs attention finitely often, we have that the use of $\Gamma(G; e)$ is bounded in the construction and so requirement I_e is met. For all e, $\lim_s w(e,s)$ exists and witnesses the fact that requirement R_e is met. □

Once we know that there exist 1-generic sets that compute their own indifferent sets, we can investigate where in the Δ^0_2 degrees such 1-generic sets exist. The verification of Theorem 10.6.4 showed that each requirement was met by constructing a descending sequence in $(\omega+1)^n$ for some n. In the following theorem, we will show how this can be turned into a permission argument. We can build a 1-generic set G that computes an indifferent set for itself below some c.e. set A, if A will provide us with enough permissions. We

will show that if A is not of totally ω^ω-c.a. degree then sufficient permissions must be given.

Theorem 10.6.8. *If a c.e. set A is not of totally ω^ω-c.a. degree then there exists G, I with $A >_T G \geq_T I$ such that G is 1-generic, and I is an indifferent set for G.*

Proof. Let A be such a set and let $f = \Psi(A)$ be a function that witnesses that A is not of totally ω^ω-c.a. degree. Let $f(x,s) = \Psi(A;x)[s]$ be a computable approximation to f. We will build G via a reduction $\Phi(A)$, and set $I = \text{rng}\Gamma(G)$ where $\Gamma(G)$ is a strictly increasing function. To construct G we will need to obtain A-permissions. We will do this by attempting to show that f is ω^ω-c.a. In fact to simplify the exposition of the proof we will attempt to show that f is $(\omega+1)^\omega$-c.a. The result will hold because our well-order $(\omega+1)^\omega$ can just be regarded as another canonical well-ordering of ω^ω.

The failure of this approach will ensure that we get the permissions we need to build G and I. We can assume that our approximation to A has the property that for all stages s, $f(x,s)$ is defined for all $x \leq s$. We let $u(x,s)$ be the use of A_s in the computation of $f(x,s)$. We can assume that u is non-decreasing in both arguments.

Our first requirement is that $\Phi(A)$ is total. Then taking $G = \Phi(A)$ we have for all $e \in \omega$:

I_e: $\Gamma(G;e)\downarrow$ and if $e > 0$, $\Gamma(G;e) > \Gamma(G;e-1)$.

R_e: $\exists w$ such that $\forall X \subseteq \text{rng}\Gamma(G)$, $G_{[X]} \upharpoonright w$ meets or avoids S_e.

The approach we use to meet a requirement R_e is as follows. Each time the requirement is injured, we start afresh with a new function g_e that we use to approximate f. We pick some number q_0 greater than the restraint imposed on R_e, and we attempt to run the same strategy of Theorem 10.6.4 using q_0 instead of r. We call q_0 a sub-strategy of R_e. To run this sub-strategy we need to be able to change our approximation to G after q_0. However, we cannot do this unless we get permission from A to do so. To get this permission we will link the A-use of the computation of G to the A-use of the computation of f. Specifically, if we act on sub-strategy q_0 at stage s, then will ensure that for all $t \geq s$, we have that $\phi(A;q_0)[t] \geq u(s,t)$. Suppose that at some later stage t, we see some string σ enter $S_{e,t}$ and we want to change G after q_0 to agree with σ. We ask A for permission by defining $g_e(x,t+1) = f(x,t+1)$ for all $x \in [r+1,s]$. Now if f changes on any of these values, then we know that A has changed below $u(s,t) \leq \phi(A;q_0)[t]$ and thus we can change G as desired. If on the other hand, f fails to change, then we have made some progress in building an approximation to f. We will show this approximation is a $(\omega+1)^\omega$ computable

approximation by defining a function $o_e(x, s)$ that tracks the changes of g_e. While we are waiting for permission to change G at q_0 we initiate another sub-strategy q_1. We will keep initiating new strategies at stages at which it appears that all earlier sub-strategies are successfully approximating f. We will show in the verification that one sub-strategy must succeed because otherwise we will establish that f is $(\omega + 1)^\omega$-c.a. We will assign all active sub-strategies one of two states: *good* or *waiting*. The *good* state indicates that a sub-strategy currently believes it has established the existence of a w witnessing that we have met R_e. The witness for a strategy q at stage s will be $\phi(A; q)[s]$. The *waiting* state indicates that the sub-strategy is seeking permission to change the current approximation to G.

Construction. Our requirements will be prioritised as follows: $R_0 > I_0 > R_1 > I_1 > \ldots$. A requirement I_e needs attention at stage $s + 1$ if $\Gamma(G; e)[s] \uparrow$. If a requirement R_e is injured then all of its sub-strategies are halted. We say that a sub-strategy q in state *waiting* has permission to act if $\Phi(A; q)[s] \uparrow$. A sub-strategy q of requirement R_e needs attention if:

(i). It is in state *waiting* and has permission to act; or

(ii). It is in state *good* and there exists $\sigma \in S_{e,s}$ with $|\sigma| > \phi(A; q)[s]$, such that:

  (a) $(\sigma \upharpoonright \phi(A; q)[s])\Delta(G_s \upharpoonright \phi(A; q)[s]) \subseteq \text{rng}\Gamma(G)[s]$, and

  (b) No initial segment of $\sigma \upharpoonright \phi(A; q)[s]$ is in $S_{e,s}$.

A requirement R_e needs attention at stage s if:

(i). It has a sub-strategy that needs attention; or

(ii). No sub-strategy of R_e needs attention, and all sub-strategies (if any) are in state *waiting*.

At stage 0, set $G_s = 0^\omega$ and for all e, x set $g_e(x, 0) \uparrow$ and $o_e(x, 0) \uparrow$.

At stage $s+1$, find the highest priority requirement that needs attention. If there are no requirements that need attention, set $G_{s+1} = G_s$.

If the requirement that needs attention is of the form I_e, take some large value n such that $G_s(n) = 0$ and set $G_{s+1} = G_s$ and $\Gamma(G; e)[s + 1] = n$. Injure all lower priority requirements and restrain $G_{s+1} \upharpoonright n$ with priority e.

If the requirement is of the form R_e and R_e has some sub-strategy that needs attention, then let q be the smallest such sub-strategy. If q is in state *good*, then change q to state *waiting* and set $G_{s+1} = G_s$. Halt any sub-strategy

q' of R_e with $q' > q$. If q is in state *waiting*, then let σ be the string in $S_{e,s}$ that caused q to change its state to *waiting*. We set:

$$G_{s+1}(x) = \begin{cases} G_s(x) & x \leq q \\ \sigma(x) & q < x < |\sigma| \\ 0 & |\sigma| \leq x. \end{cases}$$

We will commit to keeping $\phi(A; x)[t] \geq u(s+1, t)$ for all $t \geq s+1$ for any x such that $x \geq q$. Halt any sub-strategy q' of R_e with $q' > q$.

If no sub-strategy of R_e needs attention, then we start a new sub-strategy on $s+1$ and we will commit to keeping $\phi(A; s+1)[t] \geq u(s+1, t)$ for all $t \geq s+1$.

Now that we have updated G_s, we update the reduction Φ. For all $x \leq s+1$ we set $\Phi(A; x)[s+1] = G_{s+1}(x)$. As we only change G_{s+1} when we get permission from our approximation to A, Φ is a valid Turing functional (we will later verify that $\Phi(A)$ is total). We will set the use of $\Phi(A; x)[s+1]$ for all $x \leq s+1$ to be as small as possible while still honouring the commitments made.

We will now define the functions g_e and o_e. For all $e \leq s+1$, we do the following. Take any $x \leq s+1$. If $u(x, s+1) \leq \phi(A; q)[s+1]$ for some sub-strategy q of R_e that is in state *waiting*, we define $g_e(x, s+1) = f(x, s+1)$. Otherwise we set $g_e(x, s+1)$ to be undefined. Let r_e be the restraint currently imposed on requirement R_e. Let $n_x = \sum_{i=r_e+1}^{x} 2^i$. For all x such that $g_e(x, s+1)$ is defined, we will define $o_e(x, s+1)$ to be an element of the well order $(\omega+1)^{n_x}$. First, for all x we define $c_{s+1,x}$ to be an element of $(\omega+1)^{(2^x)}$. We let $c_{s+1,x} = 0^{(2^x)}$ if x is not an active sub-strategy of R_e. Otherwise, we define $c_{s+1,x}$ to be the element of $(\omega+1)^{(2^x)}$ as in Theorem 10.6.4 (we take x instead of r, and $T \subseteq S_{e,s}$ to be the strings that cause this sub-strategy to change from state *good* to *waiting*). We define $o_e(r+1, s) = c_{s+1,x}$ and then for $x > r+1$ we define o_e inductively by $o_e(x+1, s) = o_e(x, s)c_{s+1,x+1}$.

Verification. Take any requirement and assume that by stage s_0 all higher priority requirements have finished acting. If the requirement is of the form I_e then by stage s_0+1 we have defined $\Gamma(G; e)[s+1]$ and restrained G on the use of this computation.

Assume the requirement is of the form R_e. The first observation is that any sub-strategy can only be acted on a finite number of times. This follows from the verification of Theorem 10.6.4. Given any x, let s be the last stage at which we act on any sub-strategy (for any requirement R_e) less than or equal to x. Now $\phi(A; x) = u(s)$ and so we have that $\Phi(A)$ is total.

Lemma 10.6.9. *For all x, s, t with $s < t$, if $o_e(x, s) \downarrow$ and $o_e(x, t) \downarrow$ then $o_e(x, s) \geq o_e(x, t)$.*

Proof. Assume that this is false and that there is some least x for which there exist s and t with $s < t$ such that $o_e(x, s) < o_e(x, t)$. Now because $o_e(x, s) = c_{s,r+1}c_{s,r+2}\ldots c_{s,x}$, there must be some least y with $r+1 \leq y \leq x$ such that $c_{s,y} < c_{t,y}$, and y must be an active sub-strategy of R_e at stage t (because otherwise $c_{t,y} = 0^{(2^y)}$). This means that y was also an active sub-strategy of R_e at stage s because $y \leq x \leq s$ and the sub-strategy y must have been started at stage y.

Now no higher priority requirements have acted since stage s. Additionally no R_e sub-strategies less than y have acted because if they did, then sub-strategy y would be halted. Hence if for any s' with $s \leq s' < t$ we have that $G_{s'} \upharpoonright \phi(A; y)[s'] \neq G_{s'+1} \upharpoonright \phi(A; y)[s']$ this must have been the result of sub-strategy y acting and so by the verification of Theorem 10.6.4 we know that $c_{s,y} \geq c_{t,y}$ contradicting our assumption. □

Lemma 10.6.10. *For all x, s, t with $s < t$ if $g_e(x, s) \downarrow$ and $g_e(x, t) \downarrow$ with $g_e(x, s) \neq g_e(x, t)$ then $o_e(x, s) > o_e(x, t)$.*

Proof. If $g_e(x, s) \downarrow$, then $g_e(x, s) = f(x, s)$ and there exists some R_e sub-strategy $q \leq x$ such that q is in state *waiting* at stage s and $\phi(A; q)[s] \geq u(x, s)$. Now if q is in state *waiting* at stage s then all sub-strategies $q' < q$ of R_e are also in state *waiting* at stage s (when strategy q started q' was in state *waiting* and if q' ever changed to state *good* then q would be halted).

If $g_e(x, t) \downarrow \neq g_e(x, s)$, then $f(x, s) \neq f(x, t)$ and so $A_s \upharpoonright u(x, s) \neq A_t \upharpoonright u(x, s)$. This means that some smallest sub-strategy $q' \leq q$ will have been given permission to act and have acted. As this is the smallest sub-strategy to act, we know that $c_{s,q'} > c_{t,q'}$ by the verification of Theorem 10.6.4. If $q' = r + 1$ this gives us that $o_e(q', s) > o_e(q', t)$. If $q' > r+1$ then as $o_e(q'-1, s) \geq o_e(q'-1, t)$ by the previous lemma we again have that $o_e(q', s) = o_e(q' - 1, s)c_{s,q'} > o_e(q' - 1, t)c_{t,q'} = o_e(q', t)$. As $x \geq q'$, we have that $o_e(x, s) > o_e(x, t)$. □

Lemma 10.6.11. *If R_e acts infinitely often, then for infinitely many x there are infinitely many s such that $g_e(x, s) \downarrow$.*

Proof. As observed, any sub-strategy only needs to act a finite number of times to meet requirement R_e. Hence if R_e acts infinitely often then R_e must have an infinite number of sub-strategies and all of the sub-strategies must have some final state of *waiting*. Let r_e be the final restraint imposed upon R_e by higher priority requirements. Pick any $x > r_e$ and take the least sub-strategy q such that $q > x$ and some stage s such that q is in state *waiting* for all stages $t \geq s$. In this case $g_e(x, t)$ is defined for all $t \geq s$. □

If $g_e(x,s)\downarrow$, then $g_e(x,s) = f(x,s)$. Thus if R_e acts infinitely often, let $\hat{g}_e(x,s) = g_e(x, \min\{t \geq s : g_e(x,t)\downarrow\})$, and let $\hat{o}_e(x,s) = o_e(x, \min\{t \geq s : o_e(x,t)\downarrow\})$. For all $x > r_e$, $\lim_s \hat{g}_e(x,s) = \lim_s g_e(x,s) = f(x)$ and so $\hat{g}_e$ and $\hat{o}_e$ witness that f is $(\omega+1)^\omega$-c.a. This contradicts our initial assumption and so R_e acts finitely often. Hence all requirements act finitely often and so our construction is successful. □

Corollary 10.6.12. *If A is a c.e. set and $A'' >_T \emptyset''$ then A bounds a 1-generic set that computes an indifferent set for itself.*

Proof. If a c.e. set A is of totally ω^ω-c.a. degree then $A'' \equiv_T \emptyset''$ [26]. □

We will now prove an extension of Corollary 10.6.2. We will show in Theorem 10.6.16 that no 1-generic set bounded by a c.e. set that is of totally ω-c.a. degree computes a set to which it is indifferent. To establish this result, we will show that any approximation to a 1-generic set that computes an indifferent set to itself changes frequently. Given G and I with $p_I = \Gamma(G)$, we define the number of times G changes on the use of the xth point of I as follows.

$$\#G(I,x) = |\{s : \Gamma(G;x)[s]\downarrow \wedge G_s \upharpoonright \gamma(G;x)[s] \neq G_{s+1} \upharpoonright \gamma(G,x)[s]\}|.$$

Proposition 10.6.13. *Assume that $I \leq_T G \leq_T \emptyset'$ with $p_I = \Gamma(G)$ and the reduction Γ has the property that $\forall x$, $\Gamma(G;x+1) > \gamma(G;x)$. If for some computable function q, and some computable approximation to G, G_s we have that:*

$$\forall x\ q(x+1) - q(x) \geq (\#G(I, q(x+1)) + 1)^2$$

then I is not an indifferent set for G.

Proof. We will construct a set A and a c.e. set of strings V such that $A\Delta G \subseteq I$ and A does not meet or avoid V. The set A will be defined as the limit of a sequence $\alpha_0 \prec \alpha_1 \prec \ldots$. We have the following requirements:

R_e: $\alpha_{e+1} \succ \alpha_e$, α_e does not avoid V, α_{e+1} does not meet V and $\alpha_{e+1}\Delta(G \upharpoonright |\alpha_{e+1}|) \subset \text{rng}\Gamma(G)$.

We will build an approximation $\alpha(e,s)$ such that for all e, $\alpha_e = \lim_s \alpha(e,s)$. A requirement R_e needs attention at stage s, if for all $x \leq q(e+1)$ we have that $\Gamma(G;x)[s]\downarrow$; if $x > 0$ then $\Gamma(G;x)[s] \geq \gamma(G;x-1)[s]$ and either:

(i). There is no extension of $\alpha(e,s)$ in V_s, or

(ii). $\alpha(e+1,s)$ is undefined, or

(iii). The requirement last acted at stage t and

$$G_t \upharpoonright |\alpha(e+1,t)| \neq G_s \upharpoonright |\alpha(e+1,t)|.$$

At stage $s = 0$, we set $\alpha(0, 0) = \lambda$ and we declare $\alpha(e, 0)$ to be undefined for all $e > 0$. At stage $s + 1$, we find the highest priority requirement that needs attention R_e. If no requirement needs attention then for all e such that $\alpha(e, s) \downarrow$ we set $\alpha(e, s + 1) = \alpha(e, s)$ and otherwise we set $\alpha(e, s + 1) \uparrow$.

If R_e is the highest priority requirement that needs attention, then take n_e to be the largest integer such that $n_e^2 \leq q(e + 1) - q(e)$. We partition $\{\Gamma(G; x) : q(e+1)-n_e^2 < x \leq q(e+1)\}$ into sets $I_{s,1}, I_{s,2}, \ldots, I_{s,n_e}$, each of size n_e, and such that $\max I_{s,j} < \min I_{s,j+1}$. Define $\rho = G_s \upharpoonright |a(e, s)|$ and $u = \gamma(G_s; q(e + 1))[s]$. Let m be the number of times that requirement R_e has acted previously in the construction . We will take as our first construction assumption that $m < n_e-1$ and verify this assumption later.

If there is no extension of $\alpha(e, s)$ in V_s, then we take $i = \min I_{s,n_e}$, and we add the string $\alpha(e, s)\sigma$ to V_{s+1}, where σ is chosen so that $|\rho\sigma| = u$ and $\rho\sigma\Delta(G_s \upharpoonright u) = \{i\}$.

We define the following set of candidates for $\alpha(e + 1, s + 1)$:

$$U_{s+1} = \{\alpha(e, s)\sigma : |\alpha(e, s)\sigma| = u \wedge \rho\sigma\Delta(G_s \upharpoonright u) = \{i\} \wedge i \in I_{s,n_e-m-1}\}.$$

We will assume for now that there is some string $\alpha(e, s)\sigma \in U_{s+1}$ that is incomparable with all $\alpha(e + 1, t)$ for all $t \leq s$ (where defined) and also incomparable with any extension of $\alpha(e, s)$ added to V by this requirement. This is the second construction assumption. We take such a string $\alpha(e, s)\sigma$ (e.g. the lexicographically least) and set $\alpha(e + 1, s + 1) = \alpha(e, s)\sigma$. For all $d \leq e$ we set $\alpha(d, s + 1) = \alpha(d, s)$ and for all $f > e + 1$, we set $\alpha(f, s + 1)$ to be undefined.

Verification. If a requirement acts at a stage s, we will say that this action is successful if both of the construction assumptions made are met. We will inductively show that:

(i). Whenever requirement R_e acts, it does so successfully.

(ii). R_e acts finitely often.

Lemma 10.6.14. *If R_e acts at stage $s + 1$, and it has acted less than $n_e - 1$ times before, then R_e acts successfully.*

Proof. As our first construction assumption is met by hypothesis we only need to check that U_{s+1} contains a suitable string. Any strings enumerated into V_{s+1} by this requirement extend $\alpha(e, t)$, for some $t \leq s$. By our induction hypothesis (that the second construction assumption is met for R_{e-1}) this set of strings forms an anti-chain (for the case $e = 0$, we have that $a(0, t) = \lambda$ for all t). Thus there is only one string enumerated in V_{s+1} by this requirement that we need

to avoid. In addition, there are at most $n_e - 2$ strings that are equal to $\alpha(e+1, t)$ for some $t \leq s$ as this is the maximum number of times this requirement has acted before.

Let m be the number of times this requirement has acted before. Let T be the set of all strings we want to avoid. Note that $|T| \leq n_e - 1$. By induction on the stages at which R_e acts, we can assume that T is an anti-chain (i.e. the second construction assumption has held every previous time R_e has acted). The size of the set U_{s+1} is n_e. We will show that for all $\tau \in T$, there is at most one string $\alpha(e,s)\sigma \in U_{s+1}$ such that $\tau \preceq \alpha(e,s)\sigma$, or $\alpha(e,s)\sigma \preceq \tau$. Hence as $|U_{s+1}| > |T|$, we can choose some string $\sigma \in U_{s+1}$ such that $T \cup \{\alpha(e,s)\sigma\}$ is an anti-chain and set $\alpha(e+1, s+1) = \alpha(e,s)\sigma$.

Take any $\tau \in T$. If τ is not an extension of $\alpha(e,s)$ then τ must extend some $\alpha(e,t)$ for some $t < s$ with $\alpha(e,t)$ defined and $\alpha(e,t) \neq \alpha(e,s)$. Because the second construction assumption holds for all higher priority requirements, we know that $\alpha(e,s)$ and $\alpha(e,t)$ are incomparable and hence τ is incomparable with all elements of U_{s+1}. So we can assume that $\tau \succeq \alpha(e,s)$ and we can write $\tau = \alpha(e,s)\hat{\tau}$. Let $\rho = G_s \upharpoonright |a(e,s)|$. Now because U_{s+1} is an anti-chain the only way that $\alpha(e,s)\hat{\tau}$ can be comparable with two elements of U_{s+1} is if $\alpha(e,s)\hat{\tau} \preceq \alpha(e,s)\sigma_j$ and $\alpha(e,s)\hat{\tau} \preceq \alpha(e,s)\sigma_k$ for distinct $\alpha(e,s)\sigma_j, \alpha(e,s)\sigma_k \in U_{s+1}$. In this case, it must be that $\rho\hat{\tau} \preceq G_s$ because any common initial segment of two elements of U_{s+1} must agree with G_s after $|\alpha(e,s)|$.

Let t be the stage when either τ was added to V or the least stage when τ was equal to $\alpha(e+1, t)$. Let m' be the number of times that the requirement has acted before stage t. Hence $m' < m$. As no higher priority requirement has acted we have that $\rho \preceq G_t$. By construction, it must be that $\rho\hat{\tau}\Delta(G_t \upharpoonright |\tau|) = \{a\}$ for some $a \in I_{t,n_e}$ if τ was added to V, and $a \in I_{t,n_e-m'-1}$ otherwise. Now any element of the range of $\Gamma(G)[t]$ less than a is also an element of the range of $\Gamma(G)[s]$ because any element of the range of $\Gamma(G)[t]$ less than a has use less than a and $\rho\hat{\tau}$ agrees with both G_s and G_t before a.

We can draw a contradiction from this fact because $\alpha(e,s)\hat{t} \preceq \alpha(e,s)\sigma_j$ implies that the first element of the range of $\Gamma(G)[s]$ greater than or equal to a cannot be in I_{s,n_e-k} if $k \leq m$. This in turn implies that a cannot be in I_{t,n_e-k} if $k \leq m$ which contradicts our assumption that $a \in I_{t,n_e-m'-1}$ or $a \in I_{t,n_e}$. Hence there is some suitable string in U_{s+1} and requirement R_e acts successfully. □

Lemma 10.6.15. *If requirement R_e acts $n_e - 1$ times then it never needs attention again.*

Proof. If R_e acts at stage t, and then again at some $s > t$, it must be that for

some $d \leq e$, we have that for some stage s' with $t < s' \leq s$:

$$G_{s'} \upharpoonright \alpha(d+1,t) \neq G_{s'+1} \upharpoonright \alpha(d+1,t).$$

Now $|\alpha(d+1,t)| = \gamma(G; q(d+1))[t] \leq \gamma(G; q(e+1))[t]$. Thus $G_t \upharpoonright \gamma(G; q(e+1))[t] \neq G_{s'} \upharpoonright \gamma(G; q(e+1))[t]$.

This means that the number of times that R_e acts is at most $\#G(I, q(e+1))$. But as $(\#G(I, q(e+1))+1)^2 \leq q(e+1) - q(e)$, by our choice of n_e we have that $\#G(I, q(e+1)) \leq n_e - 1$. □

We have that each requirement acts finitely often. This means that for all e, $\alpha_e = \lim_s \alpha(e,s)$ is defined. Further $\alpha_{e+1} \succ \alpha_e$ because if $\alpha(e+1,s) \downarrow$ then $\alpha(e+1,s) \succ \alpha(e,s)$. Let $A = \bigcup_e \alpha_e$. Now $\alpha_{e+1}\Delta(G \upharpoonright |\alpha_{e+1}|) \subset \text{rng}\Gamma(G)$ because if s is a stage when requirement R_e acts then $\alpha(e+1,s)\Delta(G_s \upharpoonright |\alpha(e+1,s)|) \subset \text{rng}\Gamma(G)[s]$ and the use of any indifferent point in $\text{rng}\Gamma(G)[s] \cap |\alpha(e+1,s)|$ is at most $|\alpha(e+1,s)|$. Now if the approximation to G changes on the first $|\alpha(e+1,s)|$ bits, then $\alpha(e+1,s) \neq \alpha_{e+1}$ because the requirement will need attention again.

As each requirement is met we have that A does not avoid V (for all e, α_e has some extension in V). Let σ be an element of V. There is some requirement R_e that enumerated σ into V and some stage s such that $\sigma \succ \alpha(e,s)$. If $\alpha_e \neq \alpha(e,s)$ then these two strings are incomparable and so A does not meet σ. If $\alpha_e = \alpha(e,s)$, then by construction α_{e+1} is incomparable with σ. Thus A is not 1-generic and so I is not an indifferent set for G. □

Theorem 10.6.16. *If a c.e. set A is of totally ω-c.a. degree then A does not bound a 1-generic set G that computes an indifferent set for itself.*

Proof. Assume that $G = \Phi(A)$ and that $\Gamma(G)$ is equal to the principal function of some set I. We can assume that for all $x > 0$, $\Gamma(G; x+1) > \gamma(G; x)$ because if G can compute an indifferent set for itself then it can compute one with this property.

We will show that I is not an indifferent set for G. Our approach is to build a function $f \leq_T A$, where $f(x) = \Psi(A; x)$. Because A is of totally ω-c.a. degree we know that there is a pair of computable functions (g,h) such that $f(x) = \lim_s g(x,s)$ and if we define $\#g(x) = |\{s+1 : g(x,s) \neq g(x,s+1\}|$, then $\#g(x) \leq h(x)$. However, we do not know which computable functions these are. We will take an enumeration of all pairs of partial computable functions: $(g_e, h_e)_{e<\omega}$. Our objective in the construction of f is to find the pair of functions g and h that witness that f is ω-c.a., and then use this pair to slow the approximation of A sufficiently so that I is not an indifferent set for G.

We call (g_e, h_e) a *valid ω-c.a. approximation* for f if:

(i). g_e and h_e are total.

(ii). $\forall x$, $\#g_e(x) \leq h_e(x)$.

(iii). $\forall x$, $\lim_s g_e(x,s) = f(x)$.

We cannot determine whether (g_e, h_e) is a valid ω-c.a. approximation for f. But given x and s, we can determine the following. We call (g_e, h_e) a *valid ω-c.a. approximation for f until x at stage s* if for all $y \leq x$:

(i). $h_e(y)[s]\downarrow$.

(ii). $g_e(y, \max\{j : g_e(y,j)[s]\downarrow\}) = \Psi(A;x)[s]$.

(iii). $|\{j \in \omega : g_e(y,j)[s]\downarrow \neq g_e(y,j+1)[s]\downarrow\}| \leq h_e(y)[s]$.

A pair (g_e, h_e) is a valid ω-c.a. approximation for f if and only if for all x there is an s such that (g_e, h_e) is a valid ω-c.a. approximation for f until x at stage s.

Our requirements for the construction are:

(i). For all x, $\Psi(A;x)\downarrow$.

(ii). Either A is computable, or there some approximation to G, and some strictly increasing computable function q, such that:

$$\forall x\ (q(x+1) - q(x) > (\#G(I, q(x)) + 1)^2).$$

We will assume that our enumeration of A is sufficiently fast so that for all $x \leq s$, $\Gamma(\Phi(A);x)[s]\downarrow$. During the construction we will build a computable function $r(x,s)$. The function $r(x,s)$ determines a point in the set I. This point in I is computable from A by composing Γ and Φ. Hence it has an A-use. We will denote this use as $\gamma \circ \Phi(A; r(x,s))$. When possible, we will set the use of $\Psi(A;x)[s]$ to be this value.

Let $e, s, x \in \omega$ with $e, x \leq s$. At any stage s, we say that e can *take control* of x if:

(i). x is not controlled by any $d < e$.

(ii). (g_e, h_e) is a valid ω-c.a. approximation for f until x at stage s.

Construction. At each stage in the construction we will define $r(x, s)$ for all $x \leq s$. We will also ensure that $\Psi(A; x)[s]$ is defined for $x \leq s$.

At stage 0 we define $\Psi(A; 0)[0] = 0$, $r(0, 0) = 0$ and we define the use of $\Psi(A; 0)[0]$ to be $\gamma \circ \Phi(A; r(0, 0))[r(0, 0)]$.

At stage $s + 1$, we ask whether there is any $e, x \leq s$ such that e can take control of x. If so we take the least such e, and give it control of the least x it can take control of. Define $r(0, s + 1) = 0$. Now define inductively

$$r(x + 1, s + 1) = \begin{cases} r(x, s + 1) + (h_e(x + 1) + 1)^2 & \text{if } e \text{ controls } x + 1, \\ r(x, s + 1) + 1 & \text{if no } e \text{ controls } x + 1. \end{cases}$$

Now for all $x \leq s + 1$, such that $\Psi(A_{s+1}; x)[s] \uparrow$ we define

$$\Psi(A_{s+1}; x)[s + 1] = \begin{cases} \max\{g_e(x, j)[s] : j \leq s\} + 1 & \text{if } e \text{ controls } x, \\ 0 & \text{if no } e \text{ controls } x. \end{cases}$$

For these x, we set the A use of the computation of $\Psi(A_{s+1}; x)[s + 1]$ to be $\gamma \circ \Phi(A; r(x, s + 1))[r(x, s + 1)]$.

Verification. First we show that $f = \Psi(A)$ is total. This can be done by showing that the use of any computation does not tend to infinity. This is true if for all x, $\lim_s r(x, s)$ exists. Now $\lim_s r(0, s) = 0$ and if the $\lim_s r(x, s)$ exists, then as $x + 1$ has finitely many owners so $\lim_s r(x + 1, s)$ exists.

By assumption A is ω-c.a., so it follows that there is some least e such that (g_e, h_e) is a valid ω-c.a. approximation to f. We claim that e takes control of almost all x. If $d < e$, then there is some x such that for all s, (g_d, h_d) is never a valid approximation for f until x at stage s. Thus d never takes control of any element greater than or equal to x. As only a finite subset of ω is controlled any $d < e$, e will take control of the remaining elements of ω. This makes $r : \omega \to \omega$, defined by $r(x) = \lim_s r(x, s)$, computable, because for almost all x the limit occurs at the stage e takes ownership of x.

If for almost all x, $\Psi(A; x) = 0$, then A is computable. We just wait until a stage s at which e takes ownership of x. At this point, A cannot change on the use of $\Psi(A; x)[s]$; if it did, then $\Psi(A; x)$ would be set to some non-zero value. As the use of $\Psi(A)$ is unbounded we have that A is computable.

Otherwise, there exists infinitely many x where $\Psi(A; x)$ is defined after e takes ownership of x. The set of these x is c.e. and so has some infinite computable subset. We can take this subset to be the range of a strictly increasing computable function p.

Now we can speed-up the enumeration of G so that stage t in the new enumeration corresponds to the first stage s when e has control of all $x \leq t$

(that it will take control of) and (g_e, h_e) is a valid approximation for f until t. Now the computable function $r(p(x))$ has the property that for all x,

$$r(p(x+1)) - r(p(x)) \geq r(p(x+1)) - r(p(x+1)-1) \geq (h_e(p(x+1))+1)^2.$$

We claim that for all x, $\#G(I; r(x)) \leq h_e(x)$. This is because if $G_{t+1} \upharpoonright \gamma(G; r(x))[t] \neq G_t \upharpoonright \gamma(G; r(x))[t]$, then $A_{t+1} \not\succ A_t \upharpoonright \gamma \circ \Phi(A; (r(x)))$. This means that $\Psi(A; x)[t+1] > \Psi(A; x)[t]$ which in turn implies that $g_e(x; t+1) > g_e(x; t)$. We know these changes are bounded by $h_e(x)$. Thus for all x,

$$r(p(x+1)) - r(p(x)) \geq (\#G(I, r(p(x+1))+1)^2.$$

We now apply Proposition 10.6.13 to establish that I is not an indifferent set for G with $q = r \circ p$. □

10.7 Indifferent sets for weakly 1-generic sets

In this section we will further examine indifferent sets for weakly 1-generic sets. We start with an observation about the function $g(n, e)$ defined in Lemma 10.5.2. Fix $e \in \omega$. If S_e is dense, then $g(n, e)$, as a function of the first variable, is computable, because we do not need to ask $\emptyset'$ whether some string σ has an extension in S_e. The answer is always "yes." We can use this observation to prove that any hyperimmune set computes a weakly 1-generic set that it is the indifferent set for. For this section we will take $g(n, e)$ to be a partial computable function such that if S_e is dense, then $g(n, e)$ agrees with the function of Lemma 10.5.2 and if S_e is not dense then for some n, for all $m \geq n$, $g(m, e) \uparrow$. This is the result of assuming that the answer to each $\emptyset'$ query is "yes"; at some point $g(m, e)$ will fail to halt if S_e is not dense.

Theorem 10.7.1. *If I is hyperimmune, then I is the indifferent set for some weakly 1-generic set G with $G \leq_T I$.*

Proof. We will construct $G \leq_T I$ and meet the following requirements to ensure that G is weakly 1-generic.

R_e: If S_e is dense, then $\exists n\, [(G \upharpoonright n)g(n, e) \prec G$ and $[n, n + g(n, e)] \cap I = \emptyset]$.

We order our requirements by priority as follows: $R_0 > R_1 > \ldots$. We say that requirement R_e needs attention at stage $s+1$ if:

(i). $e \leq s$.

(ii). R_e is not currently satisfied.

(iii). $g(s,e)[p_I(s)]\downarrow$.

(iv). $I \cap [s, s+|g(s,e)|] = \emptyset$.

(v). $s+1$ is not restrained by any R_d with $d < e$.

At stage 0, we set $G_0 = 0^\omega$. At stage $s+1$, if no requirement needs attention then we set $G_{s+1} = G_s$. Otherwise let R_e be the highest priority requirement that needs attention. We define $G_{s+1} = (G_s \upharpoonright s)(g(s,e))0^\omega$ and if $x \leq s + |g(s,e)|$, we restrain x with priority e. We declare R_e satisfied and R_f unsatisfied for all $f > e$.

Verification. Because for all x, $G_x \upharpoonright x = G \upharpoonright x$, we have that $G \leq_T I$. To show that all requirements are met, consider any requirement R_e and let $s_0 \geq e$ be a stage such that for all $d \leq e$, R_d no longer requires attention and further that no $x \geq s_0$ is restrained by a higher priority requirement. Now if S_e is not dense then R_e is met trivially but further for some x, for all $y \geq x$, $g(y,e)\uparrow$ and so R_e only requires attention finitely often.

If S_e is dense then we claim that there exists some $t \geq s_0$ such that both $g(t,e)[p_I(t)]\downarrow$, and $I \cap [t, t+|g(t,e)|] = \emptyset$. If this claim holds then we have that $(G \upharpoonright t)g(t,e) \prec G$ and so requirement R_e is met. It needs attention finitely often because once it is acted on it will be declared satisfied and never again require attention.

To establish the claim, define the following computable function $h : \omega \to \omega$. To determine $h(x)$, let $n_{x,0} = x$ and $n_{x,i+1} = n_{x,i} + |g(n_{x,i}, e)| + 1$. Now define $h(x)$ to be the least stage such that for all i with $0 \leq i \leq x+1$, $g(n_{x,i}, e)$ has halted. We also require $h(x)$ to be greater than $n_{x,x+1}$. If we take some $x \geq s_0$ such $p_I(x) > h(x)$, then there must be some $i \leq x$ such that $[n_{x,i}, n_{x,i+1}-1] \cap I = \emptyset$. Further we have that $g(n_{x,i}, e)[p_I(x)]\downarrow$ so $g(n_{x,i}, e)[p_I(n_{x,i})]\downarrow$. Now take $t = n_{x,i}$. □

Corollary 10.7.2. *A set $I \subseteq \omega$ is the indifferent set for some weakly 1-generic set if and only if I is hyperimmune.*

Proof. The other direction, the fact that if I is an indifferent set for some weakly 1-generic set G then I is hyperimmune, follows from the proof of Theorem 10.5.1. The proof of Theorem 10.5.1 builds a dense c.e. set of strings so this proof applies to the case of weakly 1-generic sets as well. □

We will now consider which degrees can compute an indifferent set for a given weakly 1-generic set. In Lemma 10.5.2, we showed that we could make a set G meet or avoid S_e by ensuring that $(G \upharpoonright n)g(n,e) \prec G$. Now if S_e is dense,

then as the following lemma indicates, this is also a necessary condition for G to meet S_e.

Lemma 10.7.3. *If G is a weakly 1-generic set and S_e is dense, then there exist infinitely many n such that $(G \upharpoonright n)g(n,e) \prec G$.*

Proof. Given a dense set of strings S_e we will define another dense set of strings V_e as follows. At stage 0 we add $g(0,e)$ to V_e and set $l_0 = |g(0,e)|$. At stage $s+1$, for all σ of length l_s, we add $\sigma g(l_s,e)$ to V_e and set $l_{s+1} = l_s + |g(l_s,e)|$. As V_e is dense, it follows that G meets V_e infinitely often. Hence for infinitely many i, $(G \upharpoonright l_i)g(l_i,e) \prec G$. □

Theorem 10.7.4. *If G is a weakly 1-generic set, and I is a set such that p_I escapes domination by all $f \leq_T G$, then I is an indifferent set for G.*

Proof. If G is a weakly 1-generic set, and S_e is dense, there is a sequence $n_0, n_1, \ldots$ computable in G such that $(G \upharpoonright n_i)g(n_i,e) \prec G$. Further we can require that $n_{i+1} > n_i + |g(n_i+1,e)|$. Now as p_I escapes domination by any G-computable function, for some x we have that $p_I(x) > n_{x+1}$. Thus there is some $i \leq x$ such that such that $I \cap [n_i, n_{i+1} - 1] = \emptyset$. Thus we have that for all $X \subseteq I$, $G_{[X]} \upharpoonright n_i g(n_i,e) \prec G_{[X]}$ and so $G_{[X]}$ meets S_e. This is true for any e so I is an indifferent set for G with respect to weak 1-genericity. □

Theorem 10.7.5. *If G is a weakly 1-generic set, $A \geq_T G$ and $A \in \overline{\mathrm{GL}_2}$, then A computes an indifferent set for G.*

Proof. For the previous proof to hold, we needed p_I to escape domination by a set of functions f_e for each e such that S_e is dense. From $G \oplus \emptyset'$ we can define a function that majorizes $f(x) = \max\{f_e(x) : e \leq x \wedge S_e \text{ is dense}\}$. If a set S_e is not dense, then the corresponding f_e may fail to be total. However, we can concurrently attempt to compute $f_e(x)$, for any x, using G and search for a witness that shows S_e is not dense using $\emptyset'$. Either $f_e(x)\downarrow$ or we establish that S_e is not dense and we can remove e from our list of functions.

As $A \in \overline{\mathrm{GL}_2}$ and $A \geq_T G$, we have that A computes a set whose principal function escapes domination by f and hence A computes an indifferent set for G. □

We will now look at another significant difference between indifferent sets for 1-generic sets and weakly 1-generic sets.

Theorem 10.7.6. *Any Δ^0_2 weakly 1-generic set computes a set it is indifferent to.*

Proof. Let $G \leq_T \emptyset'$ be weakly 1-generic set with approximation G_s. To prove this theorem we will construct a Turing functional Γ such that $\Gamma(G)$ is total and strictly increasing and that rng$\Gamma(G)$ is an indifferent set for G.

Take any $X \subseteq$ rng$\Gamma(G)$. We will use the fact that G is weakly 1-generic to ensure that $G_{[X]}$ is also weakly 1-generic. For each c.e. set of strings S, we will enumerate our own c.e. set of strings V. If S is dense, then we will threaten to make V dense as well. Further we will construct V in such a way that if G meets it, then for all $X \subseteq$ rng$\Gamma(G)$, $G_{[X]}$ meets S. Our requirements are as follows:

I_e: $\quad \Gamma(G; e+1) \downarrow > \Gamma(G; e)$,

R_e: $\quad S_e$ dense $\rightarrow \exists[n([n, n + |g(n, e)|] \cap$ rng$\Gamma(G) = \emptyset$, and $(G \upharpoonright n)g(n, e) \prec G)]$.

A requirement I_e needs attention at stage s, if $\Gamma(G; e+1)[s] \uparrow$. We act on such a requirement by finding the least $x > \Gamma(G; e)[s]$ that is not restrained by a higher priority requirement. We set $\Gamma(G_s; e+1)[s+1] = x$ with use x. We injure all lower priority requirements of the form R_e.

We will assign each requirement R_e, a c.e. set of strings V_e. Each time R_e is injured we assign R_e a new set V_e. We will define the following computable function $m : \omega^3 \rightarrow \omega$ by

$$m(e, i, s) = \max\{x + 1 : (\exists \rho \in 2^{<\omega})(x \in \text{rng}\Gamma(\rho)[s] \wedge \rho \not\succeq G_s \upharpoonright i)\}.$$

A requirement R_e needs attention if at stage $s + 1$ if

(i). G_s does not meet V_e.

(ii). For some unused pair (e, i), there exists an s such that $g(m(e, i, s), e)[s] \downarrow$.

We act on a requirement R_e by adding $\sigma g(m(e, i, s), e)$ to V_e for all strings σ such that:

(i). $|\sigma| = m(e, i, s)$.

(ii). $\sigma \not\succeq G_s \upharpoonright i$.

Following this, we injure all lower priority requirements and we restrain any $x \leq m(e, i, s) + g(m(e, i, s), e)$ with priority e. We declare the pair (e, i) used.

The construction is simply this, at stage s we find the highest priority requirement that needs attention and we act on that requirement.

Verification. Take any requirement, and assume that all higher priority requirements require attention finitely often. If the requirement is of the form I_e then there is some maximum restraint r placed on I_e by higher priority requirements. Let x be the least value that exceeds r and, if $e \neq 0$, also exceeds $\Gamma(G; e-1)$. The construction ensures that $\Gamma(G; e) = x$ with use x and hence I_e is met and further, I_e requires attention finitely often.

If the requirement is of the form R_e, then first assume that S_e is not a dense c.e. set of strings. In this case R_e is met trivially. Further, R_e cannot need attention infinitely often because for some x for all $y \geq x$, $g(y, e) \uparrow$. Now assume that S_e is a dense c.e. set of strings. First we claim that for all i, there exists some s such that $g(m(e, i, s), e)[s] \downarrow$. This claim holds because during the construction we only define Γ on initial segments of G_s. Once our approximation settles on the first i bits, then we will no longer define Γ on strings that do not extend $G \upharpoonright i$. This means that $\lim_s m(e, i, s)$ exists and so for some s, $g(m(e, i, s), e)[s] \downarrow$.

If G did not meet V_e then R_e would act infinitely often and so V_e would be dense. However as G is weakly 1-generic then this would imply that G does meet V_e. Hence we must conclude that at some point G meets V_e and so R_e requires attention finitely often.

It remains to show that requirement R_e is satisfied. Let $\sigma g(m(e, i, s), e)$ be the string in V_e that G meets, where $|\sigma| = m(e, i, s)$. At stage s, we had that $\sigma \neq G_s \upharpoonright i$ so $G \not\succ G_s \upharpoonright i$. Further for all x such that

$$m(e, i, s) < x \leq m(e, i, s) + g(m(e, i, s), e)$$

we have that $x \notin \text{rng}\Gamma(\rho)[s]$ if $\rho \not\succ G_s \upharpoonright i$. Thus for all $x \in [m(e, i, s), m(e, i, s) + g(m(e, i, s), e)]$, $x \notin \text{rng}\Gamma(G)$ because after stage s all $x \leq g(m(e, i, s), e)$ are restrained with priority e.

□

10.8 Open questions

We conclude this chapter with some remaining questions.

Question 10.8.1. Does there exists an array computable set A, such that A computes a 1-generic set G and an indifferent set for G? If so, can A be 1-generic? If this question has a positive answer it may well provide a new example of a degree with no strong minimal cover.

Question 10.8.2. In Corollary 10.7.2 we characterised indifferent sets for weakly 1-generic sets in terms of sparseness. Can indifferent sets for 1-generic sets be characterised by a sparseness requirement?

Question 10.8.3. Can the c.e. degrees that bound 1-generic sets that compute an indifferent set for themselves be characterised using the totally α-c.e. degree hierarchy?

Question 10.8.4. In Section 10.7 we showed that any weakly 1-generic set that is either in $\overline{\mathrm{GL}_2}$ or is Δ^0_2 can compute an indifferent set for itself. However, we have not be able to answer the following question. Does there exists a weakly 1-generic set that cannot compute a set to which it is indifferent?

Question 10.8.5. What can be said about indifferent sets for (weakly) n-generic sets for $n > 1$?

Bibliography

[1] ALLENDER, E., BUHRMAN, H., AND KOUCKÝ, M. What can be efficiently reduced to the Kolmogorov-random strings? *Annals of Pure and Applied Logic 138* (2006), 2–19.

[2] ALLENDER, E., BUHRMAN, H., KOUCKÝ, M., VAN MELKEBEEK, D., AND RONNEBURGER, D. Power from random strings. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 669–678.

[3] ARSLANOV, M. M. On some generalizations of the fixed-point theorem. *Soviet Mathematics 25* (1981), 1–10.

[4] BARMPALIAS, G. Computably enumerable sets in the Solovay and the strong weak truth table degrees. In *New Computational Paradigms* (2005), S. Cooper, B. Löwe, and L. Torenvliet, Eds., Lecture Notes in Computer Science, Springer Berlin/Heidelberg, pp. 8–17.

[5] BARMPALIAS, G., DOWNEY, R. G., AND GREENBERG, N. Working with strong reducibilities above totally ω-c.e. and array computable degrees. *Transactions of the American Mathematical Society 362*, 2 (2010), 777–813.

[6] BARMPALIAS, G., AND LEWIS, A. E. M. A c.e. real that cannot be sw-computed by any Ω number. *Notre Dame Journal of Formal Logic 47*, 2 (2006), 197–209.

[7] BARMPALIAS, G., AND LEWIS, A. E. M. The ibT degrees of computably enumerable sets are not dense. *Annals of Pure and Applied Logic 141*, 1-2 (2006), 51–60.

[8] BARTLE, R. G. *The Elements of Integration*. John Wiley & Sons, 1966.

[9] BILLINGSLEY, P. *Convergence of Probability Measures*, 2nd ed. Wiley, 1999.

[10] CAI, M., AND SHORE, R. A. Domination, forcing, array nonrecursiveness and relative recursive enumerability. *To appear*.

[11] CALHOUN, W. C. Degrees of monotone complexity. *The Journal of Symbolic Logic 71* (2006), 1327–1341.

[12] CALUDE, C., COLES, R. J., HERTLING, P., AND KHOUSSAINOV, B. Degree-theoretic aspects of computably enumerable reals. In *Models and Computability. Invited Papers from Logic Colloquium '97 - European Meeting of the Association for Symbolic Logic, Leeds, July 1997*, S. B. Cooper and J. K. Truss, Eds., vol. 259 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1999.

[13] CALUDE, C., HERTLING, P., KHOUSSAINOV, B., AND WANG, Y. Recursively enumerable reals and Chaitin Ω numbers. In *STACS 98. 15th Annual Symposium on Theoretical Aspects of Computer Science. Paris, France, February 25-27, 1998* (1998), M. Morvan, C. Meinel, and D. Krob, Eds., vol. 1373 of *Lecture Notes in Computer Science*, pp. 596–606.

[14] CHAITIN, G. J. On the length of programs for computing finite binary sequences. *Journal of the ACM 13* (1966), 547–569.

[15] CHAITIN, G. J. On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the ACM 16* (1969), 145–159.

[16] CHAITIN, G. J. A theory of program size formally identical to information theory. *Journal of the ACM 22*, 3 (1975), 329–340.

[17] CHAITIN, G. J. Information-theoretic characterizations of recursive infinite strings. *Theoretical Computer Science 2*, 1 (1976), 45 – 48.

[18] CHURCH, A. A note on the Entscheidungsproblem. *The Journal of Symbolic Logic 1* (1936), 40–41.

[19] CHURCH, A. An unsolvable problem of elementary number theory. *American Journal of Mathematics 58* (1936), 345–363.

[20] CHURCH, A. On the concept of a random sequence. *Bulletin of the American Mathematical Society 46* (1940), 130–135.

[21] CSIMA, B. F., AND SOARE, R. I. Computability results used in differential geometry. *The Journal of Symbolic Logic 71*, 4 (2006), 1394–1410.

[22] DAY, A. R. On the computational power of random strings. *Annals of Pure and Applied Logic 160*, 2 (2009), 214–228.

[23] DAY, A. R. The computable Lipschitz degrees of computably enumerable sets are not dense. *Annals of Pure and Applied Logic 161*, 12 (2010), 1588 – 1602.

[24] DAY, A. R. On process complexity. *Chicago Journal of Theoretical Computer Science 4* (2010).

[25] DEMUTH, O. Remarks on the structure of tt-degrees based on constructive measure theory. *Commentationes Mathematicae Universitatis Carolinae 29*, 2 (1988), 233–247.

[26] DOWNEY, R. G., AND GREENBERG, N. A transfinite hierarchy of notions of lowness in the computably enumerable degrees, unifying classes, and natural definability. *To appear*.

[27] DOWNEY, R. G., GREENBERG, N., AND WEBER, R. Totally ω-computably enumerable degrees and bounding critical triples. *Journal of Mathematical Logic 7*, 2 (2007), 145–171.

[28] DOWNEY, R. G., AND GRIFFITHS, E. J. Schnorr randomness. *The Journal of Symbolic Logic 69*, 2 (2004), 533–554.

[29] DOWNEY, R. G., GRIFFITHS, E. J., AND REID, S. On Kurtz randomness. *Theoretical Computer Science 321*, 2-3 (2004), 249–270.

[30] DOWNEY, R. G., AND HIRSCHFELDT, D. R. *Algorithmic Randomness and Complexity*. Springer-Verlag, 2010.

[31] DOWNEY, R. G., HIRSCHFELDT, D. R., AND LAFORTE, G. Randomness and reducibility. *Journal of Computer and System Sciences 68*, 1 (2004), 96–114.

[32] DOWNEY, R. G., JOCKUSCH, JR., C. G., AND STOB, M. Array nonrecursive sets and multiple permitting arguments. In *Recursion theory week (Oberwolfach, 1989)*, vol. 1432 of *Lecture Notes in Mathematics*. Springer, Berlin, 1990, pp. 141–173.

[33] DOWNEY, R. G., JOCKUSCH, JR., C. G., AND STOB, M. Array nonrecursive degrees and genericity. In *Computability, enumerability, unsolvability*, vol. 224 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1996, pp. 93–104.

[34] FIGUEIRA, S., MILLER, J. S., AND NIES, A. Indifferent sets. *Journal of Logic and Computation 19*, 2 (2009), 425–443.

[35] GÁCS, P. On the symmetry of algorithmic information. *Soviet Mathematics Doklady 15* (1974), 1477–1480.

[36] GÁCS, P. On the relation between descriptional complexity and algorithmic probability. In *22nd Annual Symposium on Foundations of Computer Science* (Los Alamitos, CA, USA, 1981), IEEE Computer Society, pp. 296–303.

[37] GÁCS, P. On the relation between descriptional complexity and algorithmic probability. *Theoretical Computer Science 22* (1983), 71–93.

[38] GÁCS, P. Uniform test of algorithmic randomness over a general space. *Theoretical Computer Science 341*, 1-3 (2005), 91–137.

[39] HIRSCHFELDT, D. R., NIES, A., AND STEPHAN, F. Using random sets as oracles. *Journal of the London Mathematical Society 75*, 3 (2007), 610–622.

[40] HOYRUP, M., AND ROJAS, C. Computability of probability measures and Martin-Löf randomness over metric spaces. *Information and Computation 207*, 7 (2009), 830–847.

[41] ISHMUKHAMETOV, S. Weak recursive degrees and a problem of Spector. In *Recursion theory and complexity (Kazan, 1997)*, M. M. Arslanov and S. Lempp, Eds., vol. 2 of *de Gruyter Series in Logic and its Applications*. de Gruyter, Berlin, 1999, pp. 81–88.

[42] JOCKUSCH, JR., C. G. Degrees of generic sets. In *Recursion theory: its generalisation and applications*, vol. 45 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1980, pp. 110–139.

[43] JOCKUSCH, JR., C. G., AND POSNER, D. B. Double jumps of minimal degrees. *The Journal of Symbolic Logic 43*, 4 (1978), 715–724.

[44] JOCKUSCH, JR., C. G., AND SOARE, R. I. Π^0_1 classes and degrees of theories. *Transactions of the American Mathematical Society 173* (1972), 33–56.

[45] KAKUTANI, S. A generalization of Brouwer's fixed point theorem. *Duke Mathematical Journal 8* (1941), 457–459.

[46] KOLMOGOROV, A. N. Three approaches to the quantitiative definition of information. *Problems of Information Transmission 1* (1965), 1–7.

[47] KRAFT, L. G. A device for quantizing, grouping and coding amplitude modulated pulses. Master's thesis, MIT, 1949.

[48] KUMABE, M. *On the Turing Degrees of Generic Sets*. PhD thesis, University of Chicago, 1990.

[49] KUMABE, M. Degrees of generic sets. In *Computability, enumerability, unsolvability*, vol. 224 of *London Mathematical Society Lecture Note Series*. Cambridge Univ. Press, Cambridge, 1996, pp. 167–183.

[50] KUMMER, M. On the complexity of random strings (extended abstract). In *STACS '96: Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science* (London, UK, 1996), Springer-Verlag, pp. 25–36.

[51] KURTZ, S. *Randomness and genericity in the degrees of unsolvability*. PhD thesis, University of Illinois at Urbana–Champaign, 1981.

[52] KUČERA, A. Measure, Π^0_1 classes and complete extensions of PA. In *Recursion Theory Week. Proceedings of the Conference Held at the Mathematisches Forschungsinstitut in Oberwolfach, April 15-21, 1984* (Berlin, 1985), H. D. Ebbinghaus, G. H. Müller, and G. E. Sacks, Eds., vol. 1141 of *Lecture Notes in Mathematics*, Springer, pp. 245–259.

[53] LACOMBE, D. Quelques procédés de définition en topologie recursive. In *Constructivity in mathematics: Proceedings of the colloquium held at Amsterdam, 1957 (edited by A. Heyting)* (1959), Studies in Logic and the Foundations of Mathematics, North-Holland Publishing Co., pp. 129–158.

[54] LADNER, R. E., AND SASSO, L. P. The weak truth table degrees of recursively enumerable sets. *Annals of Mathematical Logic 8*, 4 (1975), 429–448.

[55] LEVIN, L. A. On the notion of a random sequence. *Soviet Mathematics Doklady 14*, 5 (1973), 1413–1416.

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

[57] LEVIN, L. A. Uniform tests of randomness. *Soviet Mathematics Doklady 17*, 2 (1976), 337–339.

[58] LEVIN, L. A. Some theorems on the algorithmic approach to probability theory and information theory: (1971 dissertation directed by A. N. Kolmogorov). *Annals of Pure and Applied Logic 162*, 3 (2010), 224 – 235. Special Issue: Dedicated to Nikolai Alexandrovich Shanin on the occasion of his 90th birthday.

[59] LEWIS, A. E. M., AND BARMPALIAS, G. Random reals and Lipschitz continuity. *Mathematical Structures in Computer Science 16*, 5 (2006), 737–749.

[60] LI, M., AND VITÁNYI, P. *An introduction to Kolmogorov complexity and its applications (2nd ed.)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[61] LUTZ, J. H. The quantitative structure of exponential time. In *Complexity Theory Retrospective II*. Springer, 1997, pp. 225–260.

[62] MARTIN, D. A. Classes of recursively enumerable sets and degrees of unsolvability. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 12* (1966), 295–310.

[63] MARTIN-LÖF, P. The definition of random sequences. *Information and Control 9* (1966), 602–619.

[64] MILLER, J. S. Degrees of unsolvability of continuous functions. *The Journal of Symbolic Logic 69*, 2 (2004), 555–584.

[65] MILLER, J. S., AND YU, L. On initial segment complexity and degrees of randomness. *Transactions of the American Mathematical Society 360*, 6 (2008), 3193–3210.

[66] MUCHNIK, A. A., AND POSITSELSKY, S. Y. Kolmogorov entropy in the context of computability theory. *Theoretical Computer Science 271*, 1-2 (2002), 15–35.

[67] MUNROE, M. E. *Introduction to Measure and Integration*. Addison-Wesley, 1953.

[68] NERODE, A. General topology and partial recursive functions. In *Summaries of talks presented at the Summer Institute for Symbolic Logic*. Cornell University, 1957, pp. 247–251.

[69] NIES, A. Lowness properties and randomness. *Advances in Mathematics 197*, 1 (2005), 274 – 305.

[70] NIES, A. *Computability and randomness*. Oxford University Press, 2009.

[71] ODIFREDDI, P. *Classical Recursion Theory. The Theory of Functions and Sets of Natural Numbers*, vol. 125 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, Amsterdam, 1990.

[72] REIMANN, J. Effectively closed sets of measures and randomness. *Annals of Pure and Applied Logic 156*, 1 (2008), 170–182.

[73] REIMANN, J., AND SLAMAN, T. A. Measures and their random reals. *To appear*.

[74] REIMANN, J., AND SLAMAN, T. A. Probability measures and effective randomness. In *13th International Congress of Logic Methodology and Philosophy of Science, Beijing* (2007).

[75] ROGERS, H. *Theory of Recursive Functions and Effective Computability*. The MIT Press, 1987.

[76] ROZINAS, M. The semilattice of e-degrees. In *Recursive functions (Russian)*. Ivanov. Gos. Univ., Ivanovo, 1978, pp. 71–84.

[77] SACKS, G. E. The recursively enumerable degrees are dense. *The Annals of Mathematics 80*, 2 (1964), 300–312.

[78] SCHNORR, C. P. A unified approach to the definition of random sequences. *Theory of Computing Systems 5*, 3 (1971), 246–258.

[79] SCHNORR, C. P. Zufälligkeit und Wahrscheinlichkeit. In *Lecture Notes in Mathematics*, vol. 218 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin-New York, 1971.

[80] SCHNORR, C. P. Process complexity and effective random test. *Journal of Computer and System Sciences 7* (1973), 376–388.

[81] SELMAN, A. L. Arithmetical reducibilities. I. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 17* (1971), 335–350.

[82] SIMPSON, S. Degrees of unsolvability: a survey of results. In *Handbook of Mathematical Logic*, J. Barwise, Ed., vol. 90 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, Amsterdam, 1977, pp. 631–652.

[83] SOARE, R. I. *Recursively enumerable sets and degrees*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[84] SOLOMONOFF, R. J. A formal theory of inductive inference, I and II. *Information and Control 7*, 1-22 (1964), 224–254.

[85] TURING, A. M. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society s2-42*, 1 (1937), 230–265.

[86] TURING, A. M. On computable numbers, with an application to the Entscheidungsproblem. A correction. *Proceedings of the London Mathematical Society s2-43*, 1 (1938), 544–546.

[87] USPENSKY, V. A., AND SHEN, A. Relations between varieties of Kolmogorov complexities. *Mathematical Systems Theory 29* (1996), 271–292.

[88] VAN LAMBALGEN, M. *Random Sequences*. PhD thesis, University of Amsterdam, 1987.

[89] VILLE, J. Étude critique de la notion de collectif. In *Monographies des Probabilités. Calcul des Probabilités et ses Applications*. Gauthier-Villars, Paris, 1939.

[90] VON MISES, R. Grundlagen der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift 5*, 52-99 (1919).

[91] WANG, Y. *Randomness and Complexity*. PhD thesis, University of Heidelberg, 1996.

[92] WEIHRAUCH, K. *Computable analysis, an introduction*. Springer-Verlag, Berlin, 2000.

[93] YU, L., AND DING, D. There is no sw-complete c.e. real. *The Journal of Symbolic Logic 69*, 4 (2004), 1163–1170.

[94] ZVONKIN, A. K., AND LEVIN, L. A. The complexity of finite objects and the development of the concepts of information and randomness of means of the theory of algorithms. *Russian Mathematical Surveys 25*, 6 (1970), 83–124.