COMPUTABILITY, TRACEABILITY AND BEYOND

by

Keng Meng Ng

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

Victoria University of Wellington

July 2009

ABSTRACT

This thesis is concerned with the interaction between computability and randomness. In the first part, we study the notion of traceability. This combinatorial notion has an increasing influence in the study of algorithmic randomness. We prove a separation result about the bounds on jump traceability, and show that the index set of the strongly jump traceable computably enumerable (c.e.) sets is Π^0_4-complete. This shows that the problem of deciding if a c.e. set is strongly jump traceable, is as hard as it can be. We define a strengthening of strong jump traceability, called hyper jump traceability, and prove some interesting results about this new class. Despite the fact that the hyper jump traceable sets have their origins in algorithmic randomness, we are able to show that they are natural examples of several Turing degree theoretic properties. For instance, we show that the hyper jump traceable sets are the first example of a lowness class with no promptly simple members. We also study the dual highness notions obtained from strong jump traceability, and explore their degree theoretic properties.

In the second part we investigate the degree theoretic aspects of different classes arising in algorithmic randomness. We show that every PA degree is the join of two random degrees. We also study the Turing degrees with effective packing dimension one. In particular, we show that these degrees are not as well-behaved as they were initially conjectured. We define a weak notion of Martin-Löf randomness, and characterize the c.e. degrees which contain such randoms. This is the first time a class of random reals is characterized using multiple permitting arguments – the latter arose in classical degree theory in connection with lattice embeddings. Lastly we investigate the sets which are low for Demuth randomness, and show that every such set is hyperimmunefree.

In the third part we explore the structure of the c.e. Turing degrees. We investigate which c.e. degrees can be split into smaller ones with low complexity. For instance we construct a c.e. degree which cannot be split into two superlow c.e. degrees. This highlights the fact that the low and superlow c.e. degrees are very different. We also prove some results about cupping classes: we show that the low$_2$-

cuppable sets are exactly the c.e. traceable-cuppable sets. We refute a conjecture of Li on the cuppable sets, by constructing a cuppable c.e. set which can only be cupped with high sets. We present some results on strong tabular reducibilities. In particular, we show that the truth table analogue and the weak truth table analogue of two classical jump inversion theorems fail. Finally, we study the Turing degrees of diagonal sets. We answer a question of Kummer and show that the semi-maximal and semi-hyperhypersimple degrees do not coincide.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Rodney Downey, for his wonderful guidance and support. He has taught me mathematics, dancing, and most importantly has been a wonderful friend throughout my stay in New Zealand. I would like to thank my co-authors for allowing me to include the work we did together in this thesis. Special thanks go out to the logic group in Wellington, for the help and support they have given me. These include Noam Greenberg, André Nies, George Barmpalias, and Laurent Bienvenu. I would like to thank Chong Chi Tat, Yang Yue and Wu Guohua for introducing me to logic. I also want to mention the people who have supported me during my visit to North America – Paul Brodhead, Douglas Cenzer, Peter Cholak, Barbara Csima, Denis Hirschfeldt, Bart Kastermans, Steffen Lempp, Antonio Montalbán, Joe Miller and Robert I. Soare. I would also like to thank my family and friends for their support. Last, but not least, I would like to thank my wife who has done more for me than any word of gratitude can express.

Contents

Chapter 1

General Introduction

A fundamental notion in the study of computability theory is *relative computability.* Given two sets of numbers A and B, we say that A is computable from B (or A is Turing reducible to B), if there is a computable process which given any input n, decides in a finite number of steps if n is in A. During the computation process we allow queries about the membership of B, and B is known as the *oracle.* We denote "A is Turing reducible to B" by $A \leq_T B$. If A can be computed from B, then A is intuitively less complex than B. A set is computable if it can be computed without any queries to the oracle. This reducibility forms a preordering on $\mathcal{P}(\mathbb{N})$. The equivalence relation $A \equiv_T B \Leftrightarrow A \leq_T B \wedge B \leq_T A$ is called Turing equivalence, and the equivalence classes under $\equiv_T$ are called *Turing degrees.*

Another fundamental concept in computability theory is *computable enumerability.* A set A is computably enumerable (c.e.) if there is a computable procedure which enumerates its members (not necessarily in order of magnitude). If a Turing degree contains a c.e. set, we call it a c.e. degree. The most famous c.e. set is the Halting problem $\emptyset' = \{e \in \mathbb{N} \mid \varphi_e(e) \text{ converges}\}$. Here $\varphi_e(x)$ is the e^{th} computer program running on input x; if $\varphi_e(x)$ converges we denote it as $\varphi_e(x) \downarrow$, otherwise we write $\varphi_e(x) \uparrow$. Computably enumerable sets arise naturally in the study of proofs. The process of generating proofs is an enumerable process. For instance the set of the first order consequences of any computable set of axioms is a c.e. set. Computably enumerable sets form the foundation of the topics we are going to explore in this thesis, namely traceability, algorithmic randomness and Turing degrees.

A main motivation for this thesis comes from the interaction between algorithmic randomness and computability. In particular we study the concept of traceability, which has been shown to be instrumental in linking algorithmic randomness and computability. Since the early days of computability theory, the relation between *descriptive complexity*, and *computational complexity* has been a recurring theme in the subject. The former is a syntatic notion which talks about how hard it is to describe, or to define a certain property. The latter is about the computational nature of a class of reals, i.e. how much computational power a class of reals has as measured by Turing machines. One of the first results in computability theory demonstrating this relation is the fact that the partial recursive functions (due to Kleene) are exactly the class of functions computable using a Turing machine. The former class has a definition we can write down explicitly, while the latter class can be evaluated by computer programs.

A theorem of Post, known as Post's Theorem again demonstrated this relationship. A set A is Σ^0_n if $A = \{z \mid \exists x_1 \forall x_2 \exists x_3 \cdots Q x_n R(z, x_1, \cdots, x_n)\}$, where R is a computable relation and Q is $\exists$ if n is odd, and $\forall$ if n is even. A set A is Π^0_n if $\bar{A}$ is Σ^0_n, and A is Δ^0_n if it is both Σ^0_n and Π^0_n. Post's Theorem says that a set A is Δ_{n+1} if and only if $A \leq_T \emptyset^{(n)}$, where $\emptyset^{(n)}$ is the Halting problem iterated $n-1$ times. Again this relates a definability property with a computational one. Our aim in this thesis is to examine the relationship between descriptive, and computational properties in connection with algorithmic randomness. An important concept is traceability, a combinatorial property originating from set theory. This has become a fundamental notion in the study of randomness, because it can be used to define classes with low complexity. Our research aims to reconcile the three notions: *computational complexity* as measured by Turing machines and algorithms, *traceability* which is a property describing the simplicity of a class of reals, and their connections with *algorithmic randomness.*

1.1 Randomness and traceability

Some of the intriguing questions which define our research are: how do we measure if a real is random? From a computability theoretic point of view, how much information can a random real contain? What can be said of the reals which are far away from being random? What is the relationship between how close a real is to being random, and its computational strength? In the first two parts of this thesis we will address these questions, and contribute to the understanding of the relationship between computability and randomness.

How do we calibrate randomness for infinite binary strings? Suppose we are given two infinite binary strings, say a sequence of 1's $11111111111\cdots$, and a sequence obtained by a fair coin toss $01001110110010\cdots$. In terms of probability theory, these two events are equally likely. However most people will agree that the sequence of coin tosses appear to be more random than the sequence of 1's. This is due to the fact that the sequence of 1's follows a simple "law" which we can describe easily, while the sequence of coin tosses is rather chaotic. It is a fundamental problem to transfer this intuition into a precise mathematical notion where we can separate the random (or chaotic) strings from the ones which obey certain simple rules. von Mises [vM19] was the first to address this problem in the early 20^{th} century. He proposed to define a random real as one which obeys all reasonable statistical tests. For instance, the number of zeroes and ones in the string should be asymtotically equal (this of course rules out the sequence of 1's from being random). He talked about "acceptable selection rules", and at that time he was unable to decide on a formal notion for acceptability.

It was only later, with the development of modern computability theory that this approach became clear. Martin-Löf [ML66] suggested that the acceptable selection rules could be taken as effectively presented sets of measure zero. Martin-Löf defined randomness by looking at the ability to "cover a string effectively" with Σ^0_1 open sets. A ML-test is a sequence of c.e. sets $\{W_{h(x)} : x \in \mathbb{N}\}$ indexed by a computable function h, such that for all x, $\mu(W_{h(x)}) < 2^{-x}$. Here, μ denotes the Lebesgue measure and $W_{h(x)}$ is viewed as a set of finite strings. Martin-Löf defined [ML66] a set to be Martin-Löf random (or simply ML-random), if it does not belong to $\cap_x W_{h(x)}$ for any ML-test $\{W_{h(x)} : x \in \mathbb{N}\}$. For instance it is easy to see that no computable set (for instance, the sequence of 1's) can be random. Martin-Löf 's approach was one of the three basic ways of calibrating randomness. It was based on the fact that a random real should not behave in a typical manner, and no effective statistical process should be able to test for it. This is sometimes called the *statistician's approach.*

Another approach to calibrating randomness is the *coder's approach.* This is based on the intuition that a random string should have initial segments which can cannot be compressed very much. For instance, a sequence of n 1's can be easily described with very little information, but it is difficult to find a pattern for the first n bits of the random coin toss. We need a way to measure the information content of a finite string. Again ideas from computability theory were essential in formalizing this notion.

Given two finite strings σ and τ, we say that τ is an M-description of σ if $M(\tau)$ converges and outputs σ, where M is a Turing machine. We can remove the dependence on the machine M, by considering a universal machine U. A finite binary string σ is said to have *plain Kolmogorov complexity* n, if the shortest length of a U-description of σ is n. This is denoted as $C(\sigma) = n$. This idea of looking at a system of descriptions was due to Solomonoff, and independently by Kolmogorov. A machine M_P is *prefix-free*, if the domain of M_P is an antichain with respect to string extension $\subset$. Similarly we say that σ has prefix-free complexity n, written as $K(\sigma) = n$, if the length of a shortest U_P-description of σ is n. Here U_P is the universal prefix-free machine. A string is random if $\forall n(K(A\upharpoonright n) \geq^+ n)$, where we write $f(n) \leq^+ g(n)$ if there is a constant c such that $f(n) \leq g(n) + c$ for every n. This approach to randomness appeared in the work of Levin [Lev73], Schnorr [Sch73] and Chaitin [Cha75].

It turns out that both definitions of randomness are equivalent. A third equivalent definition of Martin-Löf randomness, is via c.e. *martingales*, which can be thought of as a betting strategy. A set is random if one cannot win when betting against the set using any c.e. martingales. It is not surprising that this is sometimes dubbed the *gambler's approach.* Each of these three characterizations of Martin-Löf randomness expresses the intuitive notion of being "random" in some natural way. This makes ML-randomness the most natural choice for studying effective randomness.

The canonical example of a Martin-Löf random real is Chaitin's halting probability $\Omega = \sum_{U(\sigma)\downarrow} 2^{-|\sigma|}$. Variations on the effectivity of the tests have been suggested. These variations give rise to different notions of strong randomness and weak randomness in literature. Notions weaker than Martin-Löf randomness include *Schnorr randomness* where a real is Schnorr random if it does not belong to any Schnorr test.

Here a Schnorr test is a Martin-Löf test $\{U_i : i \in \mathbb{N}\}$ with $\mu(U_i) = 2^{-i}$. Another weak notion is *Kurtz randomness*, or weakly random as it is sometimes known, where the tests are of the form $\{D_i : i \in \mathbb{N}\}$, where $D_i =$ the finite set encoded by $f(i)$ for some computable function f.

Notions stronger than Martin-Löf random have also been studied. A popular variant is *weak 2-randomness* where the tests are of the form $\{U_i : i \in \mathbb{N}\}$, and $\mu(U_i) \to 0$. Another variant is *2-randomness* where a set is called 2-random if it is random relative to $\emptyset'$. The above notions of strong randomness are computationally weak, in that they form minimal pairs with $\emptyset'$. Two degrees $\boldsymbol{a}$ and $\boldsymbol{b}$ are said to form a *minimal pair*, if $\boldsymbol{a} \cap \boldsymbol{b} = \mathbf{0}$, i.e. the only degree below both of them is $\mathbf{0}$. This indicates that $\boldsymbol{a}$ and $\boldsymbol{b}$ have no nontrivial common information. These notions of strong randomness are also *generalized low (GL_1)*. Here, a set A is GL_1 if $A' \leq_T A \oplus \emptyset'$. This is often viewed as an analogue of computational lowness for the global degrees (these concepts will soon be made clear). Consequently the strong versions of randomness are sometimes thought to be the more appropriate definitions for randomness – we intuitively expect that we should not be able to extract a lot of useful information from a "random" string.

An important topic in this thesis is lowness. The traditional concept of lowness is from a computational point of view, using the *jump operator*. It is long recognized that the jump operator is a fundamental operator on the structure of the Turing degrees. For a set A, the Halting problem relative to A is denoted by A', where $A' = \{e \in \mathbb{N} \mid \Phi_e^A(e) \downarrow\}$. Here Φ_e^X is the e^{th} Turing functional. One can iterate this operation to get the iterated jumps $A', A'', A^{(3)}, A^{(4)}, \cdots$. If a set, a function or a degree is Turing reducible to $\emptyset'$, we call it Δ_2^0. A set A is *low* if $A' \equiv_T \emptyset'$. Hence a low set A is indistinguishable from the empty set as far as the jump operator is concerned. We will expect that low sets resemble computable sets, and there is a long and rich literature exploring this idea. The dual notion of lowness is called highness. A set is *high*, if $A' \equiv_T \emptyset''$. A high c.e. set resembles $\emptyset'$ in its computational power. A c.e. set is *complete* if $A \equiv_T \emptyset'$.

A different form of computational lowness and highness can be obtained by placing restrictions on the access to the oracle during a computation. We say that A is *weak truth table (wtt) reducible to B*, denoted by $A \leq_{wtt} B$, if there is a computable function that bounds the use function from above. If we require additionally that the wtt-reduction converges on every input x and every oracle X, then such a reduction procedure is a *truth table (tt) reduction*. We say that $A \leq_{tt} B$, if A can be computed from B via a truth table procedure. A set A is called *superlow* if $A' \equiv_{tt} \emptyset'$. This is equivalent to $A' \equiv_{wtt} \emptyset'$. The superlow sets resemble the computable sets closely. In a similar way, one defines a *superhigh* set A as a set where $A' \equiv_{tt} \emptyset''$.

This thesis also examines lowness in connection with randomness. These are the sets far away from being random. As in the classical notion of computational lowness, a lowness notion should generally be defined as one which is indistinguishable from the trivial sets in the operations concerned. We list some of the more famous lowness notions in randomness which have appeared in literature. A is *low for K* if $\forall\sigma(K(\sigma) \leq^{+} K^{A}(\sigma))$. Many concepts in computability theory can be *relativized*; in fact relativization plays an important role in understanding the transfer of information between sets, and also in the understanding of the global structure of the Turing degrees. If we have an effective process, then relativizing the process with respect to the oracle X, means that we run the process with oracle X present everywhere. For instance, K^{A} denotes the relativized K-complexity with oracle A. A is *low for Martin-Löf randomness*, if every ML-random relative to A is also Martin-Löf random. A is *K-trivial* if $K(A\upharpoonright n) \leq^{+} K(n)$ for every n. A set A is *low for Ω* if Ω is Martin-Löf random relative to A.

Martin-Löf random sets have been shown to be computationally powerful. For instance each Martin-Löf random set computes a diagonally noncomputable (d.n.c.) function, where a function f is d.n.c. if $f(x) \neq \varphi_{x}(x)$ for every x. A d.n.c. function is in some sense far away from being computable because we can effectively obtain an input demonstrating $f \neq \varphi_{x}$. Also the class of Martin-Löf random sets has a lot of computational power because *any* set X is Turing reducible to some Martin-Löf random set. The main questions driving our research are the following: is there a connection between computational strength, and how random a class is? Is it true that close to random means high computational power, and far from random means low computational strength? We explore these questions in the first two parts of the thesis.

In the classical case of computational lowness, there have also been evidence that descriptive properties are closely related to computational properties. For instance, Harris [Har] showed that the low$_n$ degrees can be characterized in terms of a property he called uniform escape property. For the case $n = 1$, he showed that a degree $\boldsymbol{a}$ is low iff for every $f \leq_T \boldsymbol{a}$, we can effectively find a computable function h such that f fails to dominate h. It is natural to ask if there is also a similar connection between descriptive and lowness properties in randomness. Recent research has shown that traceability plays an important role in understanding this connection.

Informally a function f from $\mathbb{N} \mapsto \mathbb{N}$ is traceable if we can obtain, in some simple way, a finite collection of possible values for each $f(x)$. This collection of possibilities is known as a trace for f, and the most common requirement on the trace is that it should be computable or c.e., although variations are considered in Chapters 3 and 4. Formally,

Definition 1.1.1. An *order function* $h : \mathbb{N} \mapsto \mathbb{N}$ is a function which is total, nondecreasing and unbounded. A function f is *traceable with respect to an order* h, if there is a computable function g such that for every x, $|W_{g(x)}| \leq h(x)$ and $f(x) \in W_{g(x)}$.

This concept differs from being approximable, in the sense that we only require a collection of possible values for $f(x)$, and we do not need to indicate the correct value in any way.

Definition 1.1.2. A set A is *jump traceable*, if the function $J^A(x)$ is traceable with respect to some computable order function.

Here $J^A(x)$ is the universal partial A-computable function, $J^A(x) = \Phi^A_x(x)$. We note that $J^A(x)$ denotes the *output* of the universal function, rather than the value of $A'(x)$. For this reason, jump traceability is akin to, but not the same as being superlow.

Figueira, Nies and Stephan [FNS06] studied a notion called strong jump traceability.

Definition 1.1.3. A set A is *strongly jump traceable*, if for every computable order function h, A is jump traceable with respect to h.

A strongly jump traceable set is very close to being computable, because we can predict properties of A to any degree of accuracy we require.

There is an obvious connection between a c.e. trace, and a system of descriptions (i.e. a prefix-free machine). The former can be viewed as a "discrete" version of the latter: a c.e. trace places discrete bounds on cardinality, while a machine talks about measure. Using this connection it is easy to see that in fact *no* Martin-Löf random set can be jump traceable. The work of Terwijn and Zambella [TZ01], Kjös-Hanssen, Nies and Stephan [KHNS05], Bedregal and Nies [BN03] have shown that a different form of traceability, called computable traceability, was related to the low for Schnorr random sets. Further research by Nies [Nie02, Nie05b], and Cholak, Downey and Greenberg [CDG08] have demonstrated that jump traceability and strong jump traceability were intimately related to the class of low for random sets.

In Chapter 2 we prove a separation result regarding the different levels of jump traceability. In particular, we show that if h_0 and h_1 are two computable order functions which grow at sufficiently different rates, then the classes of jump traceable sets which they generate are different. Our result shows that the theory of jump traceable sets was rich enough to obtain a proper hierarchy of sets. The levels in the hierarchy are classified according to how efficiently a set may be traced. In Chapter 2 we also show that the index set $\{e \in \mathbb{N} \mid W_e \text{ is strongly jump traceable}\}$ is complete amongst the sets definable with four quantifiers. The proof of Π^0_4-completeness uses a tree argument, which may be of independent technical interest. The completeness of the strongly jump traceable c.e. sets indicate that they too, have a rich and complex theory.

In Chapter 3 we use relativization to define a natural strengthening of strong jump traceability, which we call *hyper jump traceability.* This class is very far away from being random in the sense of having strong tracing properties. We will therefore expect the class to have very low computational strength. We prove that the hyper jump traceable c.e. sets exhibit somewhat unexpected properties. We first prove the basic results: there is a noncomputable hyper jump traceable c.e. set, and that not every strongly jump traceable set is hyper jump traceable. Recall that prompt simplicity is a dynamic property which describes the speed at which numbers are enumerated into a c.e. set. A promptly simple set resembles $\emptyset'$ in its dynamic properties. We prove that no hyper jump traceable c.e. set can be promptly simple.

This is a somewhat atypical behaviour, and gives the first example of a lowness class (in both classical computability and randomness) with no promptly simple members. On the other hand, this result is somewhat delightful, because we expect a lowness class to be not only computationally weak, but should not resemble $\emptyset'$ in its dynamic properties.

We next prove that the hyper jump traceable c.e. sets can be cuppable. For any two degrees $\boldsymbol{a}$ and $\boldsymbol{b}$, we say that $\boldsymbol{a}$ *cups with* $\boldsymbol{b}$, if $\mathbf{a} \cup \mathbf{b} = \mathbf{0}'$. A c.e. degree $\mathbf{a}$ is *cuppable* if there is a c.e. degree $\mathbf{b} < \mathbf{0}'$ such that $\boldsymbol{a}$ cups with $\mathbf{b}$. This gives the first example of a class which is defined without any references to cupping or capping, but is simultaneously cuppable and never low-cuppable (i.e. no low c.e. degree cups with it). Lastly we prove that there is a single c.e. set which forms a minimal pair with every c.e. hyper jump traceable set. The results in Chapter 3 show that a descriptive notion arising in randomness can also be used to obtain results in classical computability. Our research again highlights the fact that algorithmic randomness and classical computability theory are deeply intertwined.

In Chapters 4 and 5 we study the dual notion of lowness – *highness*. There are two well-known ways of defining highness in classical computability. The first is by defining the highness class based on computational properties. The other approach is by relativizing a lowness notion and considering the class of reals in which the Halting problem is low relative to. In the classical case of computational highness these two approaches coincide: a c.e. set A is high iff $\emptyset'$ is low relative to A. In Chapters 4 and 5 we study highness notions in randomness, beginning by considering the second approach. We investigate the dual notion of strong jump traceability, which we call *ultrahigh*. These are the c.e. sets A where $\emptyset'$ is strongly jump traceable relative to A. The two dual notions are connected by the pseudojump inversion theorem of Jockusch and Shore [JS83]. Properties of the strongly jump traceable sets are reflected correspondingly by the ultrahigh sets. For instance, it immediately follows from the pseudojump inversion theorem, that there is an incomplete ultrahigh c.e. set. It is also easy to see that every ultrahigh c.e. set is superhigh, and thus they resemble $\emptyset'$ in terms of their computational abilities.

In Chapter 4 we explore the degree theoretic properties of the c.e. sets close to $\emptyset'$, such as their ability to form minimal pairs. Specifically, we construct a minimal pair of superhigh c.e. sets, and a ultrahigh c.e. set which is half of a minimal pair. The techniques involved in these proofs require a careful scheduling procedure which decides when numbers are allowed to be enumerated. We expect this technique to be of use in other proofs involving structural results about the c.e. degrees, particularly in the constructions of meets. In Chapter 5 we examine the truth table degrees of ultrahigh sets. In particular we show that the diamond lattice $\{\mathbf{0}, \mathbf{1}, \boldsymbol{a}, \boldsymbol{b}\}$ can be embedded in the c.e. tt-degrees preserving top and bottom, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are ultrahigh. These results indicate that whilst the superhigh and ultrahigh c.e. sets have high computational power, they also exhibit degree theoretic properties typical of lowness classes.

In Chapter 6 we reprise the theme of relativization. This time we use partial relativization to define weak reducibilities. "Weak reducibility" is a term which loosely refers to any reducibility having a relatively simple definition, and which implies Turing reducibility. A weak reducibility induces a degree structure in the same way as Turing reducibility. We show a marked difference between $\leq_{SJT}$, the reducibility obtained by partially relativizing strong jump traceability, and Turing reducibility. In connection with randomness, we demonstrate another difference between the weak reducibility $\leq_{JT}$ obtained by partially relativizing jump traceability, and Turing reducibility. If $\leq_W$ is a weak reducibility, then a W-base for randomness is a set A such that $A \leq_W Z$ for some A-random set Z. We prove that any JT-base for randomness is trivial, while in contrast there is a noncomputable T-base for randomness. This gives two properties of Turing degrees where their analogues fail in the weak degrees, and suggests that the weak degrees are structurally very different from the Turing degrees.

In Chapter 6 we also study PA degrees and random reals. A Turing degree $\mathbf{a}$ is called PA, if it computes a complete extension of Peano's Arithmetic. Hence a PA degree contains a lot of number-theoretic information. One of the many characterizations of PA degrees says that the Turing degrees below a PA degree form a basis for Π^0_1 classes. Kučera was one of the first to draw the connection between PA degrees and ML-random reals, where in [Kuč85] he showed how coding techniques based on Π^0_1 classes of positive measure can be applied to show results about ML-random degrees. Indeed coding into PA degrees (via Π^0_1 classes) is seen to be much more permissive than coding into ML-random degrees. Kučera and Slaman [KS07] showed that there is a single low PA degree which computes all the K-trivial degrees, while it is not known if there is a low random degree which does this. The exact relation between the PA degrees and the ML-random degrees was finally clarified by Stephan [Ste06], where he showed that a PA degree is ML-random, iff it computes the Halting problem. This phenomenon is described by many as a dichotomy in the ML-random degrees. In Chapter 6 we show that every PA degree is the join of two ML-random degrees. Thus, in spite of the fact that incomplete PA degrees cannot be random, they can always be split into two random degrees. This demonstrates that computationally strong is again related to being close to random. We will also use techniques in Π^0_1 classes to obtain various results about a particular weak degree structure (called LR-degrees) arising naturally in the study of randomness.

In Chapter 7 we study effective dimensions. The classical Hausdorff and packing dimensions are ways of classifying different sets of measure zero. Various authors (Lutz [Lut00], Artheya et al [AHLM04], and Reimann [Rei04]) showed that the effective versions of these classical dimension notions can be expressed in terms of Kolmogorov complexity. The effective packing dimension of a real A is $\limsup_n K(A\upharpoonright n)/n$; while the effective Hausdorff dimension is $\liminf_n K(A\upharpoonright n)/n$. Effective packing dimension can be seen to be more tractable than effective Hausdorff dimension, because of a property called “dimension extraction”. This allows us to have a 0-1 law on the effective packing dimension of a Turing degree, while the corresponding fact is not true of effective Hausdorff dimension. We investigate the Turing degrees of effective packing dimension 1.

Effective packing dimension is a way of smooting out the oscillations in the Kolmogorov complexity. For a random real, every initial segment is of high complexity. The prefixes of a real with effective packing dimension 1 do not always need to be of high complexity. However infinitely many of its initial segments will look like an initial segment of a random real. We return to the theme of relating tracing properties with the amount of random content in a real. We demonstrate that a tempting characterization of these degrees in terms of traceability fails. Specifically the tracing notion concerned here is *c.e. traceability.* Informally a real A is c.e. traceable if every total function computable in A can be traced with respect to the identity order function. This traceability notion implies having low random content, in that every c.e. traceable degree has effective packing dimension 0. However we show that c.e. traceability is too strong for characterizing dimension 0 reals. We construct two different counterexamples, the first one we make hyperimmunefree by forcing with highly branching trees. Here a degree $\boldsymbol{a}$ is said to be *hyperimmunefree*, if for every total function $f \leq_T \boldsymbol{a}$, there is a computable function h such that $h(x) > f(x)$ for every x. We construct the second example below $\emptyset'$.

In Chapters 8 and 9 we study different variants of Martin-Löf randomness. This reveals interesting connections between the theory of algorithmic randomness, and classical Turing degree theory. In Chapter 8 we define and investigate two weak forms of ML-randomness, by looking at specific types of Martin-Löf tests. We consider ML-tests of finite cardinality, and the ML-tests of finite cardinality with a computable bound on the cardinality. We call the reals that avoid every test of the first type *finitely bounded random.* The reals that avoid every test of the second type are called *computably bounded random.* We show that finitely bounded randomness coincides with Martin-Löf randomness on the sets below $\emptyset'$, and they are different once we have enough oracle power to separate the finite from the infinite. We also characterize the c.e. degrees which compute a computably bounded random real, with the c.e. degrees which are not totally ω-c.e.. A c.e. degree is totally ω-c.e. if every function which can be computed from it is truth-table reducible to $\emptyset'$. The construction of a computably bounded random below an arbitrary c.e. degree requires the same level of "multiple permitting" as described by the non totally ω-c.e. degrees. This is the first class arising in the study of algorithmic randomness, whose construction can be described in this way. Again this relates a randomness class with a descriptive property arising in classical computability.

In Chapter 9 we study a strong version of Martin-Löf randomness, called *Demuth randomness.* The Demuth random reals lie between the 2-random reals, and the Martin-Löf random reals. We investigate the lowness notion associated with Demuth randomness. A real is *low for Demuth randomness*, if every real which is Demuth random relative to A is also Demuth random. We relate this lowness class in randomness, with another lowness notion in the global Turing degrees – hyperimmunefree. The hyperimmunefree degrees are weak in the sense that every function computable in a hyperimmunefree degree is dominated by a computable function. For this reason they have also been known by other names, such as computably dominated, or almost computable. Hyperimmunefree degrees are closely related to traceability notions. Computable traceability is sometimes known as the uniform version of being hyperimmunefree. We prove that each real which is low for Demuth randomness has to be of hyperimmunefree degree. It is still open if every low for Demuth randomness real is computable.

1.2 Turing degree theory

In the third and last part of the thesis we present results on the structure of the Turing degrees, particularly the c.e. degrees. We study various structural properties of the upper semi-lattice of the c.e. Turing degrees, such as splitting, cupping and capping. Following the theme of the first two parts, we will relate these structural properties with computational power. The techniques developed in this work will contribute to the understanding of the dynamics behind the general construction of c.e. sets.

In Chapter 10 we study the c.e. degrees that can be split into smaller c.e. degrees with low computational complexity. More specifically we are interested in the effect computational power has (in the form of highness) on the ability of a c.e. degree to split. A classical theorem of Sacks [Sac63b] says that every c.e. degree can be split into two low c.e. degrees. Hence, Sacks' theorem shows that no requirement on the computational power of a degree is needed, if we wanted to split into low degrees. Bickford and Mills [BM] showed that $\mathbf{0}'$ is the join of two superlow c.e. degrees. This result assumes we have the full computational power of the Halting problem. What about the c.e. degrees less than $\mathbf{0}'$? In Chapter 10 we show that there is an ultrahigh c.e. degree which cannot be split into two superlow c.e. degrees. This surprising result says that even ultrahighness is not sufficient to guarantee a superlow split, and that Turing completeness is necessary. We then consider if it is always possible to perform such a splitting if we require a property which is weaker than superlow. We return to one of our main themes, the c.e. traceable degrees. We show that every high c.e. degree can be split into two c.e. traceable degrees, and give an example of a high$_2$ c.e. degree which cannot be split this way.

In Chapter 11 we use computational lowness to define a new ideal in the c.e. degrees. Bickford and Mills [BM] defined a c.e. degree $\mathbf{a}$ to be deep, if it perserves the jump of every c.e. degree under join, i.e. for every c.e. $\mathbf{b}$, we have $(\mathbf{a} \cup \mathbf{b})' = \mathbf{b}'$. The motivation for looking at this class came from an interest in finding new definable ideals in the c.e. degrees. Lempp and Slaman [LS89] showed that every deep degree is computable. An attempt to salvage this idea of deepness led Cholak, Groszek and Slaman [CGS01] to define the notion of almost deep; $\mathbf{a}$ is *almost deep* if it preserves the jump of every low c.e. degree under join, i.e. $\mathbf{a} \cup \mathbf{b}$ is low for every low c.e. degree $\mathbf{b}$. They showed that this ideal is nontrivial. In Chapter 11 we define a related concept, which we call almost superdeep. $\boldsymbol{a}$ is *almost superdeep*, if $\boldsymbol{a} \cup \boldsymbol{b}$ is superlow for every superlow c.e. degree $\boldsymbol{b}$. We construct a noncomputable almost superdeep degree.

Related to the notion of splitting, is the concept of cupping. In Chapter 12 we investigate the interplay between computational lowness, traceability and the structural property of cupping. One can get a hierarchy of Δ^0_2 sets by looking at the power of the iterated jumps; this hierarchy is known as the *high-low hierarchy.* A Δ^0_2 set A is said to be *low$_n$*, if $A^{(n)} \equiv_T \emptyset^{(n)}$, and is *high$_n$* if $A^{(n)} \equiv_T \emptyset^{(n+1)}$. The position of a Δ^0_2 set A in this hierarchy describes how similar A is to $\emptyset$ or $\emptyset'$, as measured by the jump operator. On the low end of the hierarchy we have: computable $\subset$ low $\subset$ low$_2$ $\subset$ low$_3$ $\subset \cdots$, and on the other end, we have $\cdots \subset$ high$_2$ $\subset$ high $\subset$ complete. In a remarkable paper, Ambos-Spies, Jockusch, Shore and Soare [ASJSS84] showed the coincidence of several classes of c.e. degrees. They showed that the low-cuppable c.e. degrees can be characterized by prompt simplicity. Downey et al [DGMW06] showed that every low-cuppable c.e. set can also be cupped with a c.e. traceable set; the latter class they called AC-cuppable. In Chapter 12 we show that every low-cuppable set can be cupped with a low c.e. traceable set. We also show that every low$_2$-cuppable set can be cupped with a c.e. traceable set. These new cupping classes coincide with the known ones: (low+AC)-cuppable=low-cuppable, and AC-cuppable=low$_2$-cuppable. In Chapter 12, we also prove the existence of a set which is cuppable, but can *only* be cupped with high sets. This refutes the conjecture made by Li, where he postulated that every cuppable c.e. set has a low$_n$-cupping partner for some fixed n.

In Chapter 13 we investigate a notion called *contiguity*. Our motivation for this chapter is to relate definability with sets of intermediate computational strength. A c.e. degree $\boldsymbol{a}$ is said to be contiguous, if it contains a single c.e. wtt-degree. Contiguous degrees have been shown to be an important tool for transferring certain results about wtt-degrees to results about the Turing degrees (see Ladner and Sasso [LS75], and Downey [Dow87]). Contiguous degrees are all low$_2$, and are also definable in the degrees. We show that $\boldsymbol{0'} = \boldsymbol{a} \cup \boldsymbol{b}$ for some contiguous degrees $\boldsymbol{a}$ and $\boldsymbol{b}$. By a similar method to the one in Chapter 10, one might show that every *high* c.e. degree is the join of two contiguous degrees. We hope to use this method in future work to find a natural definable class of c.e. sets which are simultaneously nonhigh and nonlow$_2$.

We have seen how tabular reducibilities can be used to define lowness and highness notions. In Chapter 14, we study the interaction of strong tabular reducibilities with a classical result, jump inversion. Kleene and Post [KP54] formally introduced the jump as an operator on degrees. In the paper, they asked a number of fundamental questions about the structure of the Turing degrees $\mathcal{D}$, including the definability of the jump operator, the density of $\mathcal{D}$ and the range of the jump operator. The last question is of interest to us in Chapter 14. Friedberg showed that the range of the jump operator is $\{\mathbf{a} \mid \mathbf{a} \geq \mathbf{0}'\}$. Shoenfield [Sho59] then showed that the range of the jump operator, when the domain is restricted to the degrees below $\mathbf{0}'$, is exactly $CEA(\mathbf{0}')$. Here $CEA(\mathbf{x}) = \{\mathbf{a} \mid \mathbf{a} \geq \mathbf{x}$ and $\mathbf{a}$ is c.e. relative to $\mathbf{x}\}$. The jump $\mathbf{a}'$ of any degree $\mathbf{a}$ is necessarily above $\mathbf{0}'$, and if $\mathbf{a} \leq \mathbf{0}'$ then $\mathbf{a}' \in CEA(\mathbf{0}')$. These two old theorems show that the range of the jump operator (even on a restricted domain) is the largest possible. Later, Sacks [Sac63a] showed that additionally the range of the jump operator restricted to the c.e. sets is also $CEA(\mathbf{0}')$. These three classical jump inversion results have other consequences. For instance they contributed to the understanding of the rigidity of $\mathcal{D}$, and also showed that the c.e. degrees can be stratified nontrivially into subclasses according to the complexity of the jumps. Our interest is in the range of the jump, as an operator on the tt-degrees $\mathcal{D}_{tt}$. Mohrherr [Moh84] and Anderson [And08] proved that the range of the unrestricted jump operator on $\mathcal{D}_{tt}$ is exactly every tt-degree above $\mathbf{0}'_{tt}$. Hence their results said that the tt-analogue of Friedberg's jump inversion holds. In Chapter 14, we show that both the tt-analogue of Shoenfield's jump theorem, and the tt-analogue of Sacks' jump theorem fail. We also demonstrate that they fail in a very strong way.

In Chapter 15 we look at *diagonal sets*. A c.e. set is a diagonal if it is the Halting problem relative to an arbitrary computable numbering. Diagonal sets are analogues of $\emptyset'$, and thus contain a fair amount of nontrivial information. Kummer showed that diagonal sets can be described by a property on the lattice of c.e. sets. Let $\mathcal{L}(A)$ be the lattice of c.e. supersets of A, and I^* ($I^{c.e.}$) be the ideal of supersets B of A such that $B - A$ is finite (and c.e., respectively). We let $\mathcal{L}^*(A) = \mathcal{L}(A)/I^*$ and $\mathcal{L}^{c.e.}(A) = \mathcal{L}(A)/I^{c.e.}$. A c.e. set A is *hyperhypersimple* if $\mathcal{L}^*(A)$ is a Boolean algebra, and *maximal* if $\mathcal{L}^*(A)$ is the trivial two element Boolean algebra. A well-known theorem of Martin [Mar66] says that the c.e. degrees containing a maximal set are the c.e. degrees containing a hyperhypersimple set. These degrees also conincide with the high c.e. degrees. Our work in Chapter 15 is about the Turing degrees of analogues of these two notions.

Part of our motivation came from the work of Kummer [Kum91], and Herrmann and Kummer [HK94]. They showed that a certain analogue of these notions is connected to the study of diagonal sets. A c.e. set A is *semi-hyperhypersimple* if $\mathcal{L}^{c.e.}(A)$ is a Boolean algebra, and *semi-maximal* if $\mathcal{L}^{c.e.}(A)$ is the trivial two element Boolean algebra. Their work showed that the semi-hyperhypersimple c.e. sets are exactly the sets which are not diagonals.

We also consider a different analogue of hyperhypersimplicity and maximality. A *set splitting* of a noncomputable c.e. set A, is a pair (A_0, A_1) of disjoint noncomputable c.e. sets where $A_0 \cup A_1 = A$. A c.e. set A is *hemi-hyperhypersimple*, if it is half of a set splitting of a hyperhypersimple set (*hemi-maximal* sets are defined similarly). These sets have been instrumental in the search for orbits of automorphisms of the c.e. sets, as well as in the understanding of the dynamics behind the automorphism machinery. In Chapter 15 we answer a question of Kummer [Kum91] about the Turing degrees of these analogues. We show that the semi-hyperhypersimple, semi-maximal, hemi-hyperhypersimple and hemi-maximal degrees are all different. As a consequence, the expected analogues of Martin's theorem do not hold.

This thesis is divided into three parts, which consist of research in the topics mentioned above. Most of the work has been submitted for publication in journals. Hence each chapter will contain a short introduction of its own, explaining the motivation for the chapter. Chapter 5 is joint work with Jiang Liu, and Chapter 6 is with George Barmpalias and Andy Lewis. Chapters 7, 9 and 10 are joint work with Rod Downey, and Chapter 8 is with Rod Downey and Paul Brodhead. Chapter 12 is joint work with Noam Greenberg and Guohua Wu, and Chapter 14 is with Barbara Csima and Rod Downey.

Finally we list some notations used in this thesis. It is common to identify a set $A \subset \mathbb{N}$ with the infinite binary string α, where α is the characteristic function of A (and vice versa). For this reason we also refer to a subset of $\mathbb{N}$ as a real number. For a set A, $|A|$ or $\#A$ is used to denote the cardinality of A. If σ is a finite string then $|\sigma|$ denotes the length of σ. We use $^\frown$ to denote string concatenation. We write $\sigma \supseteq \tau$ to mean that σ extends τ, and $\sigma \supset \tau$ to mean that σ strictly extends τ. If A is an infinite binary string then $A\upharpoonright n$ denotes the first n bits of A. We say that a function f dominates another function g, if $f(x) > g(x)$ for almost every x.

Part I

Traceability

Chapter 2

On strongly jump traceable reals

Published in the Annals of Pure and Applied Logic (2008), **154**, pp. 51-69.

2.1 Introduction

One of the fundamental concerns of computability theory is in understanding the relative difficulty of computational problems as measured by Turing reducibility ($\leq_T$). In this work we are concerned with *computational lowness*, where a set A is low relative to the Turing jump if $A' \equiv_T \emptyset'$. There is a long and rich literature exploring the idea that low sets resemble computable sets. This seems particularly true for the computably enumerable sets, and we mention a few well known examples.

Soare [Soa82] showed that if A is c.e. low, then the lattice of c.e. supersets of A is isomorphic to the lattice of c.e. sets modulo finite sets. Robinson [Rob71a] extended the Sacks' Splitting Theorem [Sac63b] by showing that any c.e. degree can be split above a low one. A generalization of Lachlan's Nondiamond Theorem [Lac72] by Ambos-Spies [AS84b] over low degrees further shows that the interval $[\boldsymbol{a}, \mathbf{0'}]$ is structurally very similar to $[\mathbf{0}, \mathbf{0'}]$ when $\boldsymbol{a}$ is low.

Shore and Slaman [SS90] showed that there is no Slaman triple below a c.e. low (in fact, low$_2$) degree, while in [SS93] they proved that every high degree (A is high iff $A' \equiv_T \emptyset''$) bounds a Slaman triple. This gives an elementary property which separates the high and low (low$_2$) c.e. degrees. Further work by Soare (no low set is speedable [Soa77]), and Downey and Jockusch (every low Boolean algebra is isomorphic to a computable one [DJ94]) further demonstrate that the low sets behave just like the computable ones. In an amazing paper, Slaman and Solovay [SS91] showed that every set A which was low for EX learning, is also low (in fact 1-generic below $\emptyset'$). Slaman and Solovay's result demonstrates that lowness for various notions of computation can be intertwined. In this case, they demonstrate a relationship between a lowness concept from the theory of inductive inference and another seemingly unrelated lowness concept from computability theory. This idea is further explored in this chapter, where we will investigate lowness for Kolmogorov complexity and its relationships.

Because of these results and perhaps also because the "Robinson trick" (see Soare [Soa87], Chapter XI) was so well understood, the low computably enumerable sets were thought to be reasonably understood in terms of their degree-theoretical properties.

However, recent work has again turned the spotlight on this class, and demonstrated that low c.e. sets have an astonishing theory. This is due to their relationship with another computational lowness notion - lowness in terms of Kolmogorov complexity - and has been shown to be related to the widely studied class of the K-trivials.

The class of K-trivial reals was first introduced in [Sol75]. They are the reals α such that for some constant c, $K(\alpha\upharpoonright n) \leq K(n) + c$ for every n. That is, the K-trivials have got very low initial segment complexity, similar to the computable ones. Solovay [Sol75] used what is now known as a *cost function construction* to show the existence of noncomputable K-trivial reals. In spite of Solovay's theorem, the resemblance they bear with the computable reals makes one wonder if they are related to the low sets. Recent work has shown that this is indeed the case. In particular Nies [Nie02, Nie05b] showed that every c.e. K-trivial was superlow, and Cholak, Downey and Greenberg [CDG08] discovered that a certain class of reals exhibiting very strong "lowness properties" forms a proper natural subclass of the K-trivials.

A very important notion involved is traceability. Recent development has indicated this to be a fundamental notion involved in the study of simple classes arising in randomness. Recall that a set A is *jump traceable*, if the function $J^A(x)$ is traceable with respect to some computable order function. We mention two other variants on jump traceability. Ishmukhametov [Ish97] studied *c.e. traceability.* A set A is said to be c.e. traceable, if there is a computable order function such that for every total function $f \leq_T A$, there is a c.e. trace $\{T_x\}$ obeying h, such that $f(x) \in T_x$ for every x. In fact this class is of independent interest because it implies another important notion in computability, known as *array computability.* In the case of the c.e. sets it is equivalent to being array computable. A set is array computable (introduced by Downey, Jockusch and Stob [DJS96]) if there is some $f \leq_T \emptyset'$ such that for every $g \leq_T A$, f dominates g. Hence the computably enumerable c.e. traceable sets capture many classes of sets arising in classical computability sharing common features (demonstrated by the array computable sets). Specifically, these are the sets in which certain permitting notion (multiple permitting) fails. Examples of these characterizations include the c.e. degrees with strong minimal covers (Ishmukhametov [Ish97]), and the c.e. degrees which do not compute a Martin-Pour-El theory. This is a special type of theory which is an analogue of maximal sets. In Ng, Stephan and Wu [NSW06], it was shown that the computably enumerable c.e. traceable degrees are exactly the c.e. degrees in which every set in the degree is the arithmetic difference of two left-c.e. reals. Hence c.e. traceability is even related to concepts arising in computable analysis.

Terwijn and Zambella [TZ01] introduced the concept of *computable traceability.* A set is computably traceable if the definition for c.e. traceability holds with a computable trace in place of a c.e. trace. That is, $\forall x(T_x = D_{g(x)})$ for some computable g, and D_k is the k^{th} canonical finite set. Computable traceability can be seen as a uniform version of being hyperimmunefree. Since every total function (as an infinite string in ω^ω) can be compressed as much as we require, being computably (or c.e.) traceable with respect to a single computable order is equivalent to being traceable at every computable order. Since every set A of hyperimmunefree degree is "computably dominated", every c.e. trace for a total A-computable function can be converted into a computable trace. Hence c.e. traceability and computable traceability are the same on the HIF sets.

If R is a notion of effective randomness, then *low for R* would denote all the sets A for which $R^A = R$ (i.e. every random Z is still random relative to A). The work of Terwijn and Zambella [TZ01], Kjos-Hanssen, Nies and Stephan [KHNS05], Bedregal and Nies [BN03] have revealed an interesting interaction between "predictability" in terms of traceability, and simplicity in terms of Kolmogorov complexity. They showed that

Theorem 2.1.1. *A is low for Schnorr random iff A is computably traceable.*

This reveals that traceablility is related to at least some lowness notion in randomness. With this insight, Nies [Nie02] introduced the jump traceable sets to study lowness properties. A jump traceable set A differs from a superlow set in the sense that we are able to effectively enumerate finitely many candidates for each $J^A(x)$. For a superlow set A we are only merely able to approximate whether $J^A(x)$ converges. Since we are able to code finite information into $J^A(x)$, it might appear that being jump traceable is stronger than being superlow. However Nies [Nie02] showed that for c.e. sets, jump traceability and superlowness were the same, and asked if they were the same for n-c.e. sets. In Theorem 2.2.1, we answer the question positively:

Theorem 2.2.1. *Let A be n-c.e. for $n \geq 1$. Then A is superlow iff A is jump traceable.*

When we consider the next level on the Ershov hierarchy, these two notions separate: it is not hard to see that no jump traceable set can be ML-random, so a superlow ML-random set cannot be jump traceable. In the other direction, Nies [Nie02] showed that there was an ω-c.e. jump traceable set which was not superlow. If we consider non-Δ^0_2 sets, the situation becomes even more bizarre. There is a perfect Π^0_1 class of sets which are jump traceable, via an exponential bound. Such a phenomenon highlights an important inherent property of being traceable; we are only able to enumerate possible values of $A \upharpoonright n$, but beyond that we are given no additional information to suggest which one of the enumerated values is correct. Indeed Kjos-Hanssen and Nies [KHN] showed that jump traceable sets can even be superhigh.

In further work, Nies [Nie02, Nie05b] showed that every K-trivial real was jump traceable, with an order function of growth rate $\sim h(n) = n \log^2 n$. Inspired by these results, Figueira, Nies and Stephan [FNS06] went on to study the notion of strong jump traceability. Recall that a set is strongly jump traceable if it is jump traceable with respect to every computable order. Figueira, Nies and Stephan showed the existence of a noncomputable strongly jump traceable c.e. set, by a cost function type construction, and characterized c.e. strong jump traceability via the notion of well approximability: a set A is *well approximable*, if for every order function h, A can be effectively approximated with less than $h(x)$ many changes at each input x. Figueira, Nies and Stephan showed that if A is c.e., then A is strongly jump traceable if and only if A' is well approximable.

In [DHNS03], Downey, Hirschfeldt, Nies and Stephan showed that the K-trivial reals are natural solutions to Post's problem in the following sense:

Theorem 2.1.2 (Downey, Hirschfeldt, Nies and Stephan [DHNS03]). *Every K-trivial real is Turing incomplete.*

They used a new method widely known as the "Decanter method". This method exploits the fact that for any given K-trivial real, we could challenge its triviality very slowly. That is, if we needed certification amounting to a certain mass (measure theoretically), we could do it in small fractions repeatedly. This resembles the "drip-feeding" action of a decanter, hence the fanciful name. For a good description of this method, we refer the reader to [Dow]. Nies [Nie02, Nie05b] then applied a nonuniform method of the Decanter method to show:

Theorem 2.1.3 (Nies [Nie02, Nie05b]). *Every K-trivial real is superlow.*

The technique involved is called the "golden run" construction. Essentially this method uses many individual Decanter procedures, and the highest procedure which does not return is the golden run; the desired effective objects are built there. For a general framework of the Decanter and golden run methods, we refer the reader to the upcoming book of Nies [Nie09].

The recent work of Cholak, Downey and Greenberg [CDG08] showed that the c.e. strongly jump traceable sets form a *proper* subclass of the K-trivials. In fact, they proved that if A is c.e. and jump traceable at order $\sim h(n) = \sqrt{\log n}$, then A is K-trivial. Again this involves a new technique known as "box promotion". This can be seen as an analogue of the Decanter method; when the constraints involved are measure theoretic, one measures progress against the opponent by the Decanter method. When we are dealing with discrete combinatorial notions such as traceability, the constraints involved are now *box sizes*. This is an important concept and will be a recurring theme in the following chapters. To wit, we think of a trace $\{T_x\}$ as an infinite sequence of boxes of certain (varying) sizes. Each T_x is a box of size $h(x)$, where h is the order where $\{T_x\}$ has to obey. When we enumerate a number z into T_x, we think of z as occupying the next available level in the box T_x, and the current size of the box T_x decreases, indicating that we have used up one more location allowed. If the opponent is playing the trace $\{T_x\}$ then we say that the box T_x *has been promoted*, indicating that the opponent has fewer opportunities to change the given set. The box promotion method relies on testing (the lowness of a given set A) simultaneously on many separate boxes. If the opponent wants to change A, then he has to promote every one of these boxes. We will then use a fraction of these promoted boxes for the same strategy, while saving the rest for other situations. Thus we guarantee that any changes the opponent makes is maximally painful for him.

Cholak, Downey and Greenberg [CDG08] used this technique to show that like the K-trivials, the c.e. strongly jump traceables are also closed under $\oplus$. They constructed a K-trivial c.e. real which is not jump traceable with respect to a bound of size $\sim h(n) = \log\log n$. In further work, Downey and Greenberg [DGa] showed that *every* strongly jump traceable set was K-trivial.

Due to the characterization by the well-approximability of the jump, one can view c.e. strong jump traceability as a natural strengthening of being superlow. Figueira, Nies and Stephan [FNS06] also studied the role that the size of the bound h has on jump traceability. They showed that for any order function h, there is always some set A which is jump traceable, but not jump traceable via h. That is, no matter how fast an order h grows, there is always some jump traceable set A for which h grows too slowly still, for the purpose of jump tracing A. The result of Figueira Nies and Stephan showed that there was no single maximal bound for jump traceability; they asked if there was a minimal bound for jump traceability: is there an order function h, such that every set A which is jump traceable via h is already strongly jump traceable? In Theorem 2.3.1, we answer the question in the negative:

Theorem 2.3.1. *For any given order function h, there is a c.e. set A and an order function $\tilde{h}$, such that A is jump traceable via h, but not jump traceable via $\tilde{h}$.*

In particular, there is no single order function such that strong jump traceability is the same as jump traceability for that order. Hence, unlike in the case of computable (or c.e.) traceability, strong jump traceability is different from jump traceability; in fact there is an entire hierarchy of jump traceable sets ordered by the growth rates of the bounding functions on the size of the trace. This hierarchy contains infinitely many strata in either direction.

More specifically we mean the following. Cholak, Downey and Greenberg showed in [CDG08] that the c.e. strongly jump traceables form a nonprincipal ideal. In fact, they showed something stronger: There is an effective procedure Λ, such that given any order function h, we have another order function $\Lambda(h)$ growing more slowly than h, such that if A and B are c.e. and jump traceable with respect to $\Lambda(h)$, their join $A \oplus B$ will be jump traceable with respect to h. For each order function h, we define

$$\mathcal{H}_h := \{\Lambda^n(h) : n > 0\},$$

where Λ^n denotes iterating the Λ-procedure n times. We say that A is $\mathcal{H}_h$-jump traceable, if A is jump traceable with respect to all $g \in \mathcal{H}_h$.

Clearly for each order function h, the class of c.e. sets A which are $\mathcal{H}_h$-jump traceable forms an ideal. When h grows slowly enough ($h(n) \ll \log\log n$), the ideal it generates is contained in the ideal of the K-trivials. Since we can always diagonalize against all the functions in each $\mathcal{H}_h$, it follows from Theorem 2.3.1 that there are infinitely many intermediate ideals lying between the ideals of the K-trivial sets, and the strongly jump traceable sets.

These results give further evidence to the fact that the growth rates of the bounding functions dramatically affect the properties of the class of sets which are jump traceable obeying the bound. For instance there are uncountably many sets jump traceable at some exponential bound, while Downey and Greenberg [DGa] showed that if one got down to a level of $\log\log n$, then every jump traceable real was K-trivial and hence Δ^0_2.

Nies observed that the K-trivial reals form the first example of a natural nontrivial Σ^0_3 ideal in the c.e. Turing degrees. Had there been a slowest order we could use to define strong jump traceability, the strongly jump traceables would also have to be Σ^0_3. We have already learnt from Cholak, Downey and Greenberg [CDG08] that there are K-trivials which are not strongly jump traceable, and so the following Theorem 2.4.3 would show that in terms of the complexity of the classes, the strongly jump traceables are as complex as they could be, and in fact differ from the K-trivials as much as they possibly can:

Theorem 2.4.3. *The set $\{e \in \mathbb{N} : W_e$ is strongly jump traceable$\}$ is Π^0_4-complete.*

We will use the technique in the proof of Theorem 2.3.1 as an atomic strategy in the proof of Theorem 2.4.3. As a corollary, we get the result of Cholak, Downey and Greenberg [CDG08] that not every K-trivial is strongly jump traceable.

2.2 The coincidence of superlowness and jump traceability for n-c.e. sets

In this section, we prove that for each n, the superlow n-c.e. sets are exactly the jump traceable n-c.e. sets. A crucial ingredient in the proof of Nies [Nie02] for $n = 1$, was the fact that we will never return to a previously witnessed computation, once we moved away from it. However we can overcome the obstacles for $n > 1$, if we require more stringent certifications.

Theorem 2.2.1. *Let A be n-c.e. for $n \geq 1$. Then A is superlow iff A is jump traceable.*

Before we begin, we state the following simple observation about A. It says that even though n might be > 1 and we can return to a previous configuration, we cannot do this too often.

Fact 2.2.2. *Suppose σ_1 and σ_2 are distinct finite strings. There cannot be stages $s_1 < s_2 < \cdots < s_{n+2}$ such that we alternate between $A_s \supset \sigma_1$ and $A_s \supset \sigma_2$.*

2.2.1 Every jump traceable n-c.e. set is superlow

Suppose A is n-c.e. and jump traceable. Let $A = \cup_s A_s$ be an approximation for A, such that for all x, $|s : A_{s+1}(x) \neq A_s(x)| \leq n$. Let h be the order function, and $\{T_x\}_{x\in\mathbb{N}}$ be the c.e. trace for J^A, with bound h. Let $\alpha(e)$ be a computable function such that for all e, $J^\sigma(\alpha(e)) = \sigma$ for every σ such that $J^\sigma(e)\downarrow$.

Discussion of the strategy: let us review why this is true for $n = 1$, and we will see how to adapt the proof to the case $n > 1$. Suppose we wanted to guess the convergence or divergence of $J^A(e)$. We initially guess divergence, until $J^A(e)[s] \downarrow$ and the use σ is traced in $T_{\alpha(e)}$. If that happens, we have received a confirmation that $\sigma \subset A$, and we may now switch to predict convergence. Now of course $\sigma \not\subset A$ may be true, and it may very well be the case that at some later stage $t > s$ we have $A_t \not\supset \sigma$. If this happens then we switch back to divergence again. In the c.e. case nothing is lost because A will never extend σ again, and so the only way the opponent can force us to switch back to convergence, is to use up a different slot in $T_{\alpha(x)}$. In the general case when A is Δ^0_2, things can go wrong for us because the opponent will do the following. He would show us convergence with use σ, wait for us to predict convergence, and then move A away from σ. He then waits for us to predict divergence, and then returns to $A_s \supset \sigma$ again. He can then make us change our mind as many times as he wishes, effectively destroying our hopes of making A' ω-c.e.

How do we make use of the fact that A is, say 2-c.e.? This assumption severely restricts the opponent's ability to flip back and forth (as described above), but he can still return to previous computations and annoy us. The key is to request for certification of the correctness of A, not only on segments $\sigma \subset A_s$ where $J^\sigma(e) \downarrow$, but also on segments where the jump has not yet converged. Let us make this more precise. Each slot in $T_{\alpha(e)}$ represents a possible use. For each of these slots, say there are three possible slots, we devote a part of the universal jump function $J^A(e_1)$ for the first slot, $J^A(e_2)$ for the second slot and so on. We also assume that we are enumerating axioms for $J^X(e_i)$, in an attempt to get certifications for initial segments of A.

Suppose now that $J^\sigma(e)[s] \downarrow$ and $\sigma \subset A_s$ has been traced in the first slot of $T_{\alpha(e)}$. We will switch to convergence. Suppose next the opponent moves A away from σ to try and force us to switch prediction to divergence, at stage $t > s$. We respond by enumerating $J^{\tau_1}(e_1) \downarrow= \tau_1$ where $\tau_1 = A_t \upharpoonright |\sigma|$ *before we switch to divergence*, to try and get certification that σ was indeed wrong and that τ_1 is correct instead. Since the opponent wants to force us to switch to divergence, he has no choice but to put $A_t \upharpoonright |\sigma|$ into T_{e_1}. We are satisfied in this case, and can then switch our guess to divergence.

Suppose the opponent wants to continue with his devious plan to confuse us. His next move would be to force us to switch to convergence. If he does so by showing us $J^{\sigma'}(e)\downarrow$ on some σ' incompatible with σ, then he would use up the second slot in $T_{\alpha(e)}$ and we can now repeat our strategy by using $J(e_2)$ to test for segments of A. So, suppose he returns to σ, i.e. make $A_s \supset \sigma$ now. We will then switch our guess to convergence, with the knowledge that the opponent *cannot make* $A \supset \tau_1$ *ever again*, since A is d.c.e. Therefore, before we next switch to divergence again, we can force T_{e_1} to fill up with some τ_2 different from τ_1. Therefore, the number of times we can switch our prediction is something like $3 \cdot |T_{e_3}| = h(e) \cdot |T_{e_{h(e)}}| = h(e) \cdot h(e_{h(e)})$.

For $n > 2$ a similar argument follows. The only difference being that he can return to make $A \supset \tau_1$ again, and we would not have gained anything. But he can only do this (i.e. flip A between σ and τ_1) at most n times, which increases the bound above by a constant factor of n. Note that if A was ω-c.e. then the opponent would be able to beat us by playing this game. In this case he could set the first use σ very large so that he gets enough space to flip between σ and τ_1 enough times to beat us. We are unfortunately not able to tell the use of $J^A(e)$ in advance. Indeed there are ω-c.e. sets which are jump traceable but not superlow.

The formal proof: we let $t_{x,k}$ denote the k^{th} element enumerated into T_x, if it exists. Now let $\beta(e,k)$ be such that $J^\sigma(\beta(e,k)) = \sigma$ for every σ such that $|\sigma| = |t_{\alpha(e),k}|$. Clearly β exists, and is total computable since it represents the indices of some consistent set of axioms. Let $f(e) := 2n \cdot h(\alpha(e)) \cdot h(\beta(e, h(\alpha(e))))$, and we will construct an ω-c.e. approximation $g(e,s)$ to $A'(e)$ with at most $f(e)$ mind changes.

We start with $g(e,0) = 0$ which represents divergence of $J^A(e)$ (1 represents convergence). We always keep $g(e,s+1) = g(e,s)$ the same, unless we encounter an event at stage s which causes us to switch our prediction. The events which can trigger a switch are the following:

- we switch from 0 to 1 at stage s, if $J^A(e)[s]\downarrow$ with use σ and $\sigma \in T_{\alpha(e)}$. Note that there can only be one such σ which applies at stage s, and we say that σ causes the current switch.

- we switch from 1 to 0 at stage s, if $A_s \upharpoonright |\sigma| \in T_{\beta(e,k)}$ and $A_s \upharpoonright |\sigma| \neq \sigma$, where

$\sigma = t_{\alpha(e),k}$ and σ caused the previous switch from $0 \to 1$.

Verification: fix an e, and we will now count the number of switches. Let $s_1 < s_2 < \cdots < s_N$ be precisely all the stages where we switch from 1 to 0. Note that for each $m \leq N$ we must have $A_{s_m} \upharpoonright |\sigma| \in T_{\beta(e,k)}$ where $\sigma = t_{\alpha(e),k}$ and σ caused the previous switch from $0 \to 1$. Note also that $A_{s_m} \upharpoonright |\sigma| \neq \sigma$. We associate stage s_m with the pair $\langle \sigma, A_{s_m} \upharpoonright |\sigma| \rangle$. Namely, we let $P(m) = \langle \sigma, A_{s_m} \upharpoonright |\sigma| \rangle$; doing this will help us calculate an upper bound for N.

Now suppose that for some $m' > m$ we have $P(m') = P(m) = \langle \sigma, \tau \rangle$. Then clearly at some stage t with $s_m < t < s_{m'}$ we have a switch from $0 \to 1$, and we have $\sigma \subset A_t$. Since $\sigma \neq \tau$, and the fact that A is n-c.e., it follows that the cardinality of $\{m \leq N : P(m) = \langle \sigma, \tau \rangle\}$ for each fixed $\langle \sigma, \tau \rangle$ is at most n. How many distinct pairs $\langle \sigma, \tau \rangle$ can there be? At most $h(\alpha(e)) \cdot h(\beta(e, h(\alpha(e))))$ many. So, the number of switches of $g(e, \cdot)$ is $\leq 2N$ and will be bounded by $f(e)$.

Next, we verify that $\lim_s g(e,s) = A'(e)$. If $J^A(e)\downarrow$ with use σ, then the limit $\lim_s g(e,s)$ cannot be 0 because $J^A(\alpha(e))\downarrow= \sigma$ and consequently σ must appear in $T_{\alpha(e)}$. Conversely if $\lim_s g(e,s) = 1$ then at some final stage s we flip $g(e,\cdot)$ from 0 to 1, after observing that $J^A(e)[s]\downarrow$ with use σ, where $\sigma = t_{\alpha(e),k}$. We claim that $\sigma \subset A$. Note that we do not require that $\sigma \subset A_t$ for all $t > s$, unless $n = 1$. Suppose $\sigma \not\subset A$, then we must have $J^A(\beta(e,k))\downarrow= A \upharpoonright |\sigma| \neq \sigma$, and this eventually shows up in $T_{\beta(e,k)}$, causing us to switch $g(e,\cdot)$ after it settles, contradiction.

2.2.2 Every superlow n-c.e. set is jump traceable

Suppose A is n-c.e. and superlow. Again let $A = \cup_s A_s$ be an approximation for A, such that for all x, $|s : A_{s+1}(x) \neq A_s(x)| \leq n$. We let the computable functions g and h be such that for all e, $\lim_s g(e,s) = A'(e)$ and $|s : g(e,s) \neq g(e,s+1)| \leq h(e)$. The number of $g(e,\cdot)$ changes so far is recorded by $ch(e,s) :=$ the largest x such that $2(x-1) \leq |t < s : g(e,s) \neq g(e,s+1)|$.

The description of strategy: again we describe first what happens in the c.e. case, and then discuss how the plan goes for $n > 1$. We want to trace potential jump computations $J^A(e)$ in X_e, where we get to build X_e. We also get to enumerate the functional with index v, where v is given by the Recursion Theorem. Each time we see $J^{\tau}(e)[s]\downarrow$ with $\tau \subset A_s$, we will copy the computation by setting $J^{\tau}(v)\downarrow$. Suppose after the opponent shows us jump computations with uses $\tau_1, \tau_2, \cdots, \tau_N$, and he decides to switch $g(v,\cdot)$ from 0 to 1, we will then believe in the current jump value, and put $J^{\tau_N}(e)[s]$ into X_e. If A is c.e. then he cannot return and make $A \supset \tau_i$ for any $i < N$, so we only need to trace one value (i.e. $J^{\tau_N}(e)[s]$) from this first lot of computations. The opponent can then go on and show us a second lot of jump computations, but we would only believe in any of the computations in the second lot, if we manage to force him to switch $g(v,\cdot)$ from $1 \to 0 \to 1$ again. Hence we clearly will have $|X_e| \leq h(v)$.

The problem if A is Δ^0_2 is the following. He could show us jump computations with uses $\tau_1, \tau_2, \cdots, \tau_N$ in the first lot, and then switch $g(v,\cdot)$ from 0 to 1. He will then make $A_s \supset \tau_N$, then $A_t \supset \tau_{N-1}$, and so on. Each time he waits for us to trace $J^{\tau_i}(e)$ before moving on to make $A_s \supset \tau_{i-1}$. We would be forced to put all the values from this lot of jump computations into X_e, and all the opponent lost was that he had switched $g(v,\cdot)$ just once. Remember that he gets to decide how large N is, and so he can easily defeat us if A is a general Δ^0_2 set (or even if A is ω-c.e.). How do we salvage this proof, if A is 2-c.e.? The opponent can still play the game above. That is, he can make A extend $\tau_1, \tau_2, \cdots, \tau_N$, and then reverse the order and show us $\tau_N, \tau_{N-1}, \cdots, \tau_1$. However, after he does that, he *cannot go back up again.* That is, he cannot show us $A \supset \tau_i$ for any $i > 1$ anymore (by Fact 2.2.2).

The solution therefore is to assume control of a second index v_1, for the purpose of testing A when the opponent returns back to the previous uses. That is, we wait until he has shown us $\tau_1, \tau_2, \cdots, \tau_N$, and switched $g(v,\cdot)$ once, before we do anything with the index v_1. If he keeps $A \supset \tau_N$, then we set $J^{\tau_N}(v_1)\downarrow$ and wait for him to respond with a $g(v_1,\cdot)$ switch. The only thing the opponent can do, is to either switch $g(v_1,\cdot)$ (in which case we can trace $J^{\tau_N}(e)$ into X_e) or else he has to move A away from τ_N. In the latter case suppose he now makes $A \supset \tau_{N-1}$. In this case we would have made progress in our strategy, because we hadn't put anything unnecessary into X_e, and we know that he cannot go back and make $A \supset \tau_N$ ever again. So, we could now set $J^{\tau_{N-1}}(v_1)\downarrow$ and repeat. When he finally reaches τ_1, the total number of values we would have put in X_e would be at most $h(v_1)$, but no more from this lot of computations since he cannot go back up again.

Note that at any time he could abandon this first lot of computations and show us a second lot $\tau'_1, \cdots, \tau'_{N'}$ incompatible with the first lot. We would wait for him to change $g(v, \cdot)$ from $1 \to 0 \to 1$, and then repeat the above strategy using a new index v_2 on this second lot. The overall bound on X_e would be $h(v_1) + \cdots + h(v_{h(v)})$. The idea is fairly clear when $n = 2$. For a general n we need n different layers of indices (when $n = 2$ we had only two layers). Informally we think of using the first layer when going up in the order $\tau_1, \cdots, \tau_N$, the second layer when coming down in the order $\tau_N, \tau_{N-1}, \cdots, \tau_1$, the third layer when going up again from $\tau_1, \cdots, \tau_N$, and so forth. Each layer nests its strategy within a predecessor index belonging to the previous layer, and assumes A extends the lot of computations specified by the predecessor. We only ever trace anything into X_e, if some index on the bottommost layer responds. The idea is that we only put $J^\tau(e)$ into X_e if we had witnessed $A \supset \tau$ on n different occasions. To organize these ideas, we use a tree of indices.

The tree of indices: to make use of the superlowness of A, we will be enumerating Turing functionals, and by the Recursion Theorem we may assume that we know the values of the indices in advance. In order to help us to trace $J^A(e)$ (each e works independently), we will need n layers of indices. We represent this on a finite tree T^e, which is a collection of finite strings in $\omega^{<\omega}$ of height $n-1$. Each node $\sigma \in T^e$ is associated with an index v_σ. At level 0, we have the empty node λ, and the associated index v_λ. Suppose σ of length $k < n-1$ is defined, with associated index v_σ. There will be $h(v_\sigma)$ many successors of σ on the tree, and we associate indices $v_{\sigma^\frown 1}, \cdots, v_{\sigma^\frown h(v_\sigma)}$ with these nodes. We will construct a c.e. trace $\{X_e\}_{e \in \mathbb{N}}$ for J^A, and ensure that for all e, $|X_e| \leq \tilde{h}(e) = \sum_{\sigma \in T^e} h(v_\sigma)$. We enumerate axioms for $J^X(v_\sigma)$ for all $\sigma \in T^e$, which will help us get certification that a certain segment of A is correct. The actual value of $J^X(v_\sigma)$ does not matter since $g(v_\sigma, \cdot)$ only predicts convergence or divergence, so we will represent the functional by a prefix-free set of finite strings V_σ, representing the use. That is, we enumerate τ into V_σ if we mean to define $J^\tau(v_\sigma) \downarrow$. A slight technical point: the set V_σ will be built as a disjoint union of subsets $V_\sigma = V_{\sigma,1} \cup \cdots \cup V_{\sigma, h(v_\sigma)}$. The idea is that after $g(v_\sigma, \cdot)$ has changed its mind k times, we start enumerating strings into $V_{\sigma,k+1}$. The set $V_{\sigma,k}$ is the environment where the successor node $\sigma^\frown k$ works in.

The construction of X_e: each X_e is constructed independently, but in a uniform manner. We ensure that at all times

$$(\dagger) \text{ if } \sigma_1^\frown k \subseteq \sigma_2, \text{ then } V_{\sigma_2} \subseteq V_{\sigma_1,k}.$$

At stage $s = 0$ we make $V_{\sigma,k} = \emptyset$ for all σ and k. Suppose $s > 0$. We use the tree T^e as a reference to guide our actions at the current stage s. We let $\Diamond$ point at a certain node on T^e, and decide if $\Diamond$ currently needs action. At the start of each stage s we set $\Diamond = \lambda$, and $\Diamond$ will work its way down T^e during the stage, according to the following:

1. $g(v_\Diamond, s) = 0$: we see if $J^A(e)[s] \downarrow$. If so, place the use τ into $V_{\Diamond,ch(v_\Diamond,s)}$, unless τ is already in $V_\Diamond$. In any case, halt and proceed to the next stage.

2. $g(v_\Diamond, s) = 1$ *and* A_s *extends no* $\tau \in V_\Diamond$: we do nothing, halt and proceed to the next stage.

3. $g(v_\Diamond, s) = 1$ *and* $A_s \supset \tau$ *for some* $\tau \in V_{\Diamond,k}$: if $|\Diamond| = n - 1$, we are at the lowest level of T^e, and we can believe $J^\tau(e)$. Enumerate the value $J^\tau(e)$ into X_e. Halt and proceed to the next stage. Otherwise if $|\Diamond| < n - 1$ we reassign $\Diamond$ to be $\Diamond^\frown k$, i.e. go to the appropriate successor. Repeat the above with the new value for $\Diamond$.

Verification: since each V_σ copies the uses of convergent $J^A(e)[s]$-computations, it is clearly a prefix-free set of strings. Also $V_{\sigma,1}, \cdots, V_{\sigma,h(v_\sigma)}$ are clearly pairwise disjoint, and $(\dagger)$ holds. Since $ch(v_\sigma) \leq h(v_\sigma)$ holds at all time, it follows that we do not make any illegal assignments during the construction. We start with a crucial lemma. Basically it says that suppose the string τ_1 enters some $V_{\sigma,k}$ at the bottom level of the tree T^e before the string τ_2 does. Then, after the string τ_2 enters $V_{\sigma,k}$, we can never return to having $A_s \supset \tau_1$. This is where we have to use the fact that A is n-c.e.

Lemma 2.2.3. *Fix a number k and a string σ such that $|\sigma| = n-1$. Suppose $\tau_1 \neq \tau_2$ are placed in $V_{\sigma,k}$ at stages $u < u'$ respectively. Then for all $t > u'$, we cannot have $A_t \supset \tau_1$.*

Proof. Let $\sigma = k_1 k_2 \cdots k_{n-1}$, and $\sigma_i = \sigma \upharpoonright i$. Note that by $(\dagger)$, τ_1 and τ_2 must both be in $V_{\sigma_i,k_{i+1}}$ for all $i = 0, \cdots, n-2$ (i.e. all the predecessors of σ). Hence it follows that there are stages $s_0 < s_1 < \cdots < s_{n-2} < u$, as well as stages $t_0 < t_1 < \cdots < t_{n-2} < u'$ such that τ_1 enters $V_{\sigma_i,k_{i+1}}$ at stage s_i and τ_2 enters $V_{\sigma_i,k_{i+1}}$ at stage t_i.

We now argue that in fact we must have $\max\{s_i, t_i\} < \min\{s_{i+1}, t_{i+1}\}$. Fix an $i < n-2$ and let $\tilde{s} = \min\{s_{i+1}, t_{i+1}\}$. Observe that no further enumeration is made into $V_{\sigma_i,k_{i+1}}$ on or after stage $\tilde{s}$, because at stage $\tilde{s}$ we have $V_{\sigma_i,k_{i+1}}[\tilde{s}] \neq \emptyset$ as well as $g(v_{\sigma_i}, \tilde{s}) = 1$, and $ch(v_{\sigma_i}, \tilde{s}) \geq k_{i+1}$. Any further enumeration into V_{σ_i} has to wait until $g(v_{\sigma_i}, \cdot)$ flips to 0 again, whence $ch(v_{\sigma_i})$ will become $> k_{i+1}$. Hence, both τ_1 and σ_1 has to already be in $V_{\sigma_i,k_{i+1}}$ at stage $\tilde{s}$.

Now, choose an appropriate sequence of stages s such that we alternate between $A_s \supset \tau_1$ and $A_s \supset \tau_2$. There are altogether n alternations, and by Fact 2.2.2 there cannot be any more. $\square$

We want to show that $|X_e| \leq \tilde{h}(e)$. We will be done if we show that for each σ on T^e such that $|\sigma| = n-1$ and each k, we have $|\tau : \tau \in V_{\sigma,k} \ \wedge \ J^{\tau}(e) \in X_e| \leq 1$. However, this fact follows directly from Lemma 2.2.3. Finally we want to argue that if $J^A(e) \downarrow= q$ with use τ, then $q \in X_e$. Define the node σ of length $|\sigma| = n-1$ inductively such that for every $i \leq n-1$, we let $\sigma_i = \sigma\restriction i$ and we have the properties

1. $\lim_s g(v_{\sigma_i}, s) = 1$,

2. $\tau \in V_{\sigma_i,\sigma(i)}$,

3. for almost all stages t, $\Diamond$ will visit σ_i at stage t.

Assume σ_i has the above properties and $\sigma_{i+1} = \sigma_i^\frown\sigma(i)$ is defined. We continue the induction by showing σ_{i+1} has the properties above, and also define $\sigma(i+1)$. Clearly at every stage after $A\restriction|\tau|$ settles (say it settles by stage u), we will have $\Diamond$ visiting σ_{i+1}. We cannot have $\lim_s g(v_{\sigma_{i+1}}, s) = 0$ because otherwise $\tau \in V_{\sigma_{i+1}}$ which implies that g predicts A' incorrectly. Hence $\lim_s g(v_{\sigma_{i+1}}, s) = 1$, and it follows that τ must eventually be enumerated in $V_{\sigma_{i+1}}$, otherwise we must have $J^A(v_{\sigma_{i+1}}) \uparrow$ which means that $\lim_s g(v_{\sigma_{i+1}}, s) = 1$ is incorrect. Therefore $\tau \in V_{\sigma_{i+1},k}$ for some k, and let $\sigma(i+1) = k$. This completes the induction. Finally, since σ exists with the above properties, it follows that $J^{\tau}(e) = q$ will be eventually enumerated into X_e.

2.3 There is no minimal bound for jump traceability

We show that there is no minimal bound for jump traceability, answering a question of Figueira, Nies and Stephan [FNS06]. In particular, there is no single order function such that strong jump traceability is the same as jump traceability for that order.

Theorem 2.3.1. *For any given order function h, there is a c.e. set A and an order function $\tilde{h}$, such that A is jump traceable via h, but not jump traceable via $\tilde{h}$.*

2.3.1 Requirements

We build a c.e. set A, and a trace $\{V_e\}_{e\in\mathbb{N}}$ for $J^A(e)$ satisfying the following requirements:

$$\begin{aligned} \mathcal{N}_e &: \text{ Trace } J^A(e) \text{ into } V_e, \text{ with } |V_e| \leq h(e),\\ \mathcal{P}_e &: \text{ For some } x, \text{ either } |T^e_x| > e, \text{ or else } J^A(x) \notin T^e_x. \end{aligned}$$

Here, we let $\{T^e_x\}_{x\in\mathbb{N}}$ be the e^{th} trace in some effective enumeration of all traces $\{T^0_x\}_{x\in\mathbb{N}}, \{T^1_x\}_{x\in\mathbb{N}}, \cdots$, and $J^A(e)[s]$ be the value of the universal A-jump function $\{e\}^A(e)[s]$ at stage s. We let the use of $J^A(e)[s]$ (if convergent) be $j(e,s)$. Note that if h is an order function, we always assume that its range takes positive values. When we say that we pick a *fresh* number x at stage s, we mean that we choose x to be the least number $x > s$, and $x >$ any number used or mentioned so far.

2.3.2 Description of strategy

We have two types of requirements to handle - the negative requirements $\mathcal{N}_e$ which want to impose a restraint on A each time it sees a computation $J^A(e)[s]\downarrow$, and the positive ones $\mathcal{P}_e$ which makes enumerations into A to force a particular trace T^e_x to fill up.

The way that a particular $\mathcal{N}_e$ works is described below. Since we have to ensure that $|V_e| \leq h(e)$, we have to make sure that restraints imposed by $\mathcal{N}_e$ is not obeyed on at most $h(e) - 1$ many occasions. It is convenient to think of V_e as being made up of boxes, which we will fill with values (which are current values of $J^A(e)[s]$). By the time we use up all of the $h(e)$ many boxes allowed for V_e, we have to ensure that the last value we enumerated is correct. We start the construction with $\mathcal{N}_e$ initially being a $(h(e)-1)$-box. (This is represented in the formal construction by the variable $size(e,s)$, and represents the number of injuries $\mathcal{N}_e$ can still sustain). We also say that $\mathcal{N}_e$ is an *original* $(h(e)-1)$*-box.* Every time the restraint that $\mathcal{N}_e$ is holding is breached by some positive action, we would decrease $size(e,s)$ by 1, since a new value for $J^A(e)$ might appear in future. By the time $\mathcal{N}_e$ becomes a 0-box (corresponding to $|V_{e,s}| = h(e)$), we will have to ensure that no enumerations made in future can destroy the current $J^A(e)[s]$ computation. That is, the restraint imposed by $\mathcal{N}_e$ while it is a 0-box must be obeyed by every positive requirement.

Let us now turn our attention to the positive requirement $\mathcal{P}_e$. We describe how to meet such a requirement. $\mathcal{P}_e$ would attempt to defeat the e^{th} trace by doing the following. It will control the value of the universal jump function $J^A(x)$ of A at some location x. The Recursion Theorem supplies us with such an index x, and $\mathcal{P}_e$ will enumerate axioms into the Turing functional Ψ_e^A with index x, with use $u(e,s)$ on A. Each time the value $\Psi_e^A(x)[s]$ shows up in the trace T_x^e, we would put the use $u(e,s)$ into A to cancel all the previous axioms, and enumerate a new axiom $\langle x, y, A_s \upharpoonright u(e,s)\rangle$ for $\Psi_e^A(x)$ (with a fresh y). After doing this at most e times, we would be able to meet the requirement $\mathcal{P}_e$: recall that we have to build an order function $\tilde{h}$ globally, and all we have to do is to ensure that we define $\tilde{h}(x) = e$. On the other hand, the negative requirements would be imposing various restraints on $\mathcal{P}_e$, as described in the previous paragraph, for the sake of making A superlow. At times we would have to initialize $\mathcal{P}_e$, due to these restraints. For instance, if some $\mathcal{N}_k$ is in a state of being a 0-box (these boxes have the highest priority), with restraint larger than $u(e,s)$, then we would have to make $\mathcal{P}_e$ abandon the current index, and begin to enumerate a new functional with a new index x'. To ensure the success of $\mathcal{P}_e$, we would have to make sure that it is initialized only finitely often. In fact, to guarantee that $\tilde{h}$ is computable, we have to know in advance a bound for the number of times that $\mathcal{P}_e$ will need to be initialized. This is because we could then know how many different indices to set aside for $\mathcal{P}_e$, and hence define $\tilde{h} = e$ on these indices.

The construction will only require finite injury, with dynamic assignment of priority amongst the requirements. As we will see, the main obstacle we are facing is in having to arrange priority between the positive and negative requirements, such that we can limit the number of initializations to each $\mathcal{P}_e$ to an amount that can be pre-determined. Let us consider the case when the given h satisfies $h(0) = h(1) = h(2) = h(3) = 1$ and $h(4) = 3$. Note that in general, if the given h grows very slowly, then it becomes much harder for numbers to enter A because there are more small-sized boxes to consider. Consider a requirement $\mathcal{P}$ that wants to diagonalize some trace by enumerating into A twice. Suppose we arrange the requirements in the order:

$$\mathcal{N}_0(h=1) < \mathcal{N}_1(h=1) < \mathcal{N}_2(h=1) < \mathcal{N}_3(h=1) < \mathcal{P} < \mathcal{N}_4(h=3).$$

For $\mathcal{P}$ to succeed at a particular index x, its cycle for that x has to be:

> *Phase 1*: Set $J^A(x)[s_1]\downarrow$. Wait for the corresponding value to show up in the trace. If it does, put the use into A to reset $J^A(x)$.
>
> *Phase 2*: Set $J^A(x)[s_2]\downarrow$ again and wait for the value to show up in the trace. When it does, put the use into A to reset $J^A(x)$, set a new axiom for $J^A(x)$, and we are done.

If $\mathcal{P}$ gets blocked in phase 2, it will be initialized and will have to *start with a new index x' in phase 1.* Why is this a problem in the above example? When $\mathcal{P}$ is in phase 1, it will have a follower appointed pointing at A, which it will put in A when realized. But in the meantime we might have $\mathcal{N}_4$ imposing an A-restraint above the $\mathcal{P}$-follower. This is due to the fact that $\mathcal{N}_4$ has seen $J^A(4)$ converge with a large A-use, and $\mathcal{N}_4$ has put that value into the trace we are building for $J^A(4)$.

Suppose next, the $\mathcal{P}$-follower gets realized. It will then enumerate the $\mathcal{P}$-follower it has appointed and enter phase 2, injuring $\mathcal{N}_4$ in the process. Remember that $\mathcal{N}_4$ is allowed 2 mistakes (i.e. it is an original 2-box), and now it has used up one of them. Therefore, in future it is only allowed 1 more mistake (i.e. it has now been *promoted* to a 1-box).

$\mathcal{P}$ is now waiting in phase 2 for its follower to be realized. It might be the case that $\mathcal{N}_0$ now imposes A-restraint larger than $\mathcal{P}$'s follower, forcing $\mathcal{P}$ to be initialized and start again in phase 1. This looks bad, because the process could be repeated with $\mathcal{N}_1$ in the same manner, and $\mathcal{N}_4$ can be promoted yet again, now to a 0-box.

When $\mathcal{N}_4$ next imposes A-restraint, being a 0-box, its restraint has to be obeyed by everyone, including $\mathcal{P}$ above it. Again we could create any number of 0-boxes in this way, and in turn use them to produce even more 0-boxes further down the list of requirements, and we are faced with the same problem.

The solution is to arrange priority between $\mathcal{P}$ and the negative requirements dynamically. This priority ordering depends on whether $\mathcal{P}$ is in first phase, or in second phase. If $\mathcal{P}$ is in the first phase, we place $\mathcal{P}$ above (stronger priority than) all $\mathcal{N}_e$ which are currently at least 2-boxes, and place $\mathcal{P}$ below (weaker priority than) all $\mathcal{N}_e$ which are currently 0 or 1-boxes. If $\mathcal{P}$ is in the second phase then we place it above all $\mathcal{N}_e$, other than those that are currently 0-boxes. At the beginning of the construction, before anything is done, we have the ordering:

$$\underbrace{\mathcal{N} < \cdots}_{\text{0-boxes},h=1} < \underbrace{\mathcal{N} < \cdots}_{\text{1-boxes},h=2} < \mathcal{P}_{\text{ (phase 1)}} < \underbrace{\mathcal{N} < \cdots}_{\text{2-boxes},h=3} < \underbrace{\mathcal{N} < \cdots}_{\text{3-boxes},h=4} < \cdots$$

When $\mathcal{P}$ enters phase 2, the situation becomes

$$\underbrace{\mathcal{N} < \cdots}_{\text{0-boxes},h=1} < \mathcal{P}_{\text{ (phase 2)}} < \underbrace{\mathcal{N} < \cdots}_{\text{1-boxes},h=2} < \underbrace{\mathcal{N} < \cdots}_{\text{1-boxes},h=3} < \underbrace{\mathcal{N} < \cdots}_{\text{2-boxes},h=3}$$
$$< \underbrace{\mathcal{N} < \cdots}_{\text{2-boxes},h=4} < \underbrace{\mathcal{N} < \cdots}_{\text{3-boxes},h=4} < \cdots$$

If $\mathcal{P}$ gets initialized while in phase 2 due to one of the 0-boxes, the ordering becomes

$$\underbrace{\mathcal{N} < \cdots}_{\text{0-boxes},h=1} < \underbrace{\mathcal{N} < \cdots}_{\text{1-boxes},h=2} < \underbrace{\mathcal{N} < \cdots}_{\text{1-boxes},h=3} < \mathcal{P}_{\text{ (phase 1)}} < \underbrace{\mathcal{N} < \cdots}_{\text{2-boxes},h=3}$$
$$< \underbrace{\mathcal{N} < \cdots}_{\text{2-boxes},h=4} < \underbrace{\mathcal{N} < \cdots}_{\text{3-boxes},h=4} < \cdots$$

We claim that this solves the problem, namely that we can count the number of times $\mathcal{P}$ is forced to be initialized. The ability to perform this counting is essential for $\tilde{h}$ to be computable. Firstly, note that *no new* 0*-boxes are ever created*, unless $\mathcal{P}$ is permanently satisfied at the same time. That is, the only 0-boxes present are those original ones - namely, those $\mathcal{N}$ with $h = 1$.

The counting of injuries to $\mathcal{P}$*:* whilst in the second phase, $\mathcal{P}$ can only by initialized by a 0-box, we have already observed that these must be original 0-boxes. In the first phase, $\mathcal{P}$ would be initialized

1. either by some $\mathcal{N}$ with $h = 1$ or 2 (i.e. the original 0 and 1-boxes), or

2. a promoted 1-box.

Suppose case 2 happens at stage s. The only reason why a 2-box is promoted to a 1-box, is because it was injured by $\mathcal{P}$ and $\mathcal{P}$ moved from the first phase to the second phase, at some previous stage $t < s$. But now at stage s, $\mathcal{P}$ is back in the first phase, which means that at some time between t and s, $\mathcal{P}$ must have been initialized while in phase 2. This can only be done by some $\mathcal{N}$ with $h = 1$, i.e. one of the original 0-boxes, since these are the only requirements stronger than $\mathcal{P}$ in phase two. This means that the largest k such that some $\mathcal{N}$ with $h = k + 1$ (original k-box), is ever promoted to a 1-box, is at most $S := 2 + h^{-1}(1)$, where $h^{-1}(i) = \#$ of y such that $h(y) = i$. That is, if $k > S$ then no $\mathcal{N}$ which is originally a k-box, can ever get promoted to a box size of 1. Therefore, the number of times that $\mathcal{P}$ can be injured, has bounds of $h^{-1}(1)$ from phase 2, and $\sum_{i\leq S+1} h^{-1}(i)$ while in phase 1.

2.3.3 The general strategy

The requirement $\mathcal{P}_e$ will need to enumerate e many times without being initialized; its action will be divided into e many phases. In the discussion above, the $\mathcal{P}$ being considered is just $\mathcal{P}_2$. In the example above we had the three numbers $C^1_0 = 0$, $C^2_0 = 1$ and $C^2_1 = S$, which are called *thresholds*[1]. These are the critical numbers which we use to determine priority between $\mathcal{P}_2$ and the negative requirements. In phase 1, $\mathcal{P}_2$ would be injured by C^2_0-boxes (and smaller ones). In phase 2, $\mathcal{P}_2$ would be injured by C^1_0-boxes. To prevent the different positive requirements from interfering with each other, we ensure that no new C^2_1-boxes are ever created by the actions of $\mathcal{P}_3, \mathcal{P}_4, \cdots$. Thus, $\mathcal{P}_3$ would be injured by C^3_1-boxes in phase 1, by C^3_0-boxes in phase 2, and by C^2_1-boxes in phase 3. The values C^3_0, C^3_1, C^3_2 are defined inductively.

Each time $\mathcal{P}_e$ is initialized, its threshold would be reset to C^e_{e-2}. With each enumeration that $\mathcal{P}_e$ makes, we will decrease the threshold accordingly $(C^e_{e-3}, \cdots, C^e_0, C^{e-1}_{e-2})$. When moving from phase 1 to 2, a $(C^e_{e-2} + 1)$-box can be promoted to a C^e_{e-2}-box but this newly promoted box cannot initialize $\mathcal{P}_e$ while in phase 2 or higher, for its threshold has now decreased to C^e_{e-3} or less. The only way to initialize $\mathcal{P}_e$ after it

[1]These values are chosen for discussion purposes, and are slightly different in the formal construction. This is because we had not taken into account the other positive requirements.

has made m enumerations, would be through a C^e_{e-2-m}-box (or less). So as long as we keep the critical thresholds values $C^{e-1}_{e-2}, C^e_0, \cdots, C^e_{e-2}$ sufficiently spaced out, we will be alright.

2.3.4 Notations for the formal construction

Let $size(e,s)$ denote the size of the $\mathcal{N}_e$-box at stage s, i.e. the number of injuries that $\mathcal{N}_e$ can still sustain. At the beginning of the construction, $size(e,0)$ is set to $h(e)-1$. Each time $\mathcal{N}_e$ is injured, we reduce $size(e,s)$ by 1, and when $size(e,s)$ reaches 0, the $J^A(e)$-computation will have to be preserved forever. During the construction, all parameters retain their assigned values, unless they are initialized or reassigned a different value. We append $[s]$ to an expression to refer to the evaluation of the expression at stage s. On the other hand if the context is clear we drop the stage number from the expression. For example $size(e,s)$ becomes simply $size(e)$.

For each n, let $h^{-1}(n) := \{x \in \mathbb{N} \mid h(x) = n\}$, where h is the order function given. Since h is an order, the set $h^{-1}(n)$ and the value $|h^{-1}(n)|$ are both computable for each n. For each e, let $p(e,s)$ be a counter which records the number of different values the requirement $\mathcal{P}_e$ has managed to put into the trace $T^e_{x(e,s)}$, for the current $x(e,s)$. Note that $p(e,s)+1$ is also the phase number of $\mathcal{P}_e$, as used in the discussion in section 2.3.2. Let $S(n) = \sum_{r=1}^{n} r|h^{-1}(r)|$. That is, $S(n)$ denotes the maximum number of different values $j(k,s)$ can take, for the set of k's such that $h(k) \leq n$. This number $S(n)$ is used to give a bound on the number of times a particular $\mathcal{P}_e$ can get injured by some $\mathcal{N}_k$ which is an original $(n-1)$-box or less.

We now define the sequence of numbers $\{C^n_k \mid n \geq 0 \ \wedge \ k < n\}$ and the functions $|\mathcal{P}_n|, \tilde{I}_n, I_n : \mathbb{N} - \{0\} \mapsto \mathbb{N}$ as follows. We start with $C^0_{-1} = 0$, and in general for

$n \geq 1$, we have:

$$\begin{aligned}
\tilde{I}_n &= n + \sum_{r=1}^{n-1} r I_r, \\
C_0^n &= (C_{n-2}^{n-1} + 2) + \sum_{r=1}^{n-1} |\mathcal{P}_r|, \\
&\vdots \\
C_{k+1}^n &= (C_k^n + 2) + \sum_{r=1}^{n-1} |\mathcal{P}_r| + n\tilde{I}_n + nS(C_k^n), \\
I_n &= S(C_{n-1}^n) + \tilde{I}_n, \\
|\mathcal{P}_n| &= (n-1)I_n + 1.
\end{aligned}$$

We fix the convention $C_{-1}^n = C_{n-2}^{n-1}$. We will explain what each of the symbols represent, for $n \geq 1$. The sequence $C_{-1}^n, C_0^n, \cdots, C_{n-2}^n$ are the different critical threshold values when $\mathcal{P}_n$ is in phase $n, n-1, \cdots, 1$ respectively. The number C_{n-1}^n represents the largest number $b+1$ such that a b-box can interfere with the action of $\mathcal{P}_n$. We obviously do not want $\mathcal{P}_{n+1}$ to promote any box to a b-box, to keep the effects of different $\mathcal{P}$ disjoint; for this reason the critical threshold values of $\mathcal{P}_{n+1}$ are all larger than $b+1$. I_n represents the total number of times $\mathcal{P}_n$ may be initialized, while $\tilde{I}_n$ is the number of times that $\mathcal{P}_n$ may be initialized by the actions of a higher priority $\mathcal{P}_r$. Lastly, $|\mathcal{P}_n|$ is the maximum number of enumerations $\mathcal{P}_n$ may make into A; it may make at most $n-1$ enumerations before it is initialized, plus one more to make it permanently satisfied.

To control the value of the universal jump function $J^A(x)$ of A at different locations x, we will have an infinite list of indices (of Turing functionals) supplied by the Recursion Theorem. We let Ψ_e^A denote the functional that the requirement $\mathcal{P}_e$ is enumerating at stage s, with index $x(e,s)$ chosen from the list. The value of $x(e,s)$ will change from time to time, more specifically, whenever the requirement $\mathcal{P}_e$ is initialized. At these stages, $\mathcal{P}_e$ will start the enumeration of a new functional, with a new index taken from the list. The use of the functional with index $x(e,s)$ that $\mathcal{P}_e$ is enumerating at stage s is denoted by $u(e,s)$. Thus $u(e,s)$ is a number targeted at A, which we will have to put into A before we can enumerate a new axiom into Ψ_e^A.

In Lemma 2.3.4(iii) we show that the number of times $\mathcal{P}_e$ can be initialized is bounded by I_e. Thus to define the function $\tilde{h}$, we do the following. We reserve the first $I_1 + 1$ many indices for use by $\mathcal{P}_1$, the next $I_2 + 1$ many indices for $\mathcal{P}_2$, and so on. The function $\tilde{h}$ is defined to have value e on all the indices reserved for $\mathcal{P}_e$. Hence $\tilde{h}$ is clearly an order, and by Lemma 2.3.4, $\mathcal{P}_e$ only uses indices $x(e)$ which are reserved for it, i.e. it never runs out of indices. Hence, $\tilde{h}(\lim_s x(e,s)) = e$, and it follows that A is not jump-traceable via $\tilde{h}$.

When we *initialize* a requirement $\mathcal{P}_e$ at stage s, we do the following. Do nothing if $p(e,s) = e$ (in this case, $\mathcal{P}_e$ has already been permanently satisfied, and needs to nothing else). Otherwise, set $x(e, s+1)$ to be the next value reserved for $\mathcal{P}_e$, and reset the counter $p(e, s+1)$ to 0.

Definition 2.3.2. We say that a requirement $\mathcal{P}_e$ *requires attention at stage s*, if $p(e,s) < e$ and one of (ATT1)-(ATT3) holds.

> (ATT1) $\mathcal{P}_e$ has been initialized at some stage $t < s$, and has not received attention[2] at any stage u such that $t < u < s$.
>
> (ATT2) All of the following hold:
>
> - There is a computation in Ψ_e^A which currently apply with use $u(e,s)$ and value $r = \Psi_e^A(x(e,s))[s]$,
> - For some $k < s$, we have $J^A(k)[s]\downarrow$ with use larger than $u(e,s)$,
> - If $e = 1$, we find that $size(k,s) = 0$. Otherwise for $e > 1$, we find that
>
> $$size(k,s) \le \begin{cases} C^e_{e-2-p(e,s)}, & \text{if } p(e,s) \le e-2, \\ C^{e-1}_{e-2}, & \text{if } p(e,s) = e-1. \end{cases}$$
>
> (ATT3) There is a computation in Ψ_e^A which currently applies with use $u(e,s)$ and value $r = \Psi_e^A(x(e,s))[s]$, such that r has shown up in the trace $T^e_{x(e,s)}$.

If (ATT1) holds, then $\mathcal{P}_e$ has just been initialized, and we need to set $\mathcal{P}_e$ on a new index $x(e)$. If (ATT2) holds, the restraint on $\mathcal{P}_e$ has increased beyond $u(e,s)$, and we would need to initialize it. This would be due to some high priority box blocking $\mathcal{P}_e$. If (ATT3) holds, then the follower $u(e,s)$ has been realized, and we will need to take a positive action to defeat the trace $\{T^e_x\}_{x\in\mathbb{N}}$.

[2]This will be defined in the construction.

2.3.5 Construction of A and $\{V_e\}_{e\in\mathbb{N}}$

At stage $s = 0$, initialize all requirements and set $size(e, 0) = h(e) - 1$ for all e. At stage $s > 0$, we do the following:

- For all $e < s$ such that $J^A(e)[s]\downarrow$, we enumerate the value $J^A(e)[s]$ into V_e.
- Pick the smallest $e < s$ such that $\mathcal{P}_e$ requires attention at stage s. Take the appropriate action listed below, and declare that $\mathcal{P}_e$ has *received attention* at stage s.

  1. *(ATT1) holds*: we enumerate a computation $\Psi_e^A(x(e, s))[s]\downarrow = s$ with use $u(e, s) > s$ and $u(e, s) >$ any value previously chosen as a use.
  2. *(ATT1) fails and (ATT2) holds*: initialize requirement $\mathcal{P}_k$ for all $k \geq e$.
  3. *(ATT1) and (ATT2) fails but (ATT3) holds*: do all of the following.
     - For each $k < s$ such that $J^A(k)[s]\downarrow$ with use $j(k, s) > u := u(e, s)$, we decrease $size(k, s)$ by 1; these are the boxes which will be promoted.
     - Enumerate u into A to clear the Ψ_e^A-axioms.
     - Increase $p(e, s)$ by 1 to enter the next phase.
     - Define a new computation $\Psi_e^A(x(e))[s]\downarrow = s$ with fresh use $u(e, s)$.
     - Initialize all requirements $\mathcal{P}_k$ for $k > e$.

2.3.6 Verification

In the first lemma we show that the maximum number of different restraints held by any original b-box for any $b < n$ is at most $S(n)$. This is needed for various counting arguments to follow. If X is a set and k is a number then $X\restriction k$ denotes the string $X(0)\cdots X(k)$.

Lemma 2.3.3. *(i) For each k, $|\{A_s\restriction j(k, s) : J^A(k)[s]\downarrow \ \wedge \ k < s\}| \leq h(k)$.*

(ii) For each n, $|\{\langle A_s\restriction j(k, s), k\rangle : J^A(k)[s]\downarrow \ \wedge \ h(k) \leq n \ \wedge \ k < s\}| \leq S(n)$.

(iii) For all e, $\mathcal{N}_e$ is satisfied. Hence, A is jump traceable via h.

Proof. (i): Suppose on the contrary, there are stages $s_0 < \cdots < s_{h(k)}$ all larger than the number k, such that $A_{s_i}\restriction j(k,s_i) \neq A_{s_{i+1}}\restriction j(k,s_{i+1})$ for all $i = 0,\cdots,h(k)-1$. For each i, the change $A_{s_i}\restriction j(k,s_i) \neq A_{s_{i+1}}\restriction j(k,s_{i+1})$ must have been caused by some positive requirement receiving attention at some stage t in which we also decrease $size(k,t)$, where $s_i \leq t < s_{i+1}$. This means that by the time we reach stage $s_{h(k)-1}$, we have $size(k, s_{h(k)-1}) = 0$, putting it at the highest priority. Hence no enumeration can be made below $j(k, s_{h(k)-1})$ at any stage $t \geq s_{h(k)-1}$, a contradiction.

(ii): By part (i).

(iii): Fix an e, and we want to argue that $\mathcal{N}_e$ is satisfied. If $J^A(e)\downarrow$ then clearly it must be enumerated into V_e after stage e. Suppose that $|V_e| > h(e)$, then $|\{J^A(e)[s] : s > e\}| > h(e)$. Each different value of $J^A(e)[s]$ corresponds to a different string $A_s\restriction j(k,s)$ for the use, contradicting (i). □

The next lemma is the crucial lemma; it argues that the combinatorics we used work out fine. In particular, we argue that no original C^n_k-box can ever be promoted to a C^n_{k-1}-box.

Lemma 2.3.4. *Let $n > 0$.*

(i) If $m > C^n_0$, then for all $k \in h^{-1}(m)$ and all s, $size(k,s) > C^{n-1}_{n-2}$.

(ii) For each $1 \leq r \leq n-1$, if $m > C^n_r$, then for all $k \in h^{-1}(m)$ and all s, $size(k,s) > C^n_{r-1}$.

(iii) The total number of initializations to $\mathcal{P}_n$ is bounded by I_n.

Proof. We prove (i)-(iii) simultaneously by induction on n. Suppose the results hold for all $n' < n$. Let $\nu(e,s)$ be the critical value function

$$\nu(e,s) := \begin{cases} C^e_{e-2-p(e,s)}, & \text{if } p(e,s) \leq e-2, \\ C^{e-1}_{e-2}, & \text{if } p(e,s) = e-1. \end{cases}$$

We have the following facts:

(Fact 1): For all $n' < n$, the total number of times an enumeration is made into A by $\mathcal{P}_{n'}$, is bounded by $|\mathcal{P}_{n'}|$.

(Fact 2): The total number of stages t such that the requirement $\mathcal{P}_n$ is initialized at stage t, and $\mathcal{P}_n$ *did not* receive attention at stage t, is at most $\tilde{I}_n$. To see this, observe that at such a stage t, it must be that $\mathcal{P}_{n'}$ receives attention for some $n' < n$. At stage t, either $\mathcal{P}_{n'}$ is initialized as well, or else an enumeration is made into A. The total number of such stages t is at most $\sum_{r=1}^{n-1} I_r + \sum_{r=1}^{n-1} |\mathcal{P}_r| < \tilde{I}_n$.

(i): Suppose on the contrary that there is some $m > C_0^n$ such that $size(k,s) \leq C_{n-2}^{n-1}$ for some s and some $k \in h^{-1}(m)$. We may assume that $size(k,s) = C_{n-2}^{n-1}$, since the parameter $size(k)$ is never decreased by more than one at any stage. Since $size(k,0) \geq C_0^n$, there is a stage $s' < s$ such that $size(k,s') = C_0^n$. Suppose $t \geq s'$ is a stage where some positive requirement $\mathcal{P}_e$ receives attention in which $size(k,t)$ is decreased. It cannot be that $e > n$, because $\nu(e,t) \geq C_0^n \geq size(k,t)$; $\mathcal{P}_e$'s actions are kept from interfering with $\mathcal{P}_n$'s. If $e = n$, then $p(e,t)$ has to be $e-1$ and hence there can only be at most one such stage t where $\mathcal{P}_n$ receives attention and decreases $size(k,t)$ (since $p(e,t)$ is increased to e at stage t, and $\mathcal{P}_n$ becomes permanently satisfied after that). Hence, the number of times $size(k,t)$ can be decreased after stage s', is at most $1 + \sum_{r=1}^{n-1} |\mathcal{P}_r|$. This shows that the smallest value $size(k,t)$ can take at any stage $t \geq s'$, is $C_{n-2}^{n-1} + 1 > size(k,s)$, a contradiction.

(ii): The proof is more or less similar to (i), with more cases involved. Suppose on the contrary that there is some $m > C_r^n$ such that $size(k,s) = C_{r-1}^n$ for some s and some $k \in h^{-1}(m)$. Let $s' < s$ be such that $size(k,s') = C_r^n$. Suppose $t \geq s'$ is a stage where some positive requirement $\mathcal{P}_e$ receives attention in which $size(k,t)$ is decreased. Similar to (i), we will now count the maximum number of such stages t. It is clear that $e \not> n$, and there can be at most $\sum_{r=1}^{n-1} |\mathcal{P}_r|$ many stages t where $\mathcal{P}_e$ receives attention in which $size(k,t)$ is decreased and $e < n$. Lastly if t is a stage where $\mathcal{P}_e$ receives attention, $size(k,t)$ is decreased and $e = n$, we must have $p(e,t) > n - 2 - r$. We split the counting into the following cases:

- *Case 1: t is a stage such that $\mathcal{P}_n$ receives attention in which $size(k,t)$ is decreased and $p(n,t) = n-1$.*

  As in (i), there can only be one such t.

- *Case 2x: t is a stage such that $\mathcal{P}_n$ receives attention in which $size(k,t)$ is* *decreased and $p(n,t) = x$, where $n-1-r \leq x \leq n-2$.*

At stage t, $p(n,t)$ is set to $x+1$. In order that there is a next stage such that Case $2x$ applies again, $\mathcal{P}_n$ will have to be initialized - we can blame this on a small sized box. There are (by Fact 2) at most $\tilde{I}_n$ many times where $\mathcal{P}_n$ can be initialized by some other $\mathcal{P}$. If $\mathcal{P}_n$ is initialized at a stage $t' > t$ where it receives attention, then $p(e,t') \geq x+1 > n-1-r$. Hence there is some $k' < t'$ such that $J^A(k')[t'] \downarrow$, and

$$size(k',t') \leq \begin{cases} C^e_{r-2}, & \text{if } r > 1, \\ C^{e-1}_{e-2}, & \text{if } r = 1. \end{cases}$$

In any case by (i) and induction hypothesis of (ii), we have $h(k') \leq C^e_{r-1}$ (blaming a box of small size), and by Lemma 2.3.3(ii) there can only be at most $\tilde{I}_n + S(C^e_{r-1})$ many t's where Case $2x$ applies.

Putting together the above calculations, we see that the smallest value $size(k,t)$ can take at any stage $t \geq s'$, is $C^n_r - \sum_{r=1}^{n-1} |\mathcal{P}_r| - 1 - n(\tilde{I}_n + S(C^e_{r-1})) = C^n_{r-1} + 1 > size(k,s)$, a contradiction.

(iii): There are (by Fact 2) at most $\tilde{I}_n$ many times where $\mathcal{P}_n$ can be initialized without it receiving attention. If $\mathcal{P}_n$ is initialized at a stage t where it receives attention, there is some $k < t$ such that $J^A(k)[t] \downarrow$, and

$$size(k,t) \leq \begin{cases} C^n_{n-2}, & \text{if } n > 1, \\ C^0_{-1}, & \text{if } n = 1. \end{cases}$$

It follows by (i) and (ii) that $h(k) \leq C^n_{n-1}$, and by Lemma 2.3.3(ii) there can be at most $\tilde{I}_n + S(C^n_{n-1}) = I_n$ initializations to $\mathcal{P}_n$. □

Lemma 2.3.5. *For all e, $\mathcal{P}_e$ is satisfied.*

Proof. Fix an e, and let $x := \lim_{s\to\infty} x(e,s)$, which exists. Suppose that $J^A(x) \downarrow \in T^e_x$. Let $s_0 > e$ be a stage large enough so that we have $x(e,s_0) = x$, $J^A(x) \in T^e_x[s_0]$, and $\mathcal{P}_e$ never requires attention after stage s_0. Hence, it must be the case that $\Psi^A_e(x)[s_0] \downarrow = J^A(x)$, and also that $p(e,s_0) = e$ (else (ATT3) holds at stage s_0). This means that there are e many different stages t (before stage s_0) where $\mathcal{P}_e$ receives attention, and $p(e,t)$ is increased by 1. At each such stage t we also enumerate a new value for $\Phi^A_e(x)[t]$, and wait for it to be traced. By stage s_0, we have $|T^e_x[s_0]| > e$. □

Lemma 2.3.4(iii) actually establishes that $\mathcal{P}_e$ never uses a forbidden index (an index not meant for it). Hence $\tilde{h}(x(e,s)) = e$ for all e and s, and it follows that A is not jump-traceable via $\tilde{h}$. This ends the proof of Theorem 2.3.1. We note that the difference between h and $\tilde{h}$ is very large. A natural question is to ask how different must two computable orders be, before they start to generate different classes of jump traceable sets? In particular,

Question 2.3.6. *If a c.e. set A is jump traceable with respect to $h+1$, must A be jump traceable with respect to h?*

2.4 The c.e. strongly jump traceable sets are Π^0_4-complete

It is easy to see that the index set concerned is Π^0_4:

Lemma 2.4.1. *The set $\{e \in \mathbb{N} : W_e$ is strongly jump traceable$\}$ is Π^0_4.*

Proof. W is strongly jump traceable $\Leftrightarrow \forall e(h_e$ is an order $\Rightarrow \exists k \forall x Q(e,k,x))$, where the predicate Q is

$$Q(e,k,x) = g_k(x)\downarrow \text{ and } |W_{g_k(x)}| \leq h_e(x) \text{ and } J^W(x)\downarrow \Rightarrow J^W(x) \in W_{g_k(x)},$$

and $\{h_e\}_{e\in\mathbb{N}}$ and $\{g_k\}_{k\in\mathbb{N}}$ are effective listing of all partial computable functions. Clearly "h_e is an order" is a Π_2 fact, while $Q(e,k,x)$ is a Π_2 predicate. □

To perform the coding of a Π^0_4 set into the index set of the c.e. strongly jump traceable sets, we will need to use the following lemma; the complete proof can be found in Nies [Nie97]:

Lemma 2.4.2. *If S is a Π^0_4 set, there is a u.c.e. (uniformly computably enumerable) sequence $X_{y,e,p}$ of initial segments of ω, such that*

$$\begin{aligned} y \in S &\Rightarrow \forall e \exists p (X_{y,e,p} = \omega), \\ y \notin S &\Rightarrow \textit{For almost all } e \textit{ and } p,\ |X_{y,e,p}| < \infty. \end{aligned}$$

Proof. If S is a Π^0_4 set, then there is a u.c.e. sequence $\bar{X}_{y,e,p}$ of initial segments of ω, such that $y \in S \Leftrightarrow \forall e \exists p(\bar{X}_{y,e,p} = \omega)$. Observe that it is easy for us to define a new u.c.e. sequence $Y_{y,e,p}$, such that if it is the case that for some e, $|\bar{X}_{y,e,p}| < \infty$ for all p, then we have $|Y_{y,e',p}| < \infty$ for all p and all $e' \geq e$. On top of doing that, we also need to ensure that each sequence $\{Y_{y,e,p}\}_{p\in\mathbb{N}}$ has got at most one infinite set. To do this, we replace each sequence $\{Y_{y,e,p}\}_{p\in\mathbb{N}}$ by the sequence $\{X_{y,e,\langle p,b\rangle}\}_{p,b\in\mathbb{N}}$, where for each p, the set $X_{y,e,\langle p,b\rangle}$ is allowed to copy $Y_{y,e,p}$ if $b = |Y_{y,e,0} \cup \cdots \cup Y_{y,e,p-1}|$. Clearly for each y and e, $X_{y,e,r}$ is infinite for at most one r. $\square$

We will devote the rest of this section to the proof of:

Theorem 2.4.3. *The set $\{e \in \mathbb{N} : W_e$ is strongly jump traceable$\}$ is Π^0_4-complete.*

2.4.1 Requirements and an overview

We let S be a Π^0_4 set, and let $\{X_{y,e,p}\}$ be the corresponding sequence in Lemma 2.4.2. Fix a y, and we shall build a c.e. set A, such that

$$\begin{aligned} \forall e \exists p(X_{y,e,p} = \omega) \quad &\Rightarrow \quad A \text{ strongly jump traceable,} \\ \forall^\infty e, p(|X_{y,e,p}| < \infty) \quad &\Rightarrow \quad A \text{ is not jump traceable via some order.} \end{aligned}$$

The requirements are:

$$\begin{aligned} \mathcal{R}_{e,p} \quad : \quad & \text{If } |X_{y,e,p}| = \infty \text{ and } h_e \text{ is an order, then make} \\ & A \text{ jump traceable via } h_e. \end{aligned}$$

The requirement $\mathcal{R}_{e,p}$ also has a positive role, namely

$$\begin{aligned} \mathcal{R}_{e,p} \quad : \quad & \text{If } |X_{y,e,p}| < \infty, \text{ make } A \text{ not strongly jump traceable by defeating} \\ & \{T^k_x\}_{x\in\mathbb{N}} \text{ for some } k, \text{ with respect to some order.} \end{aligned}$$

Here, we let $\{h_e\}_{e\in\mathbb{N}}$ be an effective list of all partial computable functions such that $\forall e \forall n > 0(0 < h_e(n) \leq n)$. We shorten notation, and write $X_{e,p}$ for $X_{y,e,p}$, since y is fixed in the construction. We identify sets with their characteristic functions, and initial segments of functions with finite strings. That is, when we write $A\upharpoonright k$, we mean the finite string $A(0)A(1)\cdots A(k-1)$.

Basically, Lemma 2.4.2 helps us to arrange the coding requirements on the construction tree, which is a tree of strategies, in the style of a $\emptyset'''$-priority argument.

This method is originally due to Lachlan [Lac75], and is also presented in Chapter XIV.4 of [Soa87]. The reader is assumed to be familiar with standard tree arguments, a good exposition on this topic can be found in [Soa85]. The true path of the construction is defined as usual, as the leftmost path visited infinitely often during the construction.

We will use the construction in Theorem 2.3.1 as an atomic strategy in this construction. The discussion on how this is to be carried out will be developed over the next few pages. For now, we first give the reader a brief preview of how this is to be done. Basically we want to code a given Π^0_4 set S into the index set of the strongly jump traceable sets. For each y, we perform a separate construction, and uniformly produce a set A at each construction. The trouble is that $y \in S$ or $y \notin S$ is a Π^0_4/Σ^0_4 fact. We arrange for guesses to take place on the construction tree, and with the help of Lemma 2.4.2, the true path of the construction will reflect whether or not $y \in S$. However the construction, having to be effective, can only act on approximations to the true path. Hence there will be some stages where y looks like it is in S; at these stages we try and make A strongly jump traceable. At other stages, y will look like it is out of S. At these stages, we will try and make A not jump traceable via some order function $\tilde{h}$ which we build. However $y \notin S$, being a Σ^0_4 fact, can still cause us to have finitely many objects exhibiting infinitary behaviour. Each of these will force us to make A jump traceable via some order h_e. Therefore, at each stage of the construction where y looks like it is out of S, we have to make A jump traceable via some collection $h_{e_0}, \cdots, h_{e_k}$ of orders, and run the previous construction to make A *not* jump traceable via some $\tilde{h} \ll \min\{h_{e_0}, \cdots, h_{e_k}\}$.

The facts "$X_{y,e,p} = \omega$" and "$X_{y,e,p} < \infty$" are Π^0_2 and Σ^0_2 facts respectively, so each such statement can be measured at a single node on the construction tree, measuring infinitary or finitary behaviour. Lemma 2.4.2 says that the Π^0_4 fact "$y \in S$" can be broken down into a Π^0_2 statement regarding the true outcomes of nodes on the construction tree. That is, $y \in S$ would be equivalent to the fact that "$\forall e \exists p$ such that the node measuring $|X_{y,e,p}|$ has true infinitary outcome". Similarly, if $y \notin S$ then "for almost all e, p, the node measuring $|X_{y,e,p}|$ has true finitary outcome". Hence $y \in S$ or $y \notin S$ will determine the true path of the construction, i.e. which nodes are visited infinitely often and which are not. We will arrange the strategies on the construction tree to align our actions with the true path.

2.4.2 Description of strategies

Each node α on the construction tree has two different strategies. Suppose α is assigned the requirement $\mathcal{R}_{e,p}$, then α will have to test if $|X_{e,p}| = \infty$. Each time some number enters $X_{e,p}$, α will have to direct its efforts towards making A h_e-jump traceable (corresponding to applying negative restraints on A). This is represented by the infinitary outcome. If some time has elapsed with no changes in $|X_{e,p}|$, then α will have to try and make A not strongly jump traceable, by attempting to defeat some trace $\{T_x\}_{x\in\mathbb{N}}$ (corresponding to taking positive action on A). This is represented by the finitary outcome f. Each node α has a dual role - at expansionary stages when $|X_{e,p}|$ increases, α pursues the *negative strategy*, while at non-expansionary stages α pursues the *positive strategy*.

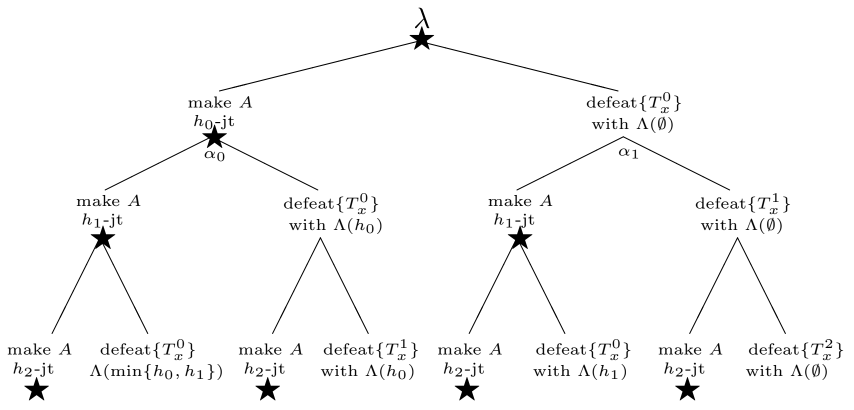
The negative α-strategy (to ensure h_e-jump traceablility) is the usual. It splits its task into infinitely many substrategies $ST_0, ST_1, \cdots$. For each $k \in \mathbb{N}$, the k^{th} substrategy ST_k works by the following: it waits for $h_e(k)\downarrow$, and when $J^A(k)[s]$ next converges, we would enumerate the value into V_k^α (the sequence $\{V_x^\alpha\}_{x\in\mathbb{N}}$ is build at α), and restrain A on the use. At this point in time, we set $size_k^\alpha = h_e(k) - 1$ (to indicate that V_k^α can take at most that many more injuries). When the $size_k^\alpha = 0$, V_k^α is totally filled and any restraint α imposes for it must be permanent. If we arrange for each substrategy ST_k to be assigned to an entire level below α, we immediately meet with a technical obstacle. Recall that in Theorem 2.3.1, the positive requirements had priority (relative to some ST_k), which was determined dynamically. This would not be easy to arrange on a tree of strategies.

Note that we could however, arrange for *all of* the α substrategies to be carried out at α itself. This means that α could impose an ever increasing restraint on the positive strategies below it, even though each substrategy ST_k contributes a finite amount. To get around this problem, we arrange for there to be infinitely many restraint functions $r_0, r_1, \cdots$, where r_k is the restraint function for ST_k, and let different positive strategies be restrained by a different r_k. Suppose each positive strategy below α only wants to enumerate once. We could then let the first positive strategy obey restraint r_0, the second positive strategy obey restraints $\max\{r_0, r_1\}$, and so on. This would be fine if h is the identity order function (otherwise we just adjust accordingly). For details of this, see Theorem 7.3 of [Dow], where this idea was used to give a direct construction of a noncomputable c.e. set which is strongly jump traceable. Carrying out all the α-substrategies at α has the effect of complicating the mechanism at a single node, but simplifies the global considerations and the notations. Each node α on the construction tree builds an entire sequence $\{V_x^\alpha\}_{x\in\mathbb{N}}$ - each α makes a separate attempt to ensure jump traceability at some order.

In Theorem 2.3.1 we had shown that there is an effective procedure Λ, such that given any order function h, the procedure $\Lambda(h)$ outputs a set A and an order function $\tilde{h}$ such that A is h-jump traceable but not $\tilde{h}$-jump traceable. We will attempt to repeat this construction Λ at each node α. At stages when α runs its positive strategy, we would make α diagonalize some trace $\{T_x\}_{x\in\mathbb{N}}$. The same atomic strategy is used for this: we can take control of $J^A(x)$ for some x. Enumerate $J^A(x)[s]\downarrow= s$ with use u. Each time $J^A(x)[s]$ appears in T_x, we put u into A and set $J^A(x)[s']\downarrow$ with another value. If we do this n times (for some chosen n), we would succeed in diagonalizing against $\{T_x\}$ at some order $\tilde{h}$, which we build.

The positive and negative strategies clearly conflict with each other. Why is it not possible to simultaneously run the positive and negative strategies of all requirements? This is equivalent to making A strongly jump traceable, *and* not $\tilde{h}$-jump traceable for some $\tilde{h}$. The trouble is that the e^{th} positive strategy has to obey restraints set up for the sake of making A jump traceable via $\min\{h_0,\cdots,h_e\}$. Even though the threshold values defined in section 2.3.4 are fixed in advance, it is not possible to compute $|h_k^{-1}(r)|$ for every k and r. We really have to make guesses as to whether or not each h_k is an order, and so the $\tilde{h}$ built this way will have to be computable in a $\emptyset''$-oracle. This is explored in [Nga], where the relativization of strong jump traceability is studied.

To make sure $\tilde{h}$ is computable, we will build a different version of $\tilde{h}$ at each of infinitely many nodes on the construction tree. These nodes are called *top nodes.* If τ is a top node, and lives below nodes assigned to requirements $\mathcal{R}_{e_0,p_0},\cdots,\mathcal{R}_{e_j,p_j}$, such that τ believes that $h_{e_0},\cdots,h_{e_j}$ are all orders, then the positive strategy of τ would repeat the construction Λ and make A not jump traceable with respect to the order function $\Lambda(g)$, where $g=\min\{h_{e_0},\cdots,h_{e_j}\}$. The following shows the tree of strategies, and how the strategies may be arranged. In the following diagram, λ represents the root node (the empty string). The left branch represents the infinitary outcome which will be visited during expansionary stages, while the right branch represents finitary outcomes at non-expansionary stages. For instance, at λ-expansionary stages, we will run the negative λ-strategy of making A jump traceable (abbreviated below by jt) via h_0. At non λ-expansionary stages we will run the positive λ-strategy of trying to defeat the trace $\{T_x^0\}_{x\in\mathbb{N}}$, by applying the procedure $\Lambda(\emptyset)$.


λ
make A h_0-jt
α_0
defeat$\{T_x^0\}$ with $\Lambda(\emptyset)$
α_1
make A h_1-jt
defeat$\{T_x^0\}$ with $\Lambda(h_0)$
make A h_1-jt
defeat$\{T_x^1\}$ with $\Lambda(\emptyset)$
make A h_2-jt
defeat$\{T_x^0\}$ $\Lambda(\min\{h_0, h_1\})$
make A h_2-jt
defeat$\{T_x^1\}$ with $\Lambda(h_0)$
make A h_2-jt
defeat$\{T_x^0\}$ with $\Lambda(h_1)$
make A h_2-jt
defeat$\{T_x^2\}$ with $\Lambda(\emptyset)$

The top nodes are marked out in the diagram above using the star symbol. At each of these top nodes, a new application of Λ is started, with a new order function $\tilde{h}$. For instance, λ is a top node where Λ is applied at non-λ-expansionary stages to diagonalize the trace $\{T_x^0\}_{x\in\mathbb{N}}$. The λ-counterexample to strong jump traceability would be the order function $\tilde{h}_\lambda = \Lambda(\emptyset)$, since λ has no stronger priority order functions to respect. The node α_0 gets to act at λ-expansionary stages, and therefore will have to live in harmony with λ's negative strategy. Thus α_0 will start its own version of Λ by building the counterexample $\tilde{h}_{\alpha_0} = \Lambda(h_0)$, at non-$\alpha_0$-expansionary stages. On the other hand, the node α_1 would continue λ's positive strategy by diagonalizing the trace $\{T_x^1\}_{x\in\mathbb{N}}$ at the same order $\tilde{h}_\lambda$, at non-α_1-expansionary stages. α_1 is said to be a *child* of λ. λ's other children are $\alpha^\frown f, \alpha^\frown f^\frown f, \cdots$ (where $^\frown$ is the string concatenation operator), which will all help to carry out the λ-version of procedure Λ, by diagonalizing the traces $\{T_x^2\}, \{T_x^3\}, \cdots$ respectively, at their non-expansionary stages.

If $y \in S$, then there will be infinitely many nodes α on the true path with true infinitary outcome. At these nodes we would succeed in making A jump traceable via arbitrarily slow growing orders, and so A would be strongly jump traceable. On the other hand, if $y \notin S$, then there is a maximal node τ^- on the true path which has true infinitary outcome, and all of its successors $\tau, \tau^\frown f, \tau^\frown f^\frown f, \cdots$ will have true finitary outcome. In this case, τ is the *final top node*. Each of the τ-children will stop running its negative strategy after a finite number of stages have elapsed. The set A would not be jump traceable via the counterexample h_τ.

There is a major modification to the basic construction Λ we have to make to ensure it runs smoothly on a tree of strategies. Suppose τ is a top node running a version of Λ, at the order $\tilde{h}_\tau$. Let $\alpha_1 \subset \alpha_2 \subset \cdots$ be the τ-children nodes, where α_n is devoted to the diagonalization of $\{T^n_x\}$ at non-expansionary stages.

There are two instances where we have to choose a new index x for α_n. The first happens when some $\beta \subset \tau$ increases A-restraint while running its negative strategy - as we have seen in Theorem 2.3.1, this is expected and τ is perfectly prepared for it by choosing the thresholds $C^n_0, \cdots, C^n_{n-1}$ to be sufficiently spaced out. The second case happens due to the fact that requirements are now arranged on a tree: at each α_i-expansionary stage (for some $i \leq n$), we would also have to choose a new index x' for α_n because stronger requirements have now acted and may need to be respected. Even though this happens only finitely often, we do not know how often. Thus it now becomes impossible for us to know how many indices to set aside for α_n.

The solution is to divide the indices into intervals called *regions*. The m^{th} region contains all the indices x such that $\tilde{h}_\tau(x) = m$, based on the calculations in section 2.3.4. Instead of getting α_n to use indices from the n^{th} region all the time, we would make α_n move on to a new region number n', at each α_i-expansionary stage, $i \leq n$. Thus, each time α_n has to choose a new index due to reasons that τ is not prepared for, we would start with the smallest index in a fresh region. This ensures that α_n never runs out of indices.

2.4.3 Construction tree layout

Due to technical reasons, the construction will take place on the full ternary tree $3^{<\omega}$, instead of a binary tree as discussed section 2.4.2. This is because before a node can begin its positive strategy, it would have to first compute all the relevant threshold values, and thus it would have to know which functions are real orders.

Nodes of length $\langle e, p \rangle$ are assigned the requirement $\mathcal{R}_{e,p}$, with three outcomes:

$\infty\infty < \infty f < f$. Outcome f means that we believe $X_{e,p}$ is finite, and we have to start the positive strategy. Outcome $\infty\infty$ means that $X_{e,p}$ is infinite, and h_e is an order. Thus, we have to continue with the negative strategy of making A h_e-jump traceable. Finally, outcome ∞f means that we believe $X_{e,p}$ is infinite, but h_e is not an order. In this case, we have to initialize every node which believes that $X_{e,p}$ is finite, and do nothing else. Let $\alpha <_{left} \beta$ denote that α is strictly to the left of β (i.e. there is some $i < \min\{|\alpha|, |\beta|\}$ such that $\alpha \upharpoonright i = \beta \upharpoonright i$ and $\alpha(i) < \beta(i)$). The construction tree grows downwards. We write "α is a $\mathcal{Q}$-node" to denote the fact that α is assigned the requirement $\mathcal{Q}$.

2.4.4 Notations

If α is a $\mathcal{R}_{e,p}$-node, we let $order(\alpha) = e$. We also let $Z^-(\alpha) := \{\beta \subset \alpha \mid \beta^\frown\infty\infty \subseteq \alpha\}$, which are all the nodes running the negative strategy extended by α. Similarly we let $Z^+(\alpha) := \{\beta \subset \alpha \mid \beta^\frown f \subseteq \alpha\}$, which are the nodes attempting the positive strategy extended by α. For each node α, we define $trace(\alpha)$ by the following: $trace(\lambda) = 0$, and for α of positive length, we let α^- be the predecessor of α, and set

$$trace(\alpha) = \begin{cases} 1 + trace(\alpha^-), & \text{if } \alpha(|\alpha^-|) = f, \\ 0, & \text{otherwise.} \end{cases}$$

$trace(\alpha)$ denotes the number k such that α needs to defeat the k^{th} trace $\{T^k_x\}_{x\in\mathbb{N}}$ for its positive strategy. We say that α is a *top node*, if $trace(\alpha) = 0$. For any node α on the tree, we define $\tau(\alpha)$, the *top of* α to be the maximal $\tau \subseteq \alpha$ such that τ is a top node. We say that α is a *child* of τ if α has top τ. Thus the children of τ are exactly $\tau \subset \tau^\frown f \subset \tau^\frown f^\frown f \subset \cdots$.

Each $\mathcal{R}_{e,p}$-node α has a number of parameters associated with it. The parameters used for its positive strategy are the following:

- If α is a top node, it will build a partial order function $\tilde{h}_\alpha$. We will get a chance to extend $dom(\tilde{h}_\alpha)[s] := \{x \in \mathbb{N} : \tilde{h}_\alpha(x)[s] \downarrow\}$, whenever one of α's children begins its positive strategy.

- $x(\alpha, s)$, which denotes the index of the functional that α is currently enumerating at stage s. It will try and cause $T^{trace(\alpha)}_{x(\alpha,s)}$ to fill up with numbers.

- $u(\alpha, s)$, which is the use of the most recent computation α has enumerated into $J^A_{x(\alpha,s)}$. This is a number pointing at A, which α may decide at a later stage to put into A, when the trace $T^{trace(\alpha)}_{x(\alpha,s)}$ increases in size.
- $region(\alpha, s)$, which denotes the number of elements that α has to try to fill $T^{trace(\alpha)}_{x(\alpha,s)}$ up with.
- $attempt(\alpha, s)$, this is a counter which reflects the progress of α in its positive strategy. This plays the same role as the parameter $p(e, s)$ in Theorem 2.3.1.

The parameters associated with the negative strategy of α are the following:

- At α, we build a uniformly c.e. sequence $\{V^\alpha_k\}_{k\in\mathbb{N}}$. This will trace J^A in the event that $\infty\infty$ is its true outcome.
- We let $size^\alpha_k[s]$ denote the size of the V^α_k-box at stage s, similar to Theorem 2.3.1. It records the number of injuries the V^α_k-box can still take. At the beginning, $size^\alpha_k$ is set to $h_e(k) - 1$, and will be reduced by 1 each time a $J^A(k)$-computation is injured after being traced in V^α_k.

For $e \in \mathbb{N}$, we define the length of convergence for h_e at stage s, to be

$$l(e, s) = \max\{y < s \mid (\forall x \leq y)\ \ (h_{e,s}(x)\downarrow\ \ \wedge\ \ h_e(x) \geq h_e(x-1)) \\ \wedge\ \ h_e(y) > h_e(y-1)\}.$$

We will sometimes write $l(\alpha, s)$ instead of $l(e, s)$. For each $n, s \in \mathbb{N}$, let

$$S(\alpha, n)[s] = \sum_{r=1}^{n} r \cdot |\{k < l(\alpha, s) : size^\alpha_k[s] = r - 1\}|.$$

Furthermore, if τ is a top node, we let

$$S(\tau, n)[s] = \sum_{\beta \in Z^-(\tau)} S(\beta, n)[s].$$

This has the same intended purpose as the parameter $S(n)$ of Theorem 2.3.1, with two marked differences. First, each τ has to now consider the total effect of all $\{V^\beta_k\}$ for every $\beta \subset \tau$ which it believes will make A jump traceable via $h_{order(\beta)}$ (unlike in Theorem 2.3.1, where we only had a single order to consider). Secondly, because the strategy of τ is based on the fact that its guesses are correct, so τ has to wait for various computations to converge before it can proceed further. Therefore, threshold values will have to be computed during the construction itself, and their values will depend on the current situation. In particular, the cardinality in the sum of $S(\alpha, n)[s]$ is computed using *current* $(r-1)$-*boxes*, instead of using original $(r-1)$-boxes as in Theorem 2.3.1.

Suppose τ is a top node. The parameters $\{C^\tau_{n,k} \mid n > 0 \ \wedge \ k < n\}$ and $\{I^\tau_n \mid n > 0\}$ helps us keep track of the threshold values, and as mentioned above, will be computed during the construction. These parameters are all set to $\uparrow$ initially. The values $C^\tau_{n,1}, \cdots, C^\tau_{n,n-1}$, and I^τ_n are associated with the n^{th} region, similar to their counterparts in Theorem 2.3.1.

There are infinitely many indices $\nu_0 < \nu_1 < \cdots$ set aside for use by τ-children. If α is a child of τ, then $x(\alpha)$ will be chosen from this list. $\tilde{h}_\tau$ will be set to a constant value over each interval $\{x \in \mathbb{N} \mid \nu_{i-1} < x \leq \nu_i\}$, for $i \in \mathbb{N}$. Thus, whenever we refer to $x(\alpha)$ or $\tilde{h}_\tau$, we mean the values modulo intervals partitioned by the ν_i's. That is, $\tilde{h}_\tau(i)$ will refer to the (common) value of $\tilde{h}_\tau(x)$ for $\nu_{i-1} < x \leq \nu_i$, and we write $x(\alpha) = i$ instead of $x(\alpha) = \nu_i$. We let Ψ^A_α denote the functional α is enumerating at stage s, with index $x(\alpha, s)$. The value of $x(\alpha, s)$ will change from time to time, specifically at those stages when α is initialized or reset (the meaning of these two terms will be explained soon). At these stages, α will start enumeration of a new functional, with a new index. α will pick the new index according to the following:

- If it is the case that α is reset, we increment $x(\alpha)$ by 1.
- If α is initialized, it will be asked to start on a fresh region n. In this case, we will set $x(\alpha) = n$.

For a node α and stage s, we define *threshold*(α, s) by

$$threshold(\alpha, s) = C^{\tau(\alpha)}_{r,r-2-a},$$

where $r = region(\alpha, s)$ and $a = attempt(\alpha, s)$. This refers to the current threshold value that α has to obey. When we *initialize* α at stage s, we do the following.

- Pick a fresh number n for *region*(α).
- Set *attempt*$(\alpha) = 0$.

- Set $x(\alpha) = n$.
- Set $u(\alpha) = \uparrow$.
- Set $C^{\tau(\alpha)}_{n,-1} = n$, and $C^{\tau(\alpha)}_{n,0} = n + 2$.

That is, α has to restart its negative strategy due to reasons that $\tau(\alpha)$ had not foreseen. If α is injured because of activity above $\tau(\alpha)$, we will *reset* α (at stage s) by doing the following. If $attempt(\alpha, s) = region(\alpha, s)$ (α's negative strategy has succeeded) or $x(\alpha) = region(\alpha, s) + I^{\tau(\alpha)}_{region(\alpha,s)}$ (i.e. α has run out of indices), do nothing. Otherwise increase $x(\alpha)$ by 1, set $u(\alpha) = \uparrow$, and set $attempt(\alpha) = 0$.

Definition 2.4.4. We say that a node α *requires positive attention* at stage s, if $attempt(\alpha, s) < region(\alpha, s)$, and one of the following (ATT0)-(ATT3) holds.

(ATT0) One of $C^{\tau(\alpha)}_{n,1}, \cdots, C^{\tau(\alpha)}_{n,n-1}$, or $I^{\tau(\alpha)}_n$ has not yet received a value, where $n = region(\alpha, s)$.

(ATT1) There is no computation in Ψ^A_α which currently applies.

(ATT2) All of the following hold:

- there is a computation in Ψ^A_α which currently applies with use $u(\alpha, s)$,
- there is some β and k such that $\beta \in Z^-(\alpha)$, and $k < l(\beta, s)$, and we have $J^A(k)[s] \downarrow$ with use larger than $u(\alpha, s)$,
- $size^\beta_k[s] \downarrow \leq threshold(\alpha, s)$.

(ATT3) There is a computation in Ψ^A_α which currently applies with use $u(\alpha, s)$ and value $r = \Psi^A_\alpha(x(\alpha, s))[s]$, such that r has shown up in the trace $T^{trace(\alpha)}_{x(\alpha,s)}$.

If (ATT0) holds, then α is not yet ready to start its positive strategy; wait until all the relevant parameters have been defined. If (ATT1) holds, we need to place an axiom into Ψ^A_α. If (ATT2) holds, the restraint on α from β above has increased beyond $u(\alpha, s)$ - there is some high priority box blocking the positive strategy of α. We have to reset α. If (ATT3) holds, we will need to take positive action to defeat the $trace(\alpha)^{th}$ trace.

2.4.5 Construction of A

At each stage s of the construction, we will define the approximation to the true path of the construction, δ_s of length $< s$. We say that α *is visited* at stage s, if $\delta_s \supset \alpha$. We will also state the actions to be taken by the nodes on δ_s. At stage $s = 0$, set $\delta_s = \lambda$ and do nothing else.

Let $s > 0$, and assume that $\alpha = \delta_s \restriction d$ has been defined for $d < s$. Suppose α is an $\mathcal{R}_{e,p}$-node. We now have to determine which of the three outcomes to take. Check if $|X_{e,p}|$ has increased since the last visit to α. If not, let $\delta_s(d) = f$. Otherwise, we let $t < s$ be the stage number of the most recent visit to α in which $\delta_t(d) \neq f$. If

- $l(\alpha, t) < l(\alpha, s)$, and
- $h_e(l(\alpha, s)) > C^{\tau}_{n,r}$ for every $n, r \in \mathbb{N}$, and every top node $\tau \supseteq \alpha^\frown\infty\infty$,

we will let $\delta_s(d) = \infty\infty$. Otherwise let $\delta_s(d) = \infty f$. Let s^- be the stage number of the previous visit to α. If

- s^- does not exist (i.e. this is the first visit to α), or
- $\delta_t <_{left} \alpha$ for some $s^- < t < s$, or
- some $\sigma \in Z^+(\alpha)$ enumerates in A between the two visits to α at s^- and s,

we will initialize α and set $V^\alpha_k = \emptyset$ for all k, and set $size^\alpha_k = h_e(k) - 1$ for all $k < l(\alpha, s)$. Next, we will take the corresponding actions depending on the outcome $\delta_s(d)$ of α determined above:

1. $\delta_s(d) = \infty\infty$: we run the negative strategy for α. For all $k < l(\alpha, s)$ such that $J^A(k)[s]\downarrow$, we enumerate the value $J^A(k)[s]$ into V^α_k. For all $k < l(\alpha, s)$ such that $size^\alpha_k$ has not yet been assigned a value, we do the update $size^\alpha_k = h_e(k) - 1$.

2. $\delta_s(d) = \infty f$: do nothing.

3. $\delta_s(d) = f$: we run the positive strategy for α. If $\delta_{s^-}(d) \neq f$, we initialize α. Next, if α does not require positive attention, we do nothing. Otherwise, take the appropriate action listed below, and declare that α has *received positive attention* at stage s.

(a) *(ATT0) holds*: if α has just been initialized at this current stage, terminate the definition of δ_s, do nothing else and go to stage $s+1$. Otherwise, do the following:

- Pick the smallest $r < n := \mathit{region}(\alpha, s)$ such that $C^{\tau(\alpha)}_{n,r} \uparrow$, and set $C^{\tau(\alpha)}_{n,r} = C^{\tau(\alpha)}_{n,r-1} + 2 + n + nS(\tau(\alpha), C^{\tau(\alpha)}_{n,r-1})[s]$.
- If $C^{\tau(\alpha)}_{n,r} \downarrow$ for all $r < n$, set $I^{\tau(\alpha)}_n = S(\tau(\alpha), C^{\tau(\alpha)}_{n,n-1})[s]$. Additionally, we set $\tilde{h}_{\tau(\alpha)}(x) = n$ for all $x \leq n + I^{\tau(\alpha)}_n$ such that $x \notin \mathit{dom}(\tilde{h}_{\tau(\alpha)})[s]$.
- We terminate the definition of δ_s, and go to stage $s+1$.

(b) $\neg$*(ATT0)*$\wedge$*(ATT1)*: we enumerate a computation $\Psi^A_\alpha(x(\alpha, s))[s] \downarrow = s$ with fresh use $u(\alpha, s)$.

(c) $\neg$*(ATT0)*$\wedge\neg$*(ATT1)*$\wedge$*(ATT2)*: reset α.

(d) $\neg$*(ATT0)*$\wedge\neg$*(ATT1)*$\wedge\neg$*(ATT2)*$\wedge$*(ATT3)*: for each k such that there is some $\beta \in Z^-(\alpha)$ and $k < l(\beta, s)$, such that $J^A(k)[s] \downarrow$ with use $j(k, s) > u := u(\alpha, s)$, we decrease $\mathit{size}^\beta_k[s]$ by 1. Enumerate u into A, and increase $\mathit{attempt}(\alpha, s)$ by 1.

This ends the construction. The purpose of terminating the definition of δ_s under step 3(a), is because we do not want the other $\tau(\alpha)$-children to pick their threshold values before α finishes with its own. The true path of the construction is defined as the leftmost path visited infinitely often during the construction.

2.4.6 Verification

The following fact is rather obvious, but important: if a node σ is initialized and $\mathit{region}(\sigma)$ set to n, then σ can make at most $1 + (n-1)(I^{\tau(\sigma)}_n + 1)$ many enumerations into A before it is next initialized.

Since δ_s can sometimes have length $< s$, we have to see that the true path of the construction exists, and is infinitely long: suppose α is a node visited infinitely often, and $\delta_t <_{left} \alpha$ for finitely many t. We need to see that only finitely often, α is visited and the definition of δ_s is terminated at α. This is an issue only when α stops playing the outcomes $\infty\infty$ and ∞f after a finite number of stages. Clearly each $\sigma \in Z^+(\alpha)$ enumerates into A only finitely often, by the above fact. Hence

α is initialized only finitely often, and so step 3(a) of the construction applies only finitely often.

Thus, for each node α on the true path, we can let $True(\alpha)$ be the least stage t, such that

- α is visited at stage t,
- for all $u > t \Rightarrow \delta_u \not<_{left} \alpha$,
- no $\sigma \in Z^+(\alpha)$ enumerates into A after stage t.

Clearly if $\alpha \subset \beta$ are both on the true path, then $True(\alpha) \leq True(\beta)$.

Lemma 2.4.5. *Let α be a $\mathcal{R}_{e,p}$-node on the true path.*

(i) If $|X_{e,p}| < \infty$, then f is the true outcome of α.

(ii) If $|X_{e,p}| = \infty$, but h_e is not an order, then ∞f is the true outcome of α.

(iii) If $|X_{e,p}| = \infty$, and h_e is an order, then $\infty\infty$ will be the true outcome of α.

Proof. Clearly (i) holds because if $X_{e,p}$ is finite, then eventually no new numbers show up and α will always play outcome f at each visit past a certain stage. For (ii) and (iii), observe that h_e is an order iff $l(\alpha, s) \to \infty$. (ii) is clear enough, so we prove (iii). Suppose that $|X_{e,p}| = \infty$, and h_e is an order, but α never plays the outcome $\infty\infty$ after stage s.

Let $s_0 > s$ be the least stage such that $h_e(l(\alpha, s_0)) > C^{\tau}_{n,r}$ for every top $\tau \supseteq \alpha^\frown\infty\infty$ and $n, r \in \mathbb{N}$. The number s_0 exists because no top $\tau \supseteq \alpha^\frown\infty\infty$ is ever visited after stage s. Let $s_1 > s_0$ be the least stage such that $l(\alpha, s_1) > l(\alpha, s_0)$. We also let $s_2 > s_1$ be the least such that $X_{e,p}[s_2] - X_{e,p}[s_1] \neq \emptyset$. Finally let $t \geq s_2$ be the least stage such that α is visited at stage t.

If α is visited at any stage u, $s_1 \leq u < s_2$, then α must play outcome f. To see this, suppose not, and let u^- be the previous visit to α prior to stage u. Note that $u^- \not< s_1$, otherwise $l(\alpha, u^-) < l(\alpha, s_1) \leq l(\alpha, u)$. But we also cannot have $s_1 \leq u^- < u$, otherwise $X_{e,p}[u] \supsetneq X_{e,p}[u^-] \supseteq X_{e,p}[s_1]$.

Since α does not play outcome f at any stage u, $s_1 \leq u < s_2$, it follows that $\delta_t \supset \alpha^\frown\infty\infty$, a contradiction. □

The following is the analogue of Lemma 2.3.3, applied to the current situation. In part (i) we show that the maximum number of different restraints that a current b-box can hold is at most $b+1$. Part (ii) says that the maximum number of different restraints held by any current b-box for $b < n$ of stronger priority, is at most $S(\tau, n)[s]$.

Lemma 2.4.6. *(i) Let β be on the true path, with true outcome $\infty\infty$. Let $k \in \mathbb{N}$, and t_0 be a stage $\geq True(\beta)$ such that $size_k^\beta[t_0] \downarrow$. Then,*

$$|\{A_s \restriction j(k,s) : J^A(k)[s] \downarrow \ \wedge \ \delta_s \supset \beta^\frown\infty\infty \ \wedge \ s \geq t_0\}| \leq 1 + size_k^\beta[t_0].$$

(ii) Let τ be a top node on the true path, $n \in \mathbb{N}$, and $t_1 \geq True(\tau)$. Then, the number of pairs $\langle A_s \restriction j(k,s), k\rangle$ for which

(a) $J^A(k)[s] \downarrow$,

(b) For some $\beta \in Z^-(\tau)$, we have $size_k^\beta[t_1] < n$,

(c) $\delta_s \supset \tau$ and $s \geq t_1$,

is at most $S(\tau, n)[t_1]$.

Proof. (i): The proof is the same as in Lemma 2.3.3(i); we include it here for the sake of completeness. Suppose that there are stages $t_0 \leq s_0 < s_1 < \cdots < s_m$ such that $A_{s_i} \restriction j(k, s_i) \neq A_{s_{i+1}} \restriction j(k, s_{i+1})$ for all $i = 0, \cdots, m-1$ (where $m = 1 + size_k^\beta[t_0]$). For each i, the change $A_{s_i} \restriction j(k, s_i) \neq A_{s_{i+1}} \restriction j(k, s_{i+1})$ must have been caused by some $\sigma \supseteq \beta^\frown\infty\infty$ receiving positive attention at some stage t where $s_i \leq t < s_{i+1}$. It must also be the case that we had decreased $size_k^\beta$ at stage t. This means that by the time we reach stage s_{m-1}, we have $size_k^\beta = 0$. Hence at all stages $t \geq s_{m-1}$, no $\sigma \supseteq \beta^\frown\infty\infty$ can make an enumeration below $j(k, s_{m-1})$, a contradiction.

(ii): Using part (i). □

The following is the counterpart to Lemma 2.3.4. We show that if $C_{n,r}^\tau[s_0]$ has been defined, then no box of a current size at least $C_{n,r}^\tau[s_0]$ can be promoted to a $C_{n,r-1}^\tau[s_0]$-box.

Lemma 2.4.7. *Let α be on the true path with true outcome f, $\tau = \tau(\alpha)$, and $\lim_{t\to\infty} region(\alpha, t) = n$. Then, the following are true.*

(i) For each $0 \leq r \leq n-1$, $\beta \in Z^-(\tau)$, each stage s_0 such that $C^\tau_{n,r}[s_0]\downarrow$, and k such that $size^\beta_k[s_0] \geq C^\tau_{n,r}$, we have

$$\forall t(t \geq s_0 \Rightarrow size^\beta_k[t] > C^\tau_{n,r-1}).$$

(ii) The total number of times which α can be reset after its last initialization, is bounded by I^τ_n.

Proof. (i): We proceed by induction on r. Suppose the results hold for all $r' < r$. Suppose on the contrary that there is some $\beta \in Z^-(\tau)$, stage s_0 and number k such that $size^\beta_k[s_0] \downarrow \geq C^\tau_{n,r}$, and some $s > s_0$ such that $size^\beta_k[s] \leq C^\tau_{n,r-1}$. We may as well assume that $size^\beta_k[s_0] = C^\tau_{n,r}$. Since $C^\tau_{n,r}[s_0]\downarrow$, it follows there is some stage $\bar{s_0} \leq s_0$, such that α is visited at stage $\bar{s_0}$ in which $C^\tau_{n,r}$ receives its definition.

Suppose $t \geq s_0$ is a stage where some σ enumerates into A, resulting in a decrease in $size^\beta_k$. It is clear that

(i) $\sigma <_{left} \alpha$,

(ii) $\sigma \supseteq \alpha^\frown\infty\infty$ or $\sigma \supseteq \alpha^\frown\infty f$,

(iii) $\sigma \in Z^+(\alpha)$,

cannot hold, lest α gets initialized after stage s_0. It also cannot be the case that

(iv) $\sigma^\frown\infty\infty \subseteq \alpha$ or $\sigma^\frown\infty f \subseteq \alpha$,

(v) $\sigma >_{left} \alpha$,

(vi) $\sigma \supseteq \alpha^\frown f$,

because otherwise $region(\sigma, t)$ has to be larger than $C^\tau_{n,r}$ when σ enumerates at stage t. This is because α is visited at stage $\bar{s_0} \leq s_0$, and before $C^\tau_{n,1}, \cdots, C^\tau_{n,r}$ receives their definition, we never get to visit any node $\delta \supseteq \alpha^\frown f$. This means that $size^\beta_k[t] \leq C^\tau_{n,r} < region(\sigma, t) \leq threshold(\sigma, t)$, a contradiction.

This leaves only the case when $\sigma = \alpha$; that is, the only node that can bring down $size^\beta_k$ after stage s_0, is α itself. We want to count the number of possible such stages t. At stage t, we must have $C^\tau_{n,r} \geq size^\beta_k[t] > threshold(\alpha, t) = C^\tau_{n,n-2-attempt(\alpha,t)}$, which puts $attempt(\alpha, t) > n-2-r$. As before, we split the counting into the cases:

- *Case x: t is a stage where α makes an enumeration which results in a decrease in $size_k^\beta$, and $attempt(\alpha, t) = x$, where $n-1-r \leq x \leq n-1$.*

  If $x = n-1$, then $attempt(\alpha)$ will be increased to n and α never enumerates again. So, suppose that $x < n-1$ (and hence $r > 0$). In order for Case x to apply again, α has to be reset at some (least) stage $t' > t$, where $attempt(\alpha, t') \geq x+1 > n-1-r$. This means that for some β' and k' with $\beta' \in Z^-(\tau)$, we have $J^A(k')[t'] \downarrow$ and $size_{k'}^{\beta'}[t'] \leq threshold(\alpha, t') \leq C_{n,r-2}^\tau$.

  We firstly claim that $size_{k'}^{\beta'}[\bar{s_0}] \downarrow$. Suppose not. Since $r > 0$, it follows that $h_{order(\beta')}(l(\beta', \bar{s_0})) > C_{n,r-1}^\tau$, and that $h_{order(\beta')}(k') > C_{n,r-1}^\tau$. Applying induction hypothesis (on $r-1$) gives us a contradiction.

  We can further conclude that $size_{k'}^{\beta'}[\bar{s_0}] < C_{n,r-1}^\tau$ (by applying the induction hypothesis on $r-1$), and by Lemma 2.4.6(ii), there can be at most $1 + S(\tau, C_{n,r-1}^\tau)[\bar{s_0}]$ many stages t where Case x applies (by associating each t with the string $A_{t'} \upharpoonright j(k', t')$).

Considering all the different cases, we see that the smallest value $size_k^\beta$ can take, is $C_{n,r}^\tau - 1 > C_{n,r-1}^\tau$, if $r = 0$. On the other hand if $r > 0$, the smallest value $size_k^\beta$ can take, is $C_{n,r}^\tau - 1 - n(1 + S(\tau, C_{n,r-1}^\tau)[\bar{s_0}]) = C_{n,r-1}^\tau + 1 > C_{n,r-1}^\tau$.

(ii): Let t_0 be the stage where I_n^τ receives its definition. If α is reset at stage $t > t_0$, then for some β and k with $\beta \in Z^-(\tau)$, we have $J^A(k)[t] \downarrow$ and $size_k^\beta[t] \leq threshold(\alpha, t) \leq C_{n,n-2}^\tau$. By part (i), it follows that $size_k^\beta[t_0] \downarrow$, and furthermore that $size_k^\beta[t_0] < C_{n,n-1}^\tau$. Again by Lemma 2.4.6(ii), there can be at most $S(\tau, C_{n,n-1}^\tau)[t_0] = I_n^\tau$ many such stages t (by associating each t with the string $A_t \upharpoonright j(k, t)$). □

Lemma 2.4.8. *$y \in S \Leftrightarrow A$ is strongly jump traceable.*

Proof. ($\Rightarrow$): Suppose $y \in S$, and we let $h = h_e$ be an order function. There is some p such that $|X_{e,p}| = \infty$, so let α be the node on the true path assigned the requirement $\mathcal{R}_{e,p}$. By Lemma 2.4.5, α has true outcome $\infty\infty$. Fix a $k \in \mathbb{N}$. If $J^A(k) \downarrow$, then clearly it will be enumerated into V_k^α after stage $True(\alpha)$. Each distinct value in V_k^α corresponds to a string $A_s \upharpoonright j(k, s)$, where $J^A(k)[s] \downarrow$, $\delta_s \supset \alpha^\frown\infty\infty$ and $s \geq True(\alpha)$ and $l(\alpha, s) > k$, so by Lemma 2.4.6(i) it follows that $|V_k^\alpha| \leq h_e(k)$.

($\Leftarrow$): Suppose now $y \notin S$. By Lemma 2.4.5, let τ be the maximal top node on the true path. $\tilde{h}_\tau$ is total because all of τ's children α are on the true path, and each

α will extend $dom(\tilde{h}_\tau)$ at least once. Furthermore, each time α defines a piece of $\tilde{h}_\tau$, it picks a value larger than anything used before, so that $\tilde{h}_\tau$ is an order.

Let $\alpha \supseteq \tau$ be on the true path, and let $n = \lim_{t\to\infty} region(\alpha, t)$. By Lemma 2.4.7(ii), α never runs out of indices after its last initialization. We let $x = \lim_{t\to\infty} x(\alpha, t)$, and clearly we have $\tilde{h}_\tau(x) = n$. Suppose that $J^A(x) \downarrow\in T_x^{trace(\alpha)}$. Let $s_0 > True(\alpha)$ be large enough so that $J^A(x) \in T_x^{trace(\alpha)}[s_0]$, and $x(\alpha, s_0) = x$. It must be the case that $\Psi_\alpha^A(x)[s_0] \downarrow= J^A(x)$ (because any axiom $\langle x, y, \sigma\rangle$ we enumerate into Ψ_α^A after stage s_0, must have $y \geq s_0 > J^A(x)$). Hence it must be the case that $attempt(\alpha, s_0) = n$, otherwise (ATT3) would hold and we would destroy the correct axiom in Ψ_α^A. This means that there are n many different stages t (before stage s_0) where α receives positive attention, and $attempt(\alpha, t)$ is increased by 1. After each such stage t we also enumerate a new value for $\Psi_\alpha^A(x)$, and wait for it to be traced. By stage s_0, we have $|T_x^{trace(\alpha)}[s_0]| \geq n$. $\square$

The proof of Theorem 2.4.3 is complete, upon observation that the construction of A is uniform in y, and by using a uniform version of the Recursion Theorem.

Chapter 3

Beyond strong jump traceability

The work in this chapter has been submitted for publication.

3.1 Introduction

The last few years have seen an extraordinarily fruitful interaction between the areas of algorithmic information theory and computability theory. A highlight in the programme of trying to understand algorithmic information has been the work on various notions of lowness. A notion of lowness is one indicating weakness as an oracle. Intuitively, a notion of lowness means that a set does not give any extra power to certain operations when it is used as an oracle; its role as an oracle can be dispensed with. The classical example is where the operation is the Turing jump, where a set A is low if $A' \equiv_T \emptyset'$. The notion of lowness can be considered for many other concepts where relativization makes sense.

The work in this chapter was motivated by the beautiful interactions of lowness and simplicity in Kolmogorov complexity with concepts from classical computability. The work of Terwijn and Zambella [TZ01], Kjos-Hanssen, Nies and Stephan [KHNS05], Bedregal and Nies [BN03] showed that the notion of being low for Schnorr randomness coincided with a natural combinatorial notion: being computably traceable. Perhaps the best known work on the K-trivials is the work of Nies showing the coincidence of several simple classes:

Theorem 3.1.1 (Nies [Nie97, Nie05b]). *A is K-trivial iff A is low for Martin-Löf randomness iff A is low for K.*

A real A is *low for K* if $\exists c \forall \sigma (K(\sigma) \leq K^A(\sigma) + c)$; that is, A does not help in the compression of strings when used as an oracle. The robustness of this class is further demonstrated when various other characterizations were found; for instance the reals low for weak 2-randomness, and the bases of Martin-Löf randomness [HNS07]. Here A is a base for (Martin-Löf) randomness if $A \leq_T Z$ for some Z which is random relative to A. Intuitively there cannot be many possibilities for initial segments of a base for randomness, because we can use the given Turing reduction Φ (where $A = \Phi^Z$) to lower the Kolmogorov complexity of possible initial segments of Z. This is in fact the driving force behind the "hungry sets" theorem of [HNS07]. We refer the reader to Franklin and Stephan [FS08] for a Schnorr random version of a base. Other notions of bases which have been studied are the LR-base for randomness [BLN, BLS08b], and the JT-base for randomness which we will cover in Section 6.3. These are notions obtained from a base for randomness, by replacing Turing reducibility with different weak reducibilities.

These characterizations show that the notion of ML-lowness is a very robust one. It is very natural to ask if the class relates to notions from classical computability in any meaningful way. For example, is there a computability-theoretical characterization like the one for EX-lowness? Is there a combinatorial characterization in terms of classical computability like the one for Schnorr lowness?

The results linking K-triviality to superlowness and jump traceability show that K-triviality is essentially an enumerable phenomenon. This connects K-triviality (as a notion of lowness in terms of randomness) with the traditional notion of lowness (in terms of Turing jump operators). These results go some way towards understanding ML-lowness in classical terms. Nies' proof that K-trivial sets are superlow actually showed that their *jump* has a tracing property similar to computable traceability. In particular, every K-trivial real is jump traceable with an order function of growth rate $\sim h(n) = n \log n$. Nies' results suggested that perhaps K-triviality is related to the growth rate of orders for jump tracing. Armed with this insight, Figueira, Nies and Stephan introduced and studied strong jump traceability. They asked if this was perhaps the coveted characterization of K-triviality in classical and also combinatorial terms. This question was also raised in Miller and Nies [MN06].

Subsequent research by Cholak, Downey and Greenberg showed that strong jump traceability is too strong a combinatorial notion to characterize K-triviality. Many questions thus suggest themselves. How then does the class of strongly jump traceable sets relate to other notions from Kolmogorov complexity such as ML-cuppability and the like. Is this the limit of the cost function construction? Is there any natural proper subclass of this ideal? What else can be said about the class of strongly jump traceable reals, as they seem a very interesting class in their own right. Even though neither notion (of jump traceability and strong jump traceability) gives us an exact characterization of the K-trivials, the associated results provide a good idea of the upper and lower bounds on the order of jump traceability which would capture K-triviality. By analyzing the proofs which give the lower bound $\sim \sqrt{\log n}$ and upper bound $\sim n \log^2 n$, two possible characterizations have been suggested. Greenberg suggested that A is K-trivial iff A is jump traceable for all orders h with $\sum_{n\in\omega} \frac{1}{h(n)} < \infty$. This has been refuted by Barmpalias, Downey and Greenberg [BDG]. In Theorem 3.3.3 we (independently) extend the result of Barmpalias, Downey and Greenberg by showing that there is a pair of superlow c.e. sets whose join has complete Turing degree. The second conjecture is that the collection of orders should be the class of all orders h satisfying $\sum_{n\in\omega} 2^{-h(n)} < \infty$, and this is still open.

This chapter and the previous one are devoted to the constellation of questions above. In Section 3.3 we show that not every c.e. set jump traceable at identity is K-trivial, disproving one of the conjectures towards a combinatorial characterization of ML-lowness. This result shows that any hope of giving a combinatorial characterization of K-triviality in terms of jump traceability, must include at least some functions which grow more slowly than the identity.

3.2 Subclasses of the K-trivials

The above describes progress in an ongoing programme to characterize a notion involving Kolmogorov complexity in terms of a combinatorial definition (free of measure theoretic facts). We also mention that there have been results which go the other way. Namely there are results which describe the strongly jump traceable sets by simplicity in terms of randomness. The first result of this kind is the following

Theorem 3.2.1 ([FNS06]). *A is strongly jump traceable iff for each computable order h, and for every x, $C(x) \leq^+ C^A(x) + h(C^A(x))$.*

If we replace the condition above by "low for C", then all such sets are C-trivial and hence computable by a theorem of Chaitin [Cha76]. However if the condition is weakened slightly (h can be very slow growing), then we would be able to characterize the strongly jump traceable sets.

The standard way of constructing a K-trivial set (and indeed, a noncomputable member for every known subclass of the K-trivials) is by a cost-function construction. For instance, an easy way of building a promptly simple K-trivial set A, is to specify an enumeration A_s for A, so that the e^{th} promptly simple requirement enumerates x into A at stage s only if $c(x,s) < 2^{-e}$. Here $c(x,s) = \sum_{x<y\leq s} 2^{-K(y)[s]}$ represents the cost of enumerating x at s; as long as we change $A_s(x)$ whenever the associated cost is small, we will be able to specify a machine (using the approximation A_s) which witnesses the K-triviality of A. By formalizing the notion of a cost function, Greenberg and Nies [GN] were able to characterize the strongly jump traceable c.e. sets as those with slow enumerations obeying every benign cost function. The reader should think of a benign cost function as one in which the number of times the cost function increases by 2^{-e} is bounded by a computable function $g(e)$. For instance, the standard cost function above for constructing a K-trivial is benign.

Another central theme in expressing strong jump traceability in terms of algorithmic randomness is the idea of a "diamond class".

Definition 3.2.2. If $\mathcal{C} \subseteq 2^{\mathbb{N}}$, then let $\mathcal{C}^\diamond = \{A \mid A$ is c.e. and $\forall Y \in \mathcal{C}$ and Y random, we have $A \leq_T Y\}$.

The idea is that if $A \in \mathcal{C}^\diamond$ for some collection of sets exhibiting highness properties, then A should be computationally weak in the sense that many sets compute it. Hirschfeldt and Miller showed that if $\mathcal{C}$ is any Σ^0_3 null class, then $\mathcal{C}^\diamond$ contains a promptly simple c.e set. Since every c.e. set which is bounded by an incomplete random has to be K-trivial, $\mathcal{C}^\diamond$ will be a subclass of the K-trivial sets whenever $\mathcal{C}$ contains an incomplete random. Greenberg, Hirschfeldt and Nies [GHN] showed that $\{Z \mid Z$ is superlow$\}^\diamond$ and $\{Z \mid Z$ is ω-c.e.$\}^\diamond$ were exactly the c.e. strongly jump traceable sets, and Nies [Nie] characterized this class by $\{Z \mid \emptyset'' \geq_{tt} Z'\}^\diamond$.

In this chapter we introduce a proper subclass of the strongly jump traceable reals. This class has a number of very interesting properties and is the first class which cannot, for instance, be promptly simple, nor can it be carried out below an arbitrary c.e. degree. We believe its study may have significant implications in classical computability.

We will study a variant of strong jump traceability by relativizing (partially) the concept of traceability. In Section 3.4, we define what it means for a c.e. set A to be strongly jump traceable by another set X, denoted by the relation "$A \in SJT(X)$". We study this binary relation, keeping in mind the interaction with computational lowness. The main result in Section 3.4 is Theorem 3.4.5, where we construct two c.e. sets A and B which are both strongly jump traceable, but $A \notin SJT(B)$. This is somewhat contrary to intuition, since if B is computationally very weak, we would expect that $SJT(B)$ is exactly the class of strongly jump traceable c.e. sets.

Theorem 3.4.5 provides us with the encouragement that we need to take this study further. We define a new class of c.e. sets $\mathcal{H} := \bigcap\{SJT(W) \mid W \text{ is c.e.}\}$, and call these sets *hyper jump traceable*. These sets are all strongly jump traceable, with an even more marked resemblance to the computable sets. In Section 3.6 we construct a noncomputable hyper jump traceable c.e. set, by using a degenerate $\emptyset'''$-priority argument. We also show that such sets can be cuppable. Thus, there is a fundamental property which separates them from the computable sets. In Section 3.7 we show that no hyper jump traceable c.e. set can be low cuppable, and this implies that the hyper jump traceable c.e. sets cannot be constructed using a single cost function. This forms the first known example of a subclass of the K-trivials free of promptly simple sets. Moreover, they form the first class of sets which are constructed without any cupping notions in mind, but which *can* be cuppable, yet *cannot* be promptly simple.

In Section 3.8 we show that no c.e. set A is strongly jump traceable by *all* Δ^0_2 sets, apart from the computable sets. Therefore in some sense, such a variation of strong jump traceability is the strongest possible for the c.e. sets. In Section 3.9 we show that not only is every hyper jump traceable c.e. set cappable, but there is also a single capping companion for the entire class $\mathcal{H}$. Hence, $\mathcal{H}$ is far away from being promptly simple.

3.3 Not every c.e. set jump traceable at identity is K-trivial

It is known that the K-trivial reals form an extremely robust class, having different characterizations. Existing results tell us that in the c.e. case, we have A is jump traceable at order $\sqrt{\log n} \Rightarrow A$ is K-trivial $\Rightarrow A$ is jump traceable at order $n \log n$. The following two conjectures have been suggested.

Conjecture 3.3.1. *A is K-trivial if and only if A is jump traceable at all order functions h such that $\sum_{n=1}^{\infty} h(n)^{-1} < \infty$.*

Conjecture 3.3.2. *A is K-trivial if and only if A is jump traceable at all order functions h such that $\sum_{n=1}^{\infty} 2^{-h(n)} < \infty$.*

We show that not every c.e. set jump traceable at identity is K-trivial, hence giving a negative answer to Conjecture 3.3.1. We do this by constructing a pair of c.e. sets which are both jump traceable at identity, and whose join has complete Turing degree. This result says that any class of order functions characterizing the K-trivials must contain a member which doesn't dominate the identity function. Our result is consistent with Conjecture 3.3.2, and that is still open.

Bickford and Mills [BM] introduced the concept of superlowness, although they called such sets "abject". The term "superlow" was coined by Mohrherr and she also studied this class in her thesis. Bickford and Mills [BM] constructed a pair of c.e. superlow sets which joins to $\emptyset'$. A calculation of the underlying order function in their proof yields a growth rate of around n^2; in the following theorem we will improve this to the identity function. Since the K-trivial sets are closed under $\oplus$, one of the two sets constructed in Theorem 3.3.3 cannot be K-trivial. The exact statement of the theorem is:

Theorem 3.3.3. *There are c.e. sets A_0 and A_1 which are both jump traceable at identity, such that $\emptyset' \leq_T A_0 \oplus A_1$.*

Since when tracing partial functions, the order makes a difference, we have to be more precise when looking at the exact order which a set is jump traceable at.

Definition 3.3.4. We say that A *is jump traceable at* h, or that A is h-jump traceable, if there is a c.e. trace $\{T_x\}$ obeying h, such that $J^A(x) \in T_x$ for every x.

The criticism for the above definition is that the notion is not (or at least, not known to be) degree-invariant. Since a notion of computational weakness (such as traceability) should be at least downwards closed in the Turing degrees, a more robust, degree-invariant definition was suggested by [BDG].

Definition 3.3.5. We say that a Turing degree $\mathbf{a}$ *is jump traceable at* h, if every $\mathbf{a}$-partial computable function has a c.e. trace obeying h.

Clearly if A is jump traceable at h, then $deg(A)$ is jump traceable at some $\tilde{h}$. However the obvious choice for $\tilde{h}$ will grow faster than h, and depends on the choice for the pairing function and the *s-m-n* function. We remark that it is unknown if Definitions 3.3.4 and 3.3.5 are equivalent; the two might be equivalent for some orders and different for others. Nevertheless, we observe that Theorem 3.3.3 implies as a corollary, the following result of Barmpalias, Downey and Greenberg.

Theorem 3.3.6 (Barmpalias, Downey and Greenberg [BDG])**.** *There is a c.e. set A which is not K-trivial, but for every order h such that $\lim_n \frac{h(n)}{n} = \infty$, $deg(A)$ is jump traceable at h.*

To see this, observe that the proof of Theorem 3.3.3 does not use the universality of J^X. Rather we only use the fact that J^X is a partial X-computable function. Hence we could build an identity bounded trace for Γ^{A_0} and Γ^{A_1}, where the functional $\Gamma^X(2^e(2x+1))$ returns $\Phi_e^X(x)$ for all e, x. Therefore, for each e, we can produce a trace for Φ_e^A with bound $\lambda x(2^e(2x+1))$. Then any h such that $\lim \frac{h(n)}{n} = \infty$ has to dominate $\lambda x(2^e(2x+1))$.

As mentioned at the beginning of this section, the search for a combinatorial characterization for the K-trivials remains open. Our result raises a number of interesting questions.

1. How slowly can we allow h to grow, such that there are c.e. sets A and B which are both jump traceable at order h, but $A \oplus B$ is of complete Turing degree? We know from Cholak, Downey and Greenberg [CDG08] that h cannot be arbitrarily slow growing. On the other hand our strategy is rather specific, and does not even work for the order $\frac{1}{2}n$.

2. Is there a pair of sets which cups to $\emptyset'$, but have identity traceable *degrees*? Even though the proof of Theorem 3.3.3 does not require the universality of J^X, we do require that we only need to trace a *single* function.

3. Is every c.e. set which is K-trivial also jump traceable at identity? An analysis of the Decanter method in Downey [Dow] shows that if A is K-trivial, then A is jump traceable at all orders h such that $\sum_{n=1}^{\infty} h(n)^{-1} < \infty$, but it is not known if the method adapts to the case of $h(n) = n$.

3.3.1 The strategy

We construct traces $\{T^i_x\}_{x>0}$ for J^{A_i}, $i = 0, 1$, as well as a Turing functional Γ ensuring that $\emptyset' = \Gamma^{A_0 \oplus A_1}$. The stage s use of this computation is recorded by $\gamma(x, s)$, which we think of as a moving marker pointing at a particular number not yet in $A_0 \cup A_1[s]$. To move the marker $\gamma(x)$, we would have to remove the Γ-axiom we had previously set; to do this we have to put[1] $\gamma(x)$ into either A_0 or A_1. If $\gamma(x)$ is not enumerated then it is not allowed to move. We fix an enumeration $\emptyset'[s]$ of the halting problem, such that at most one number is enumerated per stage. When we pick a *fresh* number at stage s, we mean a number $> s$ and $>$ any number used so far. The construction involves a careful monitoring of box sizes. We think of each T^i_x as being made up of locations, or "boxes", which we will fill with current values of $J^{A_i}(x)[s]$. As more elements enter T^i_x, the "box size" (i.e. the number of available free slots) goes down. This box intuition is convenient, and references of this sort will be made throughout this chapter. This concept also appears in [CDG08], where the *box promotion method* was used to prove several results.

In order to avoid ambiguity, we assume that if $J^\rho(x)[s] \downarrow$ for any string ρ and x, s, then this fact can be seen at the beginning of stage s. Also, we want to distinguish between the two computations $J^\rho(x)$ and $J^{\rho'}(x)$ which may have the same value but have different use $\rho \neq \rho'$. Therefore, instead of enumerating potential values of $J^{A_i}(x)$ into T^i_x, we will enumerate the use of these potential computations (as finite strings) into T^i_x. As long as we ensure that $J^\rho(x) \downarrow$ for some $\rho \subset A_i \Rightarrow \rho \in T^i_x$, then

[1]Hence the use of $\Gamma^{A_0 \oplus A_1}(x)$ is really $2\gamma(x)$. $\gamma(x)$ marks the part of A_0 and A_1 which is accessed by the computation.

A_i would be jump traceable via some suitable transformation of $\{T^i_x\}$. This feature is not necessary, but useful as it allows us to avoid considering different cases. We say that a computation $J^\rho(x)\downarrow$ *has been i-traced* if ρ has been enumerated in T^i_x.

We think of each T^i_x as a collection of x many boxes, which we will each fill with potential jump computations. Initially T^i_x starts off as having box size $x-1$; we also say it is an $(x-1)$-box. $x-1$ represents the fact that T^i_x has not yet been injured, and hence can take $x-1$ many more injuries. Each time a potential jump computation is enumerated in T^i_x and subsequently destroyed, we decrease the box size by one. To ensure jump traceability, all we need to do is to ensure that at all times, no box of size 0 is injured on either side.

To make this compatible with coding requirements, we pick a number $forbid(e)$ for each marker $\gamma(e)$, and then ensure that every time we see $\gamma(e) < j^{A_i}(w)$ for some w of small T^i_w-box size $\leq forbid(e)$ on the A_i-side, we move $\gamma(e)$ out of the way. We do this by enumerating $\gamma(e)$ on the other side, into A_{1-i}, and then pick a fresh marker location for $\gamma(e) > j^{A_i}(w)$. If small box sizes appear on both sides A_0 and A_1, we pick any of the two sides to injure; this does not matter as both of these computations must have newly converged, and we would not have traced either of these computations yet (we arrange for computations to be traced at the end of each stage, so that only the surviving computations are traced). If we ensure that at all times, $0 < forbid(1) < forbid(2) < \cdots$, then no traced 0-boxes can ever be injured. This ensures jump traceability of both A_0 and A_1.

The above plan is general enough to work for all orders h, and yet we know that if h is sufficiently slow-growing, we cannot code $\emptyset'$ into the join of two sets jump traceable at order h. We want to show that the identity $h(n) = n$ grows quickly enough to allow coding. The problem we have clearly not yet considered, is the fact that $\gamma(e)$ might not settle. Let us consider $\gamma(1)$, and suppose that $forbid(1) = F$ (which never changes since $\gamma(1)$ is of the highest priority). Now $\gamma(1)$ might be put into A_0 because some box of size $\leq F$ has converged above $\gamma(1)$ on the A_1-side. Our strategy ensured that there was no box *which is already traced*, and of size $\leq F$ converging above $\gamma(1)$ on the A_0-side. However, there might be boxes of size $F+1$ with convergent computations above $\gamma(1)$ on the A_0-side, which have already been traced. Thus, in trying to avoid boxes of small size on the A_1-side, we might create more boxes of size F on the A_0 side. These new small-sized boxes we created on the A_0 side might come back and force us to create more boxes of size F on the A_1-side, and so on. Hence $\gamma(1)$ never settles since it spends all of its time avoiding these small boxes.

This obstacle cannot be overcome if h is sufficiently slow growing. The intuition is that if h grows very slowly, then every time we avoid a single box on one side, we will create a lot of new F-boxes on the other side. However, if h is the identity, then the creation of new F-boxes is controlled; avoiding a single box on one side creates (more or less) just a single extra F-box on the other side. We make this more precise: we want to argue that at identity $h(n) = n$, the marker $\gamma(1)$ is only moved finitely often. $\gamma(1)$ moves only to avoid an F-box T^i_w, and once $\gamma(1)$ moves to its new location the same box T^i_w is never a problem again on the A_i side, since the computation will be preserved forever. We wait until we have done this $F + 1$ times on each side, at some stage s. We can ensure that at each stage t we only make traces into T^i_x for $x < t$, hence at stage s, T^i_s is empty and hence must be an $(s - 1)$-box for both i (this is where we make use of the fact that h is identity). In future, how many times may T^0_s be injured? As described above, each time T^0_s is injured, we brush aside some T^1_w, of current size $\leq F$. We may assume that $w < s$, otherwise T^1_s will become an F-box before T^0_s does, in which case we argue with T^1_s in place of T^0_s. Each of these T^1_w only blocks $\gamma(1)$ once, and there are altogether only $s - 1 - (F + 1)$ many of these boxes to consider, since at stage s we had already had $F + 1$ many of these boxes converging below $\gamma(1)$. Therefore, T^0_s can be reduced to at most a $(F + 1)$-box, hence neither of T^0_s nor T^1_s can become an F-box. Hence this process stops, and $\gamma(1)$ eventually settles. An inductive argument can be applied for $\gamma(e)$.

Note this observation works just as well if $h(n) = n - c$ for any constant c, but no longer works if h is unbounded away from the identity, e.g. $h(n) = \frac{n}{2}$. Any attempt to improve Theorem 3.3.3 will have to address the issue of *amplification*. That is, the opponent might pursue some sort of strategy along the lines of the Decanter or box promotion methods, and amplify small errors repeatedly. In this construction, we stop the opponent from doing this by isolating his actions on each of our requirements. We do so by setting up barriers; if the opponent has caused us to make an error on $\gamma(x)$ then we isolate his actions on $\gamma(x+1), \gamma(x+2), \cdots$, so that errors we make on these cannot be amplified to cause us grief on $\gamma(x)$. The identity function grows just fast enough to permit us to do that. The formal construction and verification fills in the rest of the details.

3.3.2 Notations

We record the size of T_k^i by the parameter $size_i(k, s)$, where $k > 0$. This is initially set to $size_i(k, 0) = k - 1$ for all $k \in \mathbb{N} - \{0\}$ and $i = 0, 1$. At times we decrease $size_i(k)$ when T_k^i is injured, and when $size_i(k)$ reaches 0 then the use of $J^{A_i}(k)$ must be preserved when the computation next converges. We ensure $size_i(0, s) \leq size_i(1, s) \leq size_i(2, s) \leq \cdots$ for each i, s. Thus when a T_x^i box is injured we will have to decrease $size_i(k)$ for all k such that $size_i(k) = size_i(x)$; we pretend that all these boxes are injured as well. Formally, during the construction, before we enumerate a number x into A_i at stage s, we will *update the box size* by doing the following: let $X = \{k < s : J^{A_i}(k)[s] \downarrow \text{with use} > x, \text{and has already been } i\text{-traced}\}$. For every k' such that $size_i(k') = size_i(k)$ for some $k \in X$, we decrease $size_i(k')$ by 1. That is, if we decrease $size_i(k)$ for $k \in X$, we will also decrease $size_i(k')$ for all k' in the "same block" as k. We have a variable $forbid(x, s)$ which represents the region forbidden to $\gamma(x)$ - i.e. the largest number w such that $\gamma(x)$ is not allowed to be below the use of $J^{A_i}(y)$ for $i = 0, 1$ and any T_y^i with box size $\leq w$.

3.3.3 The construction

At stage 0 initialize the parameters according to the following. Set $size_i(k) = k - 1$ for all $k > 0$ and $i = 0, 1$. Set $forbid(x) = x$ for all x, and do nothing else. Suppose now that $s > 0$. There are three parts to the construction:

1. First, check if any correction to Γ is necessary: if there is some $x \in \emptyset'[s] - \emptyset'[s-1]$, we update the box size and enumerate $\gamma(x)$ (if defined) into A_0. If no correction to Γ is necessary, we extend the definition of Γ by setting $\Gamma^{A_0 \oplus A_1}(x)[s] \downarrow= \emptyset'(x)[s]$ for the least $x < s$ where no axioms currently apply; do this with a fresh $\gamma(x)$-use.

2. Pick the least $x < s$ such that $\gamma(x,s)\downarrow$ and $\gamma(x,s) < j^{A_i}(w,s)$ for some i and w such that $size_i(w) \leq forbid(x,s)$, and coding is not yet done (i.e. $x \notin \emptyset'[s]$). Note that the $J^{A_i}(w)[s]$ computation currently blocking $\gamma(x)$ might not have been i-traced, but we will need to take action for it anyway. Update the relevant box size, and enumerate $\gamma(x,s)$ into A_{1-i} (choose the smaller i, if we have a choice) where the above holds. Set $forbid(y) = s + y - x$ for all $y > x$.

3. For each $i = 0, 1$, and each $k < s$ such that $J^{A_i}(k)[s]\downarrow$, such that A_i had not changed below the use due to (1) and (2) above, we enumerate the use into T^i_k. Note that at stage s we only trace jump values $J^{A_i}(k)$ for $k < s$.

3.3.4 Verification

It is clear that for each $i = 0, 1$, and $x > 0$, if $J^{A_i}(x)\downarrow$, then $A_i \upharpoonright j^{A_i}(x) \in T^i_x$. Suppose that for some i, x, we have $|T^i_x| > x$. There are strings $\rho_0, \rho_1, \cdots, \rho_x$ which have been enumerated into T^i_x in that order. By the use principle, for each $j < x$, there must be a change in $A_i\upharpoonright|\rho_j|$ between the enumerations of ρ_j and ρ_{j+1}. The first such change would cause $size_i(x)$ to be decreased by 1, and therefore when ρ_{x-1} is enumerated, we must already have $size_i(x) \leq 0$. Suppose ρ_{x-1} is traced under step 3 of the construction, at some stage t. Hence $A_i\upharpoonright|\rho_{x-1}|$ was unchanged throughout stage t. If $\gamma(0)$ was defined and below $|\rho_{x-1}|$, then we would have enumerated $\gamma(0)$ under step (2); in any case at the end of stage t, $\gamma(0)$ must be undefined, so there can be no markers below $A_i \upharpoonright |\rho_{x-1}|$ (which have not yet been coded). This is a contradiction as there cannot be any change below $A_i\upharpoonright|\rho_{x-1}|$ after stage t. Hence, A_0 and A_1 are both jump traceable with respect to the identity function.

Next, we have to establish the crucial fact: for each e, $\gamma(e)$ eventually settles. Suppose the result holds for all $e' < e$. Hence $F = \lim forbid(e)$ exists. Let s_0 be a stage where $forbid(e)$ has settled. For each $s \geq s_0$, we let $\Theta_i[s] = \{x : size_i(x,s) \leq F\}$ (this is an initial segment of $\mathbb{N}$, since we keep $size$ nondecreasing), and let $\bar{\Theta}_i[s] = \{x < s : size_i(x,s) > F\}$ be the rest. We further split $\Theta_i[s]$ into two parts: $\Theta^-_i[s] = \{x \in \Theta_i[s] : J^{A_i}(x)[s]\downarrow$ and has been i-traced$\}$, and the rest goes in $\Theta^+_i[s] = \Theta_i[s] - \Theta^-_i[s]$. Basically, Θ^-_i contains all those boxes with high e-priority, which no longer need to be considered by $\gamma(e)$, while Θ_i^+ contains all those high priority boxes which might act and block $\gamma(e)$ at any time in the future.

Suppose for a contradiction, that $\gamma(e)$ moves infinitely often after s_0. Hence $e \notin \emptyset'$. Define the function $gap_i(s) = |\Theta_i^-[s]|$, where the expression is evaluated at the end of stage s. A contradiction is derived using the following lemmas. The first lemma says that each time a marker $\gamma(e)$ is moved, we never promote a box forbidden to it:

Lemma 3.3.7. *Let s be a stage, such that $\gamma(r)$ was enumerated in A_i at stage s. If $size_i(k)$ was decreased from $a+1$ to a in the corresponding update of box size, then necessarily $a \geq forbid(r)$.*

Proof. Suppose the lemma holds for all stages $s' < s$. We prove the case for stage s. There must be some k' such that $J^{A_i}(k')[s] \downarrow$ with use $> \gamma(r)$, which has already been i-traced. That use must have been i-traced at some stage $t < s$, which means that A_i had not changed below that use during the entire stage t. We must have $\gamma(r)[s] = \gamma(r)[t]$, which means that $size_i(k')[t] > forbid(r,t) = forbid(r,s)$, lest $\gamma(r)$ be moved at stage t. If $size_i(k')[t] = a+1$ then we are done, so assume that $size_i(k')$ is decreased from $a+2$ to $a+1$ between stages t and s, and this has got to be due to some $\gamma(r')$ being moved, for some $r' > r$. By induction hypothesis, we have $a+1 \geq forbid(r') > forbid(r,s)$. □

Lemma 3.3.8. *There is $s_1 > s_0$ such that $gap_i(s_1) > F$ for both $i = 0, 1$.*

Proof. Note that for all $s > s_0$, we have $gap_i(s+1) \geq gap_i(s)$ for both i (by Lemma 3.3.7). Furthermore each time the marker $\gamma(e)$ was put in A_{1-i}, it must be under step 2, so each enumeration corresponds to an increase in gap_i. If $gap_1(\cdot)$ is bounded, then only finitely many enumerations are made into A_0. Hence no box of current size $\leq F+1$ can ever be promoted once enumeration of $\gamma(e)$ into A_0 stops, by Lemma 3.3.7, since only markers $\gamma(e+1)$ and above can be enumerated on the A_0-side. This means that $\Theta_0[s]$ reaches a limit, say θ. Since obviously $gap_0(s) \leq \theta$, it follows that $gap_0(\cdot)$ is bounded as well, contradicting the fact that $\gamma(e)$ is moved infinitely often. Hence, neither gap_0 nor gap_1 can be bounded. □

Lemma 3.3.9. *For all $t \geq s_1$ and both i, we have $size_i(s_1, t) > F$.*

Proof. Note that $size_i(s_1, s_1) = s_1 - 1$. Suppose on the contrary, that at some stage $s_2 > s_1$ and some i, we update box size causing $size_i(s_1)$ to become $= F$. Since box sizes are updated on only one side each time, we assume that we still have $size_{1-i}(s_1, s_2) > F$. Each time $size_i(s_1)$ is decreased between s_1 and s_2, it has to be due to $\gamma(e)$ being enumerated into A_i, as we may assume that $forbid(e') > s_1$ at the end of stage s_1 for all $e' > e$. Each time $\gamma(e)$ enters A_i, we must also have a corresponding decrease in $|\Theta^+_{1-i}|$. That is, there is some number which drops out of Θ^+_{1-i} (and goes into Θ^-_{1-i}), and this has to be one of the numbers in the set $\Theta^+_{1-i}[s_1] \cup \bar{\Theta}_{1-i}[s_1]$. Hence, $size_i(s_1)$ can be decreased at most $|\Theta^+_{1-i}[s_1] \cup \bar{\Theta}_{1-i}[s_1]| = s - 1 - |\Theta^-_{1-i}[s_1]| = s - 1 - gap_{1-i}(s_1) < s - 1 - F$ times during the interval from s_1 to s_2. Since $size_i(s_1, s_1) = s_1 - 1$, a contradiction follows. $\square$

The above lemmas show that for both i, Θ_i is of bounded size. We get a contradiction to the fact that $\gamma(e)$ is moved infinitely often. Lastly, it is not hard to see that Γ is total, and gives a correct reduction. This ends the proof of Theorem 3.3.3.

3.4 A first glimpse of hyper jump traceability

Part of the motivation of this chapter is to study variations of strong jump traceability, by looking at relativizations. In particular we want to investigate variations in strong jump traceability which are computationally weak. These motivations suggest the following definition.

Definition 3.4.1. We say that A is *strongly jump traceable by* X, if for every X-computable order h^X, there is a total computable function g, such that for all x, we have $|W^X_{g(x)}| \leq h^X(x)$ and $J^A(x) \in W^X_{g(x)}$.

We let $SJT(X)$ denote the class of c.e. sets strongly jump traceable by X. Note that this is not a true relativization, which would call for $J^{A \oplus X}$ to be traced instead of just J^A. There is no difference if $X = \emptyset$; in both cases we have the class of strongly jump traceable c.e. sets. However things behave a little differently if we consider an arbitrary $X >_T \emptyset$. Having X as an oracle helps us in the sense that we have more traces at our disposal (now we have all X-c.e. traces instead of just c.e. traces). However, we also suffer a drawback because we now have to trace J^A respecting more order functions (we have to respect all X-computable orders instead of just computable orders). Therefore it is not immediately clear that if A is strongly jump traceable, then it must also be strongly jump traceable by all sets X. Some sets X might produce very slow growing X-order functions which we cannot trace using only X-c.e. traces. The only thing we are sure of is that $A \leq_T X \Rightarrow A \in SJT(X)$. In fact, in Theorem 3.4.5 we produce a strongly jump traceable c.e. set A such that $A \notin SJT(X)$ for some c.e. X.

The first indication that we can further extend the notion of strong jump traceability in a meaningful way comes from the investigation of the transitivity of the binary relation "$A \in SJT(X)$".

Lemma 3.4.2. *Suppose that $g(x)$ is a total computable function. Then, there is a total computable function $\alpha(x, n)$, such that for all numbers x and n, and all sets A,*

$$J^A(\alpha(x,n)) = \begin{cases} n^{th} \text{ value in the} & \text{if } |W^A_{g(x)}| \geq n, \\ \text{enumeration of } W^A_{g(x)}, & \\ \uparrow, & \text{if } |W^A_{g(x)}| < n. \end{cases}$$

Proof. By the s-m-n Theorem. □

Suppose we wanted to show that the relation $A \in SJT(X)$ is transitive. We might make a first attempt at proving transitivity below $\emptyset$, and then try and relativize the proof. That is, we can try first of all to show that if A is strongly jump traceable in X, and X is strongly jump traceable, then A is also strongly jump traceable. While this is true (Corollary 3.4.4), the proof does not actually relativize to give transitivity in the general setting. The following Theorem 3.4.3 gives the correct relativization:

Theorem 3.4.3. *Suppose that $A \in SJT(B)$ and $B \in SJT(C)$, such that $C \leq_T B$. Then, $A \in SJT(C)$.*

Proof. Let h be a C-order. Define

$$p(x) = \sqrt{h^C(x)} \text{ (rounded down).}$$

Then, p is a B-computable order, and so there is some total computable g such that for all x,

(i) $|W^B_{g(x)}| \leq p(x)$, and

(ii) $J^A(x) \in W^B_{g(x)}$.

Let α be the function from Lemma 3.4.2. Define the C-computable function q by : $q(z) = \sqrt{h(x)}$, where x is the least number such that $z \leq \alpha(x, i)$ for some $i \leq p(x)$. Then, q is a C-order, and so there is some total computable g' such that for all x,

(i) $|W^C_{g'(x)}| \leq q(x)$, and

(ii) $J^B(x) \in W^C_{g'(x)}$.

Finally, we define the uniformly C-c.e. sequence $\{V_x\}_{x \in \mathbb{N}}$, by letting

$$V_x = \bigcup_{i=1}^{p(x)} W^C_{g'(\alpha(x,i))}$$

We can easily see that for all x,

(i) $|V_x| \leq p(x)q(x) \leq h(x)$, and

(ii) $J^A(x) \in V_x$,

thus giving a trace for J^A from C. □

Corollary 3.4.4. *If $A \in SJT(B)$, and B is strongly jump traceable, then A is strongly jump traceable.*

We need the extra condition of $C \leq_T B$ to ensure that we do not get more order functions as we pass from one side to the other of the binary relation. Corollary 3.4.4 is interesting on its own, in that it says that any set which is "extremely low" over another which is extremely low, must itself be also extremely low. This brings us back to the question as to whether this is true in general. We provide a negative answer to this:

Theorem 3.4.5. *There are strongly jump traceable c.e. sets A and B, such that $A \notin SJT(B)$.*

This theorem has a few interesting consequences. Firstly it shows that "$A \in SJT(X)$" is not a transitive relation, and so this relation cannot be used to generate a degree structure in the usual way. It also answers a question regarding the relationship of SJT with upper and lower Turing cones. In particular, while every set is strongly jump traceable by any set which computes it, it is however not true that every set is strongly jump traceable by a set which it computes. This is not even true for sets which join to it, i.e. if $A \equiv_T A_0 \oplus A_1$ then $A \in SJT(A_0)$ is not true in general.

Thirdly, one might expect that if a set B is very low in terms of its power when serving as an oracle, then every set which is strongly jump traceable is also strongly jump traceable by B. This is not the case. Lastly and perhaps most importantly, this theorem shows that the class of strongly jump traceable c.e. sets and the class $\mathcal{H} := \bigcap\{SJT(W) \mid W \text{ is c.e.}\}$ are different. This suggests that we can look to $\mathcal{H}$ as a possible candidate for an even stronger version of computational lowness. This will be taken further in the next section, and will form the central theme for the rest of the chapter.

We remark that Nies [Nie09] has defined relativization of strong jump traceability in a different way. Namely he defines *A to be strongly jump traceable plop X*, if for every computable order h, there is an X-c.e. trace for J^A obeying h. Since only computable orders are considered, this relation is indeed transitive. Nies defines the general notion of a *weak reducibility*, and denotes "A is SJT plop X" by "$A \leq_{SJT} X$". More on these weak reducibilities will be covered in Section 6.3.

Name	Relativization	Strong notion
A is sjt relative to X	Full relativization. Considers X-computable orders, X-c.e. traces and traces $J^{A\oplus X}$	Computable
$A \in SJT(X)$	Considers X-computable orders, X-c.e. traces and traces J^A	$\mathcal{H}$
$A \leq_{SJT} X$	Considers computable orders, X-c.e. traces and traces J^A	sjt

The above table summarizes the different levels of (partial) relativizations involved. The strong notion refers to the class of c.e. sets which are strongly jump traceable relative/by/plop every c.e. set. To see that the strong notion for full relativization implies being computable, note firstly that the relativized version of the join theorem in [CDG08] reads: if A and B are computably enumerable relative to X, and strongly jump traceable relative to X, then $A \oplus B$ is also strongly jump traceable relative to

X. Suppose A is c.e. and is strongly jump traceable relative to every c.e. set. We mention a result of Jockusch, Li and Yang:

Theorem 3.4.6 ([JLY04]). *If D is c.e. and noncomputable, then there is a low$_2$ c.e. set E such that $D \oplus E$ is high.*

By the above theorem, there would be a low$_2$ c.e. set B such that $A \oplus B$ is high. By the relativized version of the join theorem, this means that $A \oplus B$ is strongly jump traceable relative to B. Since $A \oplus B \leq_T \emptyset'$, it is not hard to see that $(A \oplus B)' \leq_T B'$, which is a contradiction unless, of course, A is computable.

3.4.1 Requirements

We now prove Theorem 3.4.5. We build the c.e. sets A and B, and a Turing functional Ψ to meet the requirements :

$\mathcal{N}_e$: If h_e is an order, make $A \oplus B$ jump traceable relative to h_e.

$\mathcal{P}_e$: Defeat the e^{th} trace. That is, for some x, either $|T_x^e| \geq \Psi^B(x)$, or else $J^A(x) \notin T_x^e$.

Here, we let $\{T_x^e\}_{x \in \mathbb{N}}$ be the e^{th} B-trace, in some effective listing of all traces computing with an oracle. For simplicity we drop the oracle B from the notations. We let $J^X(e)[s]$ be the value of the universal jump function $\{e\}^X(e)[s]$ at stage s. The use of $J^X(e)[s]$ (if convergent) is $j^X(e,s)$. We also let $\{h_e\}_{e \in \mathbb{N}}$ be an effective list of all partial computable functions.

When we say that we pick a *fresh* number x at stage s, we mean that we choose x to be the least number $x > s$, and $x > 1+$any number used or mentioned so far. We drop the stage number from the notations if the context is clear.

3.4.2 Description of strategy

The proof of this theorem uses the ideas in Theorem 2.3.1 closely. We refer the reader there if he has any doubts about the basic techniques. How are we going to make use of the techniques in Theorem 2.3.1? It is impossible to make A strongly jump traceable, yet at the same time make A not jump traceable relative to a computable order function $\tilde{h}$. However if we allow $\tilde{h}$ to be computable in a c.e. oracle, we can combine the positive and negative requirements stated in Section 3.4.1. We describe how to do this. To make A and B strongly jump traceable, we make $A \oplus B$ strongly jump traceable.

We describe exactly how we intend to carry out the strategy for a single $\mathcal{N}_e$. We need to make $A \oplus B$ jump traceable respecting h_e. Suppose α is a node on the construction tree which is assigned the requirement $\mathcal{N}_e$. It splits its task into infinitely many substrategies $ST_0, ST_1, \cdots$. For each $k \in \mathbb{N}$, the k^{th} substrategy ST_k works by the following: it waits for $h_e(k) \downarrow$, and when $J^{A\oplus B}(k)[s]$ next converges, we would enumerate the value into V_k^α (the sequence $\{V_x^\alpha\}_{x\in\mathbb{N}}$ is built at α), and restrain $A \oplus B$ on the use. At this point in time, we set $size_k^\alpha = h_e(k) - 1$ (where $size_k^\alpha$ denotes the corresponding α-box size). When $size_k^\alpha = 0$, V_k^α is totally filled and any restraint α imposes for it must be permanent. If we arrange for each substrategy ST_k to be assigned to an entire level below α, we immediately meet with a technical obstacle. Recall that in the discussion in Theorem 2.3.1, the positive requirements had priority (relative to some ST_k), which was determined dynamically. This would not be easy to arrange on a tree of strategies.

Note that we could however, arrange for *all of* the α substrategies to be carried out at α itself. This means that α could impose an ever increasing restraint on the positive strategies below it, even though each substrategy ST_k contributes a finite amount. To get around this problem, we arrange for there to be infinitely many restraint functions $r_0, r_1, \cdots$, where r_k is the restraint function for ST_k, and let different positive strategies below α, be restrained by a different r_k. Suppose each positive strategy below α only wants to enumerate once. We could then let the first positive strategy obey restraint r_0, the second positive strategy obey restraints $\max\{r_0, r_1\}$, and so on. This works in the case when h is the identity order function (otherwise we just make suitable adjustments). For more details on this, see Theorem 7.3 of [Dow].

Now we describe the positive requirements. We first consider (the weaker case of) only making A strongly jump traceable. We also make A not jump traceable relative to B via the order $\tilde{h} = \Psi^B$. They work in almost the same way as in Theorem 2.3.1, with two main differences (due to the fact that we now have an oracle B):

1. To defeat the e^{th} trace, we want it to fill up with trash. We can stimulate responses from T^e_x by emulating the computation $J^A(x)$ at some x. However, when some number enters T^e_x, it will have a corresponding B-use. To keep the number in T^e_x, we have to now preserve B on this use. This gives the positive node an extra responsibility - holding B on some use. This extra restraint is finitary and can be very easily made compatible with the rest of the construction.

2. $\tilde{h}$ is now built globally. Suppose a positive node σ needs to make the statement of $\mathcal{P}_e$ true at some x, with $\tilde{h}(x) = \Psi^B(x) = 2$. Now we have the ability to change Ψ^B whenever we want. σ might start off by picking a follower x and setting $\Psi^B(x)[s] = 2$. At a later stage σ might be blocked by an increased A-restraint and hence cannot proceed with diagonalization with x anymore. It will then pick a fresh $x' > x$ and clear all $\Psi^B(y)$ axioms for $y > x$ which may have been set by other positive nodes in the meantime. σ can do this by enumerating $\psi(x)$ into B, and redefine $\Psi^B(y) = 2$ for all $x \leq y \leq x'$, and proceed with diagonalization with new follower x'. It is not hard to arrange things so that the A-restraint on σ is finite, so that this injury only happens finitely often.

We have no problems putting together the positive and negative requirements, if we only needed to make A strongly jump traceable, because each time σ needs a new follower for diagonalization, we simply change B to create more followers for σ. The problem now is that to make $A \oplus B$ strongly jump traceable, the negative requirement $\mathcal{N}_e$ also needs to set up restraints to protect B. The following summarizes the actions of the positive and negative nodes:

$\mathcal{P}_e$	Puts numbers into A and B. Prevents numbers from entering B.
$\mathcal{N}_e$	Prevents numbers from entering A and B.

Each enumeration into B made by σ will now cause boxes above σ to be promoted as well. Thus, in the process of trying to create more followers for diagonalization, σ will further promote boxes which will return and block σ when it starts the next round of diagonalization.

To take a first step towards solving this problem, we will first ensure that σ makes less enumerations into B. Enumerations into B will have to obey restraints from above, since we have boxes tracing $J^{A\oplus B}$. To cut down on the number of enumerations made into B, σ will have to do a little more work. Before σ begins any diagonalization attempt, it will first compute the relevant thresholds $C_{n,-1}, \cdots, C_{n,n-1}$ for some selected n. These thresholds will be computed based on the procedure $\Lambda(\min\{h_{e_0}, \cdots, h_{e_k}\})$, where σ believes that $h_{e_0}, \cdots, h_{e_k}$ are orders. Of course these values will not return if the h_{e_i} are not true orders, but we get to change our mind on Ψ^B, so we can carry on with the rest of the construction while waiting for these values to return. If and when these values return, σ will then adjust Ψ^B accordingly by changing B once, and setting aside enough followers x with $\Psi^B(x) = n$. This ensures that σ only changes B once.

Note that σ will not be able to change B whenever it wants, because we might have boxes of small size blocking B. On the other hand, we have to be careful when we allow σ to change B, because whenever σ changes B, it will promote some boxes above σ (remember these are boxes tracing $J^{A\oplus B}$). If we are not careful, σ will promote boxes to take on very small sizes and affect the other positive nodes above σ.

In view of these considerations, we will arrange for σ to have a separate threshold value, called $setter(\sigma)$, which is to be used only when σ is in the preliminary stage of setting the parameters needed for diagonalization. This number is chosen to be large, so that σ does not interfere with the positive requirements above σ (i.e. σ does not promote any boxes to have size $< setter(\sigma)$). When σ is computing the thresholds it needs for diagonalization, it will monitor all boxes currently of size $\leq setter(\sigma)$. Once any of them imposes an $A \oplus B$ restraint above $\psi(x)$, σ will have to abandon its current set of parameters, and pick new $n' > n$ and $x' > x$. So long as no new $setter(\sigma)$-boxes are created, σ will eventually be able to begin diagonalization.

A final technical point to consider: σ makes an enumeration into B before it actually begins diagonalization. Therefore, boxes are already promoted by σ before σ begins its Λ-strategy. Fortunately, this sort of promotion due to changes in B only happens once, so when σ actually begins diagonalization, it knows that a current b-box it encounters was actually a $(b+1)$-box. The threshold values can be pre-chosen to take this into account.

3.4.3 Construction tree layout

The construction takes place on a subtree of the full binary tree. Nodes of length $2e$ are assigned the requirement $\mathcal{N}_e$, with outcomes $\infty <_{left} f$. This stands respectively for the Π^0_2 (Σ^0_2) fact that h_e is (is not) an order function. The positive requirements have to act based on guesses to these outcomes, and have to deal with increased $(A \oplus B)$-restraint. Nodes of length $2e+1$ are assigned the requirement $\mathcal{P}_e$, with a single outcome 0; these nodes are each involved in only finitely much action.

Let $\alpha <_{left} \beta$ denote that α is strictly to the left of β, i.e. there is some $i < \min\{|\alpha|, |\beta|\}$ such that $\alpha\restriction i = \beta\restriction i$ and $\alpha(i) <_{left} \beta(i)$. We say that α *is a $\mathcal{Q}$-node* if α is assigned the requirement $\mathcal{Q}$. $\mathcal{N}_e$ nodes are *negative nodes* wanting to impose restraint on $A \oplus B$, while $\mathcal{P}_e$-nodes are *positive nodes* wanting to put things into A and B. Note that even though we consider a $\mathcal{P}_e$-node to have primarily a positive role, there will be times when it will want to preserve a segment of B. This happens when trash gets filled up in the B-trace, and we want to keep it filled. There will only be however, a finite amount of B-restraint due to this.

3.4.4 Notations used in the formal construction

At each $\mathcal{N}_e$-node α, we build a u.c.e. sequence $\{V^\alpha_x\}_{x\in\mathbb{N}}$; the purpose is to trace $J^{A\oplus B}$ in the event that h_e turns out to be an order. We define the length of convergence for h_e at stage s, to be:

$$\begin{aligned} l(e,s) = \max\{y < s \mid (\forall x \le y)\ (h_{e,s}(x)\downarrow\ \wedge\ h_e(x) \ge h_e(x-1)) \\ \wedge\ h_e(y) > h_e(y-1)\}. \end{aligned}$$

Sometimes we will write $l(\alpha, s)$ in place of $l(e,s)$, and h_α in place of h_e. Since h_e is (partial) computable, it is clear that $l(e,s)$ is nondecreasing over time, and $l(e,s) \to \infty$ iff h_e is an order. We let $size^\alpha_k[s]$ denote the *size of the V^α_k-box at stage s*. It records the number of injuries the V^α_k-box can still take. At the beginning, $size^\alpha_k$ is set to $h_e(k)-1$, and will be reduced by 1 each time a $J^{A\oplus B}(k)$-computation is injured after being traced in V^α_k. When $size^\alpha_k$ reaches 0, the restraint that α imposes for $A \oplus B$ must be obeyed by all.

Each $\mathcal{P}_e$-node α is in charge of diagonalizing the e^{th} B-trace. We do this by enumerating a functional Δ_α^A with index x for some x, which simulates the value of $J^A(x)$. If at any point in time we want give up all axioms enumerated into Δ_α^A, we have to pick a new index $x' > x$ and work on $J^A(x')$. There are a few other parameters associated with α:

- We let $x(\alpha, s)$ denote the index of the functional which α is enumerating at stage s, with A-use $\delta(\alpha, s)$. These indices are chosen from an infinite list of indices supplied by the Recursion Theorem.

- $\mathit{region}(\alpha, s)$ denotes the number n, such that α will define $\Psi^B(x) = n$ for every index x that it uses after stage s. In other words, this represents the number of elements α has to enumerate into A, and is also the amount of unwanted trash that α has to fill T_x^e up with, at some x.

- We record the number of elements that α has managed to force into $T_{x(e)}^e$ by the parameter $\mathit{attempt}(\alpha, s)$. When this parameter value reaches $\mathit{region}(\alpha)$, then α's work would be done and is permanently satisfied.

- $\mathit{setter}(\alpha, s)$ denotes the smallest number q such that α is allowed to promote $(q+1)$-boxes while setting the axioms for Ψ^B.

The stage s B-use of the functional $\Psi^B(x)$ that we are enumerating is denoted by $\psi(x, s)$. To ensure that Ψ^B is total and nondecreasing, we adopt the following convention: when we define $\Psi^B(x)[s] \downarrow= y$ with use u at a particular stage s, we mean that we enumerate the axioms $\langle x', y, B_s \upharpoonright u+1 \rangle$ for all $x' \leq x$, where no other axioms currently apply with input x'. Also set $\psi(x', s) = u$ for all such x'. Note that Ψ^B is maintained globally, so this ensures that its axioms are consistent regardless of which portion of the construction tree we visit.

If α is a negative node, then for each $n, s \in \mathbb{N}$, we let

$$S(\alpha, n)[s] = \sum_{r=1}^{n} r \cdot |\{k < l(\alpha, s) : \mathit{size}_k^\alpha[s] = r - 1\}|.$$

For a positive node σ, we let

$$S(\sigma, n)[s] = \sum_{\beta \in Z^-(\sigma)} S(\beta, n)[s],$$

where $Z^{-}(\sigma) := \{\beta \subset \sigma \mid \beta \text{ is negative, and } \beta^\frown\infty \subseteq \sigma\}$; these are all the negative nodes $\beta \subset \sigma$ which have to trace $J^{A\oplus B}$ at a certain order. Informally, the value $S(\alpha, n)[s]$ denotes the maximum number of different values $j^{A\oplus B}(k, s)$ can take, for the set of k's such that $size^{\alpha}_{k}[s] < n$. The number $S(\sigma, n)[s]$ is used to provide a bound on the number of times σ can be blocked by some current $(n-1)$-box (or less) of some negative $\beta \subset \sigma$. The threshold values will be computed during the construction itself, and their values will depend on the current observed situation.

The parameters $\{C_{n,k} \mid n > 0 \ \wedge \ k < n\}$ and $\{I_n \mid n > 0\}$ help us keep track of the threshold values, and as mentioned above, will only be computed during the construction. These parameters are all set to $\uparrow$ initially. The order function Ψ^B will have domain divided into intervals which we call *regions*. The n^{th} region will consist of all the numbers x such that $\Psi^B(x) = n$; this is related to the parameter $region(\alpha, s)$ mentioned above in the following way: the values $C_{n,-1}, C_{n,0}, \cdots, C_{n,n-1}$, and I_n are associated with the n^{th} region, and will get their values evaluated and assigned by the positive node α such that $region(\alpha) = n$. Once α sets these parameter values, it will use them to define $\Psi^B = n$ on the n^{th} region. Informally, $C_{n,k}$ represents the critical threshold value of α, when $attempt(\alpha) = n - 2 - k$. I_n represents a bound on the number of indices that α needs to use for the n^{th} region.

We describe the use of indices in some detail: there will be infinitely many indices $\nu_0 < \nu_1 < \cdots$ set aside for use by the positive nodes. If α is a positive node then $x(\alpha)$ is always chosen from this list. Once α finishes computing all of the threshold values it needs, it will set Ψ^B to a constant value over each interval $\{x \in \mathbb{N} \mid \nu_{i-1} < x \leq \nu_i\}$, for all ν_i in the $region(\alpha)^{th}$ region. Thus, whenever we refer to $x(\alpha)$ or Ψ^B, we are referring to the values modulo the intervals partitioned by the ν_i's. That is, $\Psi^B(i)$ will refer to the (common) value of $\Psi^B(x)$ for all $\nu_{i-1} < x \leq \nu_i$, and we write $x(\alpha) = i$ instead of $x(\alpha) = \nu_i$.

For a positive node α and stage s, we define $threshold(\alpha, s)$ by

$$threshold(\alpha, s) = C_{r, r-2-a},$$

where $r = region(\alpha, s)$ and $a = attempt(\alpha, s)$. This represents the current threshold value that α has to obey, based on its progress in its atomic strategy.

When we *initialize a negative node* α at stage s, we set $V^{\alpha}_{x} = \emptyset$ for all x, and set

$size^{\alpha}_{x}[s] = h_{\alpha}(x) - 1$ for all $x < l(\alpha, s)$. As for a positive node α, there are three ways in which the atomic strategy of α may be restarted.

1. *A (full) initialization to* α: we set $region(\alpha), x(\alpha), setter(\alpha)$ and $\delta(\alpha)$ all $\uparrow$, and set $attempt(\alpha) = 0$. In this case, α has to restart its strategy due to reasons that it had not foreseen, for example due to an incorrect guess, or some positive node acting above α. All parameter values are cancelled.

2. *A self-imposed initialization*: we set $region(\alpha), x(\alpha)$ and $\delta(\alpha)$ all $\uparrow$, and set $attempt(\alpha) = 0$. This happens if the following scenario takes place. While α is waiting to compute its threshold values, some higher priority box of size $\leq setter(\alpha)$ imposes restraint on $A \oplus B$, above $\psi(x(\alpha))$. Since α always has to respect boxes of size at most $setter(\alpha)$ when setting its Ψ^B-axioms, this means that the current position of $x(\alpha)$ is too small. So, we have to abandon the current $x(\alpha)$ value, *as well as* the current $region(\alpha)$ value, since α is no longer able to effect changes in Ψ^B below the current $region(\alpha)^{th}$-region.

   Such initializations will be performed only when α is visited. We want to distinguish between full and self-imposed initializations because we have to be careful about when we cancel the $setter(\alpha)$ parameter.

3. *A reset of* α: the third way to disrupt the atomic strategy of α, is through the activity of some node in $Z^{-}(\alpha)$. α is fully prepared for injury of this sort, and would have reserved enough indices in the $region(\alpha)^{th}$ region for this. In this case we do the following. If $attempt(\alpha, s) = region(\alpha, s)$ (i.e. α is permanently satisfied) or $x(\alpha) = region(\alpha, s) + I_{region(\alpha,s)}$ (i.e. α has run out of indices), do nothing. Otherwise increase $x(\alpha)$ by 1, set $\delta(\alpha) =\uparrow$, and set $attempt(\alpha) = 0$.

If σ is a positive node and $k \in \mathbb{N}$, we define the *k-restraint function on σ* at stage s to be

$$\begin{aligned} r(\sigma, k, s) = \max\{ j^{A\oplus B}(p, s) \mid J^{A\oplus B}(p)[s]\downarrow \text{ for some } \beta \in Z^{-}(\sigma) \ \wedge \\ p < l(\beta, s) \ \wedge \ size^{\beta}_{p}[s]\downarrow \leq k\}. \end{aligned}$$

That is, $r(\sigma, k)$ represents the total amount of restraint on $A \oplus B$ imposed on σ, by some current k-box (or less) above σ.

We say that a positive $\mathcal{P}_e$-node α *requires attention* at stage s, if $\mathit{attempt}(\alpha, s) < \mathit{region}(\alpha, s)$ provided they are defined, and one of the following (A0)-(A4) holds:

(A0) $\mathit{region}(\alpha) \uparrow$.

(A1) $\mathit{region}(\alpha) \downarrow = n$ for some n, and one of $C_{n,-1}, C_{n,0}, \cdots, C_{n,n-1}$, or I_n has not yet received an assignment.

(A2) There is no computation in Δ^A_α which currently applies.

(A3) There is a computation in Δ^A_α which currently applies with use $\delta(\alpha, s)$, such that $\delta(\alpha, s) < r(\alpha, \mathit{threshold}(\alpha))[s]$.

(A4) There is a computation in Δ^A_α which currently applies with use $\delta(\alpha, s)$ and value $r = \Delta^A_\alpha(x(\alpha))[s]$, such that r has shown up in the trace $T^e_{x(\alpha,s)}$.

(A0)-(A4) can be thought of to be the state of the atomic strategy of α. When α has just been initialized, (A0) holds and we will pick a fresh value for $\mathit{region}(\alpha)$. Once that has been done, (A1) will now apply and we will have to wait for all the relevant parameters to be defined. α will next move on to (A2) and begin diagonalization with Δ^A_α. After that, α will move to either (A3) or (A4); if (A3) holds then the restraint on α from above has increased beyond $\delta(\alpha, s)$ - there is some high priority box blocking the strategy of α and we have to reset α. If (A4) holds then the opponent has responded by filling $T^e_{x(\alpha)}$ up. We can now turn the contents of $T^e_{x(\alpha)}$ into trash by enumerating $\delta(\alpha, s)$ into A.

Let α be a positive node, and $p, s \in \mathbb{N}$. We *make α promote all boxes with use p at stage s* by doing the following: for each k such that there is some $\beta \in Z^-(\alpha)$ and $k < l(\beta, s)$, such that $J^{A\oplus B}(k)[s] \downarrow$ with use $j^{A\oplus B}(k, s) > p$, we decrease $\mathit{size}^\beta_k[s]$ by 1. That is, this action adjusts the size of all boxes with use larger than p, because an enumeration of p into A or B is imminent.

3.4.5 The Construction

At each stage s of the construction, we will define the approximation to the true path of the construction, δ_s of length $< s$. We say that α *is visited* at stage s, and equivalently that s is an α-stage, if $\delta_s \supset \alpha$. The nodes along δ_s will get to act at stage s. At stage $s = 0$, initialize all nodes and set $\delta_s = \langle\rangle$.

Suppose $s > 0$, and assume that $\alpha = \delta_s \restriction d$ has been defined for $d < s$. We first consider the case where α is an $\mathcal{N}_e$-node. If

- $l(\alpha, s^-) < l(\alpha, s)$ where s^- is the previous α-stage, and
- $h_e(l(\alpha, s)) > C_{n,r}$ for every $n, r \in \mathbb{N}$ such that $C_{n,r} \downarrow$ and $n = \mathit{region}(\sigma)$ for some $\sigma \supseteq \alpha^\frown\infty$,

then we say that stage s is *α-expansionary*, otherwise it is non-α-expansionary. If stage s is α-expansionary, do the following: for all $k < l(\alpha, s)$ such that $J^A(k)[s] \downarrow$, we enumerate the value $J^A(k)[s]$ into V_k^α. For all $k < l(\alpha, s)$ such that size_k^α has not yet been assigned a value, we update the size and set $\mathit{size}_k^\alpha = h_e(k) - 1$. Set $\delta_s(d) = \infty$. On the other hand if stage s is non-α-expansionary, we set $\delta_s(d) = f$, and do nothing else.

Now assume that α is a $\mathcal{P}_e$-node. Let $\delta_s(d) = 0$. If α does not require positive attention, we do nothing. Otherwise, initialize all nodes $\beta \supset \alpha$, take the appropriate action listed below, and declare that α has *received attention* at stage s. Let x be the least such that (Ax) holds.

- $x = 0$: Pick a fresh number n for $\mathit{region}(\alpha)$, and set $x(\alpha) = n$. Also set $C_{n,-1} = n$ and $C_{n,0} = n + 3$. Set $\Psi^B(n)[s] \downarrow= n$ with fresh ψ-use. Furthermore if $\mathit{setter}(\alpha)$ is undefined, assign $\mathit{setter}(\alpha) = n$.
- $x = 1$: go down the following list, pick the first that applies, and perform the action stated.
  - If $\psi(x(\alpha), s) < r(\alpha, \mathit{setter}(\alpha))[s]$, do a self-imposed initialization of α.
  - If there is a smallest $q < n = \mathit{region}(\alpha)$ such that $C_{n,q} \uparrow$, set $C_{n,q} = C_{n,q-1} + 3 + n + nS(\alpha, C_{n,q-1})[s]$.
  - Otherwise if $C_{n,q} \downarrow$ for all $q < n$, set $I_n = S(\alpha, C_{n,n-1})[s]$. Adjust the size of boxes by making α promote all boxes with use $\psi(x(\alpha), s)$. Additionally, we put $\psi(x(\alpha), s)$ into B to clear the definition of $\Psi^B(x')$ for all $x' \geq x(\alpha)$. Set $\Psi^B(n + I_n)[s] \downarrow= n$ with fresh ψ-use.

- $x = 2$: we enumerate a computation $\Delta_{\alpha}^{A}(x(\alpha))[s]\downarrow = s$ with fresh use $\delta(\alpha, s)$.
- $x = 3$: reset α.
- $x = 4$: adjust the size of boxes by making α promote all boxes with use $\delta(\alpha, s)$. Enumerate $\delta(\alpha, s)$ into A, and increase *attempt*(α, s) by 1.

This concludes the definition of δ_s. Finally, we initialize all nodes $\beta >_{left} \alpha$, and proceed to the next stage. The true path of the construction is defined as usual to be the leftmost path visited infinitely often during the construction. If α is visited and receives attention under (A0) or (A1), we say that α *is setting its parameters.* That is, α is computing the initial values of various parameters it needs to start its atomic strategy. When α finishes setting its parameters, it will mark the end of the process with an enumeration into B, to adjust Ψ^B. Thereafter, α will begin diagonalization under (A2)-(A4), and never return to (A0)-(A1) again unless it is initialized.

3.4.6 Verification

The main bulk of the verification will focus on showing that various counting arguments work. Again we follow Theorem 2.3.1 closely. In the following lemma we are going to count the number of different strings ρ, such that J^ρ can be traced by a negative node. This affects the combinatorics used by the positive nodes. In (i) below, we show that the maximum number of different restraints that a current b-box can put up is at most $b+1$. Part (ii) says that the maximum number of different restraints held by any current b-box for $b < n$ of stronger σ-priority, is at most $S(\sigma, n)$. Because the part of the oracle accessed during computations can be different while possibly having the same length at different stages, we focus on strings ρ for J^ρ, rather than the length of the use $j^{A\oplus B}$.

Lemma 3.4.7. *(i) Let β be a negative node on the true path with true outcome ∞. Let $k \in \mathbb{N}$, and t_0 be a stage after which β is never initialized*[2]*, such that $size_k^\beta[t_0]\downarrow$. Then,*

$$|\{(A \oplus B)\restriction j(k)[s] : J^{A\oplus B}(k)[s]\downarrow \ \wedge \ \delta_s \supset \beta^\frown\infty \ \wedge \ s \geq t_0\}| \leq 1 + size_k^\beta[t_0].$$

[2]We have not yet shown that such a stage must exist. For this lemma we assume its existence.

(ii) Let σ be a positive node on the true path, $n \in \mathbb{N}$, and t_1 be a stage after which σ is never initialized. Then, the number of pairs $\langle (A \oplus B)\restriction j(k)[s], k\rangle$ for which

(a) $J^{A\oplus B}(k)[s]\downarrow$,

(b) for some $\beta \in Z^-(\sigma)$, we have $size_k^\beta[t_1] < n$,

(c) $\delta_s \supset \sigma$ and $s \geq t_1$,

is at most $S(\sigma, n)[t_1]$.

Proof. This is the same as Lemma 2.4.6, we include the proof here for the sake of completeness.

(i): this should be easy to see, because informally the statement says that if we currently have a b-box, then there can only be at most $b+1$ many possible observed versions ρ of the use for J^ρ, since we can have a different observed use only if the box is promoted.

Formally, we suppose for a contradiction, that there are stages $t_0 \leq s_0 < s_1 < \cdots < s_m$ such that $(A \oplus B)\restriction j(k)[s_i] \neq (A \oplus B)\restriction j(k)[s_{i+1}]$ for all $i = 0, \cdots, m-1$ (where $m = 1+size_k^\beta[t_0]$). For each i, the change $(A\oplus B)\restriction j(k)[s_i] \neq (A\oplus B)\restriction j(k)[s_{i+1}]$ must have been caused by some positive node $\beta' \supseteq \beta^\frown\infty$ receiving attention at some stage t where $s_i \leq t < s_{i+1}$. Hence β' must have promoted the box V_k^β and decreased $size_k^\beta$ while it is receiving attention at stage t. This means that by the time we reach stage s_{m-1}, we have $size_k^\beta = 0$. Hence at all stages $t \geq s_{m-1}$ including s_{m-1} itself, no node β' extending $\beta^\frown\infty$ is allowed to make an enumeration into A or B below $j(k, s_{m-1})$, since $r(\beta', y, t) \geq j(k, s_{m-1})$ for all $y \in \mathbb{N}$, a contradiction.

(ii): Using part (i). $\square$

Lemma 3.4.8. *Let σ be a positive node, receiving attention at some stage s, and let N be the largest parameter value of σ at stage s. Suppose $\beta \in Z^-(\sigma)$ and $k \in \mathbb{N}$. Then, the V_k^β-box cannot be promoted to become an N'-box after stage s (for all $N' \leq N$) by a node either extending σ, or to the right of σ.*

The statement of the lemma is rather long, but is really quite intuitive. What it says is the following. Take σ to be a positive node receiving attention at stage s, where it sets one of its parameter values $= N$ at stage s. Consider the V_k^β-boxes (these have higher priority than σ). It would be bad if a lot of new boxes of this type are promoted to have a small box size $\leq N$, since this messes up the combinatorics set up by σ. The lemma says that any such promotion cannot be due to nodes of lower priority than σ. Therefore, once σ asserts control during the construction, any such promotion which creates boxes of small size have to be due to σ's actions alone; this isolates the effects of the other positive nodes.

Proof of Lemma 3.4.8. Let σ' be a possible counterexample for the promotion of V_k^β. Since σ' is initialized at stage s, this means that $\mathit{setter}(\sigma')$ and $\mathit{threshold}(\sigma')$ must be larger than N whenever they are defined after stage s. This makes promotion of the type mentioned impossible, because $r(\sigma', N+1)$ will have to be obeyed. $\square$

Next, we will argue inductively that along the true path, all requirements are satisfied. We prove the following simultaneously by induction on $|\alpha|$, where α is on the true path:

(I1) if α is a $\mathcal{Q}$-node, then the statement of $\mathcal{Q}$ is satisfied,

(I2) if α is positive then it requires attention finitely often.

Suppose (I1) and (I2) holds for all $\beta \subset \alpha$, and α is on the true path. Firstly, suppose that α is an $\mathcal{N}_e$-node, and that h_e is an order. By induction hypothesis, α is initialized only finitely often, so we can consider the true version of $\{V_k^\alpha\}$. Fix a $k \in \mathbb{N}$. If $J^A(k)\downarrow$, then clearly it will be enumerated into V_k^α at a large enough α-expansionary stage. Each distinct value in V_k^α corresponds to a string $(A \oplus B)\upharpoonright j(k)[s]$ (i.e. the use) in the statement of Lemma 3.4.7(i), with $t_0 =$ least stage after which α is never initialized. Hence, it follows that $|V_k^\alpha| \leq h_e(k)$.

Now suppose that α is a $\mathcal{P}_e$-node. Let s_0 be the least α-stage after which α is never initialized, and $q = \mathit{setter}(\alpha, s_0)$ be the final value.

Lemma 3.4.9. *There are only finitely many self-imposed initializations to α.*

Proof. Note that once α finishes setting its parameters after s_0, then (A0) and (A1) will never apply to α again, so the lemma is clearly true. So, we may assume for the sake of argument, that α never finishes setting its parameters, hence never enumerates anything (into B). We argue that $r(\alpha, q)$ eventually settles, for a contradiction. We first claim that for each $\beta \in Z^-(\alpha)$, and each $p > l(\beta, s_0^+)$ such that s_0^+ is a stage after s_0 with $h_\beta(l(\beta, s_0^+)) > q+1$, we have $size_p^\beta > q$ whenever it is defined: when $size_p^\beta$ first gets defined, it is certainly larger than q. If the V_p^β box is to be promoted after stage s_0^+, it has to be due to some positive node σ. Clearly σ cannot be on top or to the left of α, lest α is initialized. Also $\sigma \neq \alpha$ by assumption. Thus we have $\sigma \supset \alpha$ or $\sigma >_{left} \alpha$. Applying Lemma 3.4.8, it follows that $size_p^\beta > q$ always holds. Hence only finitely many boxes contribute to $r(\alpha, q)$, and by Lemma 3.4.7(ii) this means that $r(\alpha, q)$ is bounded, a contradiction. $\square$

Hence, α will eventually finish setting its parameters, with the final value for $n = \lim region(\alpha)$. By Lemma 3.4.8 boxes can be promoted to small sizes only by α itself. We now argue that amongst the different threshold values $C_{n,-1}, C_{n,0}, \cdots$, box sizes are also kept disjoint. That is, once $C_{n,r}[t_0]$ has been defined, then no box of a current size at least $C_{n,r}[t_0]$ can be promoted to size $C_{n,r-1}[t_0]$.

Lemma 3.4.10. *(i) For each $0 \leq r \leq n-1$, $\beta \in Z^-(\alpha)$, each stage t_0 such that $C_{n,r}[t_0] \downarrow$, and k such that $size_k^\beta[t_0] \downarrow \geq C_{n,r}$, we have $\forall t(t \geq t_0 \Rightarrow size_k^\beta[t] > C_{n,r-1})$.*

(ii) The total number of times which α can be reset after it sets its parameters, is bounded by I_n.

Proof. (i): This follows Lemma 2.4.7 closely. We proceed by induction on r. Suppose the results hold for all $r' < r$. Suppose the statement fails for some $\beta \in Z^-(\alpha)$, t_0 and k. Let $s > t_0$ such that $size_k^\beta[s] \leq C_{n,r-1}$, and we may as well assume that $size_k^\beta[t_0] = C_{n,r}$. Since $C_{n,r}[t_0] \downarrow$, it follows there is some α-stage $\bar{t_0} \leq t_0$, such that $C_{n,r}$ receives its definition.

Suppose $t \geq t_0$ is a stage where some σ promotes V_k^β. It is clear from Lemma 3.4.8 that σ has to be α. We want to count the number of such stages t where α promotes V_k^β. At stage t, promotion happens because α needs to enumerate into A or B. The latter case only happens once after stage s_0. In order for promotion to take place at stage t due to enumeration into A, we must have $C_{n,r} \geq size_k^\beta[t] > threshold(\alpha, t) = C_{n,n-2-attempt(\alpha,t)}$, which means that $attempt(\alpha, t) > n-2-r$. We split the counting into the possible cases z for $n-1-r \leq z \leq n-1$:

Case z: t is a stage where α promotes V_k^β, and $attempt(\alpha, t) = z$.

If $z = n-1$, then *attempt*(α) will be increased to n and α never enumerates again. So, suppose that $z < n-1$ (and hence $r > 0$). In order for Case z to apply again, α has to be reset at some (least) α-stage $t' > t$, where $attempt(\alpha, t') \geq z+1 > n-1-r$. This must be due to some small box size blocking α. That is for some $\beta' \in Z^-(\alpha)$ and k', we have $J^{A\oplus B}(k')[t'] \downarrow$ and $size^{\beta'}_{k'}[t'] \leq threshold(\alpha, t') \leq C_{n,r-2}$.

We claim that $size^{\beta'}_{k'}[\bar{t_0}] \downarrow$. Suppose not. Since $r > 0$, it follows that $h_{\beta'}(l(\beta', \bar{t_0})) > C_{n,r-1}$ being a β'-expansionary stage, and therefore $h_{\beta'}(k') > C_{n,r-1}$. Applying induction hypothesis (on $r-1$) gives us a contradiction. The above not only shows that at stage $\bar{t_0}$, $size^{\beta'}_{k'}[\bar{t_0}]$ must be defined, but in fact that $size^{\beta'}_{k'}[\bar{t_0}] < C_{n,r-1}$. Applying Lemma 3.4.7(ii), there can be at most $1 + S(\alpha, C_{n,r-1})[\bar{t_0}]$ many stages t where Case z applies (by associating each t with the string $(A \oplus B)\upharpoonright j(k')[t']$).

Totalling the effects from all the different cases, we see that the smallest value $size^{\beta}_{k}$ can take, is $C_{n,r} - 1 - 1 > C_{n,r-1}$, if $r = 0$. On the other hand if $r > 0$, the smallest value $size^{\beta}_{k}$ can be reduced to, is $C_{n,r} - 1 - n(1 + S(\alpha, C_{n,r-1})[\bar{t_0}]) - 1 = C_{n,r-1} + 1 > C_{n,r-1}$.

(ii): this follows by a similar counting argument as (i). □

By Lemma 3.4.10(ii), α never runs out of indices after it is done setting parameters, so let $x = \lim x(\alpha)$. After a large enough amount of time has passed, the only reason why α requires attention will be under (A2) or (A4). When *attempt*(α) reaches n, α will no longer require attention, so (I2) is true. We now show (I1) holds. Clearly $\Psi^B(x) \downarrow = n$. Suppose that $J^A(x) \downarrow = w$ and $w \in T^e_x$. Let $s_1 > s_0$ be large enough so that $w \in T^e_x[s_1]$, and that $\Delta^A_\alpha(x)[s_1] \downarrow = w$. Observe that it must be the case that $attempt(\alpha, s_1) = n$, otherwise we would destroy the correct axiom in Δ^A_α at the next α-stage. Hence there are n many different stages $t \leq s_1$ and $attempt(\alpha, t)$ is increased by 1. After each such stage t we also enumerate a new value for $\Delta^A_\alpha(x)$, and wait for it to be traced. Thus we have $|T^e_x| \geq n$, since each time α sees a new value appearing in T^e_x, it initializes all nodes of lower priority. So, (I1) is true. This completes the induction.

The last thing we have to do, is to argue that Ψ^B is a B-order, in the sense that it is total, nondecreasing and unbounded. This follows from the fact that for every positive node α on the true path, we have $\Psi^B(\lim x(\alpha))\downarrow = \lim region(\alpha)$. This ends the proof of Theorem 3.4.5.

3.5 Strong jump traceability with slow growing orders

Theorem 3.4.5 raises a fundamental issue. Can there possibly be (noncomputable) sets A which has such low computational strength, such that given *any* set X not computing A, we are able to make A strongly jump traceable by X? As we have seen, it is generally *harder* to jump trace a set A by a noncomputable oracle X, compared to simply jump tracing A with no oracle, because we have to deal with more (slow-growing) X-orders.

Theorem 3.4.5 shows that such behaviour has to be at least distinct from strong jump traceability. To what extremes can this concept be taken? Is it reasonable to require that A is strongly jump traceable by *all* sets?

Definition 3.5.1. We say that a c.e. set A is *hyper jump traceable*, if A is strongly jump traceable by every c.e. set W.

That is, we call the strong notion $\mathcal{H}$ corresponding to the partial relativization $SJT(X)$, the hyper jump traceable c.e. sets. This defines a class with very strong similarities to $\emptyset$: not only can we approximate J^A with respect to all computable order functions, but also for all c.e. sets W, we can W-approximate J^A with respect to all W-computable order functions.

The rest of this chapter will be devoted to studying $\mathcal{H}$. In this section, we give a separate motivation for studying the hyper jump traceable. In Theorem 3.6.1, we first give a sketch of the construction of a noncomputable c.e. set which is hyper jump traceable. Hence $\mathcal{H}$ is a new nontrivial subclass of the strongly jump traceable and K-trivial sets. We then construct a hyper jump traceable set which is cuppable. Thus, whilst the c.e. hyper jump traceable sets resemble $\emptyset$ very closely, there is at least a fundamental property which separates them from the computable sets. In Theorem 3.7.1 we show that no construction of a c.e hyper jump traceable set is compatible with making it low cuppable. In Section 3.8, we show that no c.e. set A can be strongly jump traceable relative to *all* Δ_2^0 sets, apart from the computable sets. In Section 3.9 we show that there is a single capping companion for the entire class $\mathcal{H}$.

3.5.1 A different approach for strengthening strong jump traceability

The above Definition 3.5.1 might strike the reader to be unnatural. After all, if we wanted to strengthen the notion of traceability, we might not at first think of the approach by relativization. Arguably, a more natural approach towards strengthening strong jump traceability would be to first consider traceability with respect to a larger (and yet natural) class of order functions. Instead of considering only computable orders, we would consider the next level of (non)-effectivity; these are those functions with computable approximations allowing a controlled number of mind changes.

We formalize this idea. For $\alpha \leq \omega$ we say that a total function h is an α-order, if it is nondecreasing unbounded and $h^{-1}(n)$ is α-c.e. Here we define $h^{-1}(n) =$ least x such that $h(x) \geq n$. For example the computable order functions are exactly the 1-order functions. We can define strong jump traceablility as usual with respect to these classes of slow growing order functions. Specifically we say that A is *α-sjt* if for every α-order h, there is a c.e. trace obeying h which traces J^A. Therefore, for the α-orders, we are allowed to change our mind on the output values; if a set is able to be traced by these functions, then it should be computationally even more feeble. We show however, that these preliminary efforts to strengthen strong jump traceability give nothing new.

Lemma 3.5.2. *For each $n < \omega$, every n-order dominates a computable order.*

Proof. Let h be a proper n-order, in the sense that for infinitely many m, $h^{-1}(m)$ moves exactly n times. We build a computable order $\tilde{h}$ such that $h(x) \geq \tilde{h}(x)$ for every x. We are given an n-c.e. approximation $h^{-1}(x)[s]$. We write $ch(m, s) = |\{t < s : h^{-1}(m)[t] \neq h^{-1}(m)[t+1]\}|$, the number of mind changes to the approximation of $h^{-1}(m)$ up to s. Define the functions $P(z), t(z)$ as follow. Let $t(z)$ be the least

$t > t(z-1)$ for which there is some $m > P(z-1)$ such that $ch(m,t) = n$. Let $P(z)$ be the least such m.

The idea is that $P(0)$ is the first argument (in chronological order) m which attains a mind change of $n+1$. $P(1)$ the first argument (again with respect to time) larger than $P(0)$ with this property. Note that P is a total computable function, in fact it is strictly increasing. Define $\tilde{h}(y)$ by the following. Search for the largest z such that $h^{-1}(P(z))[t(z)] \leq y$ and let $\tilde{h}(y) = P(z)$.

Note that in order for h to be an order, it must be that for every m, $\lim_s h^{-1}(m)[s] \leq \lim_s h^{-1}(m+1)[s]$. It is easy to verify that $\tilde{h}$ is a computable order, since the function $\lambda z(h^{-1}(P(z))[t(z)])$ is computable, nondecreasing and unbounded. We argue that for (almost) every y, $\tilde{h}(y) \leq h(y)$. By the definition of $\tilde{h}(y) = P(z)$, we had $y \geq Q$ where $Q = h^{-1}(P(z))[t(z)]$ for the largest z satisfying the equation. Note that in fact $Q = \lim_s h^{-1}(P(z))[s]$ so that $h(Q) \geq P(z)$. We have $h(y) \geq h(Q) \geq P(z) = \tilde{h}(y)$. □

Theorem 3.5.3. *For each $n < \omega$, we have A is strongly jump traceable $\Leftrightarrow$ A is n-sjt. If A is ω-sjt, then A is computable.*

Proof. The first statement is a direct corollary of Lemma 3.5.2. For the second statement we construct an ω-order h meeting all the requirements

$$\mathcal{R}_e: \text{ either } |T_x^e| \leq e \text{ for almost every } x, \text{ or there is some } x \text{ such that } |T_x^e| > e \text{ and } h(x) \leq e.$$

Here $\{T_x^e\}_{x\in\mathbb{N}}$ is the e^{th} c.e. trace. This is enough to prove the theorem, since if A can be traced by a c.e. trace with a constant bound, then A is computable. Note that since every c.e. trace has infinitely many indices, the requirements also imply that for any c.e. trace T_x with unbounded size, there are infinitely many x such that $|T_x| > h(x)$.

We proceed by a straightforward construction. We give an ω-c.e. approximation $g(n,s)$ where $\lim_s g(n,s)$ is strictly increasing and unbounded. We let $h(y) = n$ where n is the largest such that $\lim_s g(n,s) \leq y$.

Construction of $g(n,s)$: At the beginning set $g(n,0) = n$ for every n. At stage s look for the least $e < s$ which has not yet received attention, and such that there is some $x < s$ with $|T_x^e[s]| > e$ and $g(e,s) < x$. Pick the least such x, and define $g(e', s+1) = g(e', s)$ if $e' < e$ and $g(e', s+1) = s + (e' - e)$ if $e' \geq e$. Also declare that e has received attention. It is clear that for every e, s, we have $g(e,s) \leq g(e, s+1)$ and $g(e,s) < g(e+1, s)$. We also have $h^{-1}(n) = \lim_s g(n,s)$ for every n, with at most n many mind changes. It is also easy to verify that g and the resulting h have the required properties, and satisfy all requirements $\mathcal{R}_e$. □

The above shows that if we want to strengthen strong jump traceability by considering slow growing orders, we cannot expect the trace to remain computably enumerable. A natural way to try and salvage this notion would be to require the traces to be for instance uniformly ω-c.e. or even $\omega + 1$-c.e., when considering functions h where h^{-1} is ω-c.e.. However the problem with this approach is that we need the complexity of the trace to depend on the complexity of the order function which it has to obey. For instance we could change the definition of ω-sjt to read: "for each ω-order function h, there is a uniformly ω-c.e. trace obeying h for J^A". The trouble with this is that we allow non-c.e. traces even for the computable orders, and so this notion does not strengthen strong jump traceability.

A second requirement which we expect any strengthening of sjt to obey is that in the construction of the trace, one has to assume complete knowledge of the order function which is to be obeyed by the trace. The construction in Theorem 3.5.3 is in some sense cheating because it uses the fact that the c.e. traces had no knowledge of the ω-order being constructed, and naturally cannot be expected to obey it (and yet be unbounded). In view of these two considerations, we make the following definition.

Definition 3.5.4. If $\mathcal{C}$ is a class of total, nondecreasing and unbounded functions from $\mathbb{N} \mapsto \mathbb{N}$, and A is a real, then we say that *A is $\mathcal{C}$-strongly jump traceable* if for every $h \in \mathcal{C}$, there is a computable function g such that $\{W_{g(x)}^h\}_{x \in \mathbb{N}}$ traces J^A obeying h.

That is, we require the traces to be computably enumerable relative to the order it is expected to obey. We show that the hyper jump traceable sets can now be characterized naturally in this way.

Theorem 3.5.5. *A is hyper jump traceable iff A is $\mathcal{C}$-sjt where $\mathcal{C}$ = all functions of c.e. degree.*

Proof. We prove the nontrivial direction. Suppose A is $\mathcal{C}$-sjt, and V is a c.e. set and Φ^V is an order. Note that Φ^V can have very low Turing degree, and may even be computable. It is easy to define an order $h \equiv_T V$ such that $h(x) \leq \Phi^V(x)$ for every x. Let $p = (\Phi^V)^{-1}$. We define h^{-1}, from this it is straightforward to recover h. Let $h^{-1}(n) = h^{-1}(n-1) + p(n)^{s_n}$ where s_n is the least stage such that n is enumerated in V, and $s_n = 1$ if $n \notin V$. It is easy to see that $h^{-1}(n) \geq p(n)$ for every n, and that $h \leq_T V$. To see that $h \geq_T V$, observe that for almost every n, $h^{-1}(n) \geq s_n$, so that $n \in V \Leftrightarrow n \in V[h^{-1}(n)]$. □

We show that if we look at the class of Δ^0_2-functions, then it is too large a class to have any meaning amongst the c.e. sets.

Proposition 3.5.6. *$A \in \cap\{SJT(V) \mid V \leq_T \emptyset'\}$ iff A is $\{h \mid h \leq_T \emptyset'\}$-sjt. Consequently the only c.e. sets which are $\{h \mid h \leq_T \emptyset'\}$-sjt are the computable sets.*

Proof. The first statement is trivial. The second statement follows Theorem 3.8.1. □

Question 3.5.7. *Can hyper jump traceability be characterized in terms of a diamond class, or in terms of randomness?*

More generally we ask if $\mathcal{C}$-sjt for different sub-classes of the Δ^0_2 functions are equivalent (or at least implies) any of the "strong diamond classes". Here a strong diamond class refers to a subclass of the strongly jump traceable sets obtained by the diamond operator. For instance one could look at $\{\omega + 1\text{-c.e.}\}^\diamond$ or even $\{\omega^2\text{-c.e.}\}^\diamond$. Note that $\{\omega^2\text{-c.e.}\}$ is a Σ^0_3 class, so the associated diamond class is non-empty.

3.6 A cuppable hyper jump traceable c.e. set

Theorem 3.6.1. *There is a c.e. hyper jump traceable set $A >_T \emptyset$.*

Sketch of the construction. There are two categories of requirements to be considered here. We have the positive requirements making A noncomputable, as well as the negative requirements of the form

$$\mathcal{N}: \text{ If } h^W \text{ is an order, build a trace } \{T_x^W\}_{x\in\mathbb{N}} \text{ for } J^A \text{ respecting } h^W.$$

The standard strategy for building a noncomputable strongly jump traceable set (without considering oracles) is the following. Suppose the opponent shows us $h^{\emptyset}(x)[s]\downarrow= 2$ for some x. We could then safely trace $J^A(x)[s]$ into T_x (the trace we build), and then impose the A-restraint $j(x,s)$. This is alright because in future, we can still allow one positive requirement to act below the restraint $j(x,s)$, so this restraint is not absolute. The main trouble now is that h^W might not be computable; the opponent has much more freedom now in the sense that he could show us an (incorrect) segment $h^W\upharpoonright n$, and then later on changes his mind on that segment. In particular, we might first be shown $h^W(x)[s]\downarrow= 2$ for some x, and when we decide to trace $J^A(x)[s]$, the opponent would then tell us that in fact, $h^W(x) = 1$. In that case, the A-restraint $j(x,s)$ now becomes of the highest priority, and no positive action below it is allowed. If the opponent tricks us infinitely often in this way, we would not be able to get anything into A.

However, just like the opponent, we are also gifted with some degree of freedom to change our mind on whatever we are building. We could "wrap" our trace T_x^W axioms around the opponent's axioms for h. In particular, when the opponent first shows us $h^W(x)[s]\downarrow= 2$, we would enumerate $J^A(x)[s]$ into T_x^W with W-use $u(x)[s] =$ use of $h^W(x)[s]$. When he changes his mind and later on shows us $h^W(x)[s]\downarrow= 1$, he has to change $W\upharpoonright u(x)[s]$; the same W-change would also remove the value $J^A(x)[s]$ from our trace T_x^W. This allows us to open up a gap by dropping A-restraint momentarily before tracing $J^A(x)[t]$ the next time. When the opponent next shows us $h^W\upharpoonright n+1[s'] = 1^n2$ (for some $n > x$), we would set our T_x^W-use to be the same as $u(n)[s']$. In this way, if the opponent tries to show us initial segments of h^W of the form 1^m2 for infinitely many m's, we would also have ensured that infinitely many gaps are open, in which positive requirements can act.

The above obstacle can be turned into a construction of a noncomputable order g such that no c.e. set (other than the computable sets) can be jump traced respecting g using only a plain sequence $\{T_x\}$. Therefore if we consider jump traceability via noncomputable orders, we will need a suitable oracle to help us build the trace.

The above describes a basic module of $\mathcal{N}$. The requirement $\mathcal{N}$ now has to be split into infinitely many subrequirements, and each subrequirement will run several of these basic modules. For each y (corresponding to a basic $\mathcal{N}$-module), we need to guess whether or not there are infinitely many n's such that the opponent shows us an initial h^W-segment of the form $\sigma y^n(y+1)$ for some string σ. Here y^n denotes y repeated n times. The infinitary outcome of a subrequirement corresponds to infinitely many A-gaps open (i.e. $\liminf$ of the restraint is 0), while the finitary outcome means that the restraint on A eventually settles down.

We can arrange these requirements on a tree of strategies, in the style of a $\emptyset'''$-priority argument. This is due to Lachlan [Lac75] in his monster paper, and is also presented in Chapter XIV.4 of [Soa87]. The reader is assumed to be familiar with standard tree arguments, and a good exposition on this topic can be found in [Soa85]. Deciphering the outcome of an $\mathcal{N}$-requirement requires a $\emptyset'''$-oracle, because "h^W is an order" is a Π^0_3 fact, as discussed above, and our $\mathcal{N}$-strategies depend crucially on this fact. The full construction is given in Theorem 3.6.2.

On a side note we remark that this strategy will meet with fatal problems if W is a Δ^0_2 set. This is because the opponent could challenge us as above, and wait for us to enumerate $J^A(x)[s]$ into T^W_x at stage s. Suppose the next thing he does is to change W, hence emptying T^W_x for us, and we would follow up by opening up an A-gap below $j(x,s)$, according to the plan. At the next stage the opponent could *recover* W back to the configuration at stage s, i.e. restore W back to the state before the A-gap was opened. We would be in trouble, because T^W_x now contains the $J^A(x)[s]$ (having been brought back by the opponent), and the opponent could now make $J^A(x) \neq J^A(x)[s]$. Since A is c.e. there is no way for us to get the correct $J^A(x)$ into our trace, unless we increase the size of T^W_x illegally. In Section 3.8, we show that this obstacle cannot be overcome - no c.e. set is strongly jump traceable by every Δ^0_2 set. It is not known however, if we can build an $A \leq_T \emptyset'$, such that A is strongly jump traceable by every Δ^0_2 set. □

Theorem 3.6.2. *There is a c.e hyper jump traceable set A, and an incomplete c.e. set C, such that $\emptyset' \leq_T A \oplus C$.*

3.6.1 Requirements

We code $\emptyset'$ into $A \oplus C$ by building the reduction $\emptyset' = \Gamma^{A\oplus C}$, and act for it globally, i.e. outside of the construction tree. There are two types of requirements with conflicting interests, namely the ones making A hyper jump traceable:

$$\mathcal{N}_e \quad : \quad \text{If } h_e^{W_e} \text{ is an order, then build a trace } \{T_x^{W_e}\}_{x\in\mathbb{N}} \text{ for } J^A \text{ respecting } h_e^{W_e},$$

and the ones making C incomplete, by building an auxiliary c.e. set D:

$$\mathcal{R}_e \quad : \quad \Phi_e^C \neq D.$$

Here, $\langle W_e, \Phi_e, h_e\rangle_{e\in\mathbb{N}}$ is an effective list of all triples such that W_e is a c.e. set, and Φ_e, h_e are Turing functionals. Also let $J^A(x) = \{x\}^A(x)$. The stage s use of the computations $J^A(x), \Phi_e^C(x)$ and $h_e^{W_e}(x)$ are denoted by $j(x,s), \varphi_e(x,s)$ and $u_e(x,s)$ respectively.

Since we are concerned only with the functions $h_e^{W_e}$ that are total, it does no harm for us to assume that for $x < y$, we have $u_e(x,s) < u_e(y,s)$ whenever they are both defined. When we say that we pick a *fresh* number x at stage s, we mean that we choose x to be the least number $x > s$, and $x >$ any number used or mentioned so far. We will also drop the stage number from the notations if the context is clear. All parameters will retain their assigned values until initialized or reassigned. We append $[s]$ to an expression to mean the value of the expression as evaluated at stage s.

At each requirement $\mathcal{N}_e$, we will build a u.c.e. trace $\{T_x^{W_e}\}_{x\in\mathbb{N}}$ such that for all $x \in \mathbb{N}$, we have (if $h_e^{W_e}$ is an order)

(i) $|T_x^{W_e}| \leq h_e^{W_e}(x)$, and

(ii) $J^A(x)\downarrow \Rightarrow J^A(x) \in T_x^{W_e}$.

As discussed in Theorem 3.6.1, we divide the requirement $\mathcal{N}_e$ into the subrequirements $\mathcal{N}_{e,0}, \mathcal{N}_{e,1}, \cdots$, where $\mathcal{N}_{e,i}$ is responsible for tracing $J^A(x)$ for all x such that $m_e^{i+1} < h_e^{W_e}(x) \leq m_e^{i+2}$, where $m_e^i = 2^{2\langle e,i\rangle+2}$. This number is chosen based on technical reasons, which will become clear later.

3.6.2 Description of strategy

In this construction we replace the noncomputable requirements with the (stronger) cupping requirements. This consists of two parts - the coding of $\emptyset'$ into $A \oplus C$, as well as the incompleteness strategies $\mathcal{R}$. The coding of the Turing complete set is done outside the tree, and is not affected by the movement of the accessible string δ_s (the accessible string determines which nodes get to act during stage s of the construction). This involves maintaining a set of markers $\gamma(0) < \gamma(1) < \cdots$, where the current values of $\gamma(n), \gamma(n+1), \cdots$ are dumped into the set C whenever some n enters $\emptyset'$. New values will be chosen for these markers, and we say that these markers are *moved*. More details will be given in Section 3.6.4; at this stage the reader will only need to note that the coding strategy can only move markers by dumping them into C.

The second part to cupping involves building a set D not computable in C. The atomic strategy for an $\mathcal{R}$-requirement is the standard Friedberg strategy: it picks a number z not yet in D, and waits for $\Phi^C(z)[s] \downarrow = 0$. When that happens, we enumerate z into D and preserve C; at this point we say that we *believe in the Φ^C-computation.* In order for the strategy to succeed, we have to ensure that $\mathcal{R}$ believes in only finitely many wrong Φ^C-computations. To limit the number of times a believed computation may be injured and hence become incorrect, $\mathcal{R}$ will first pick a number b, and then only believe in a Φ^C-computation if $\varphi(z, s) < \gamma(b, s)$; to do this it may have to enumerate $\gamma(b, s)$ into A to move the marker $\gamma(b)$ above $\varphi(z, s)$.

Let us consider an $\mathcal{R}$-strategy below a subrequirement $\mathcal{N}_{e,0}$ of $\mathcal{N}_e$. Suppose that $\mathcal{N}_{e,0}$ is assigned to the node α, and is responsible for tracing $J^A(x)$ for all the x such that $h_e^{W_e}(x) = 1$. The locations in $\{T_x^{W_e}\}$ which trace these values are called 1-*boxes*, and once these 1-boxes are filled with a value, the restraint $\mathcal{N}_{e,0}$ imposes for it has to be respected everywhere. At all times $\mathcal{N}_{e,0}$ is in charge of a certain number of 1-boxes; however since $h_e^{W_e}$ may not be computable, the number of 1-boxes it is responsible for may increase with time. Recall that $\mathcal{N}_{e,0}$ will have two outcomes, the infinitary ∞ outcome which opens a gap each time $\mathcal{N}_{e,0}$ is entrusted with more 1-boxes to trace, and the finitary outcome f which $\mathcal{N}_{e,0}$ will take if the situation in $h_e^{W_e}$ is stable.

There will be two corresponding versions of the $\mathcal{R}$ strategy assigned to the nodes

$\beta_0 = \alpha^\frown\infty$ and $\beta_1 = \alpha^\frown f$ below α. These two use a different set of parameters, i.e. b_{β_0}, z_{β_0} for β_0 and b_{β_1}, z_{β_1} for β_1. How may β_1 be affected by α? Each time α increases A-restraint due to activity in one of its 1-boxes (remember these boxes are of the highest priority), we would have to move the marker $\gamma(b_{\beta_1})$ above the A-restraint, by dumping into C. However β_1 might encounter infinitely many 1-box restraints, but if that is the case then β_0 will be visited infinitely often, and hence be the one to meet $\mathcal{R}$. To ensure that $\gamma(b_{\beta_1})$ settles, we have to pick a new value for b_{β_1} each time α opens a gap and is shown with even more 1-boxes to trace. Hence, β_1 (or β_0, depending on the true outcome of α) will eventually settle on some final value for b_{β_1}, and receive a finite amount of restraint from α above (in the case of β_0 there is no restraint from above). We want to see that it succeeds in meeting $\mathcal{R}$ with some z.

It is possible for β_1 to believe in a false computation, if some marker $\gamma(k)$ for $k < b_{\beta_1}$ is dumped into C after we believe. Thus, it is possible for β_1 to make more than one enumeration into A. Each A-enumeration that β_1 makes has to respect the restraint from 1-boxes above it (i.e. from α), but ignores the restraints imposed by boxes below it. In particular, $\mathcal{N}_{e,1}$ might occupy a level below β_1, and be in charge of a number of 2-boxes, which all get *promoted*[3] to 1-boxes due to the positive actions of β_1. When these locations next get traced, *their restraints have to be obeyed even by β_1 above*, because $\mathcal{N}_{e,1}$ while of a lower local priority, actually has a higher global priority than β_1. Therefore, the blockages on A that β_1 has to consider become slightly more complicated; in particular β_1 has to search through the entire sequence $\{T_x^{W_e}\}$ to see which boxes have been promoted and must now be obeyed.

This brings up another problem. Notice that there might be many levels below β_1, which are assigned some other incompleteness requirement $\mathcal{R}'$, say at some $\sigma \supset \beta_1$. When σ acts, it may in turn promote $\{T_x^{W_e}\}$-boxes living *below* it, causing these boxes to become 1-boxes. Hence, infinitely many 1-boxes may be promoted in this way down the tree; β_1 now has to deal with all these boxes, which is bad because β_1 is on the true path and there is no other backup strategy for $\mathcal{R}$.

Before we proceed further, let us pause for a moment and think why this does not

[3]That is, a location $T_y^{W_e}$ with $h_e^{W_e}(y) = 2$ is now filled with a single wrong value, reducing the size of the tracing location.

happen in Theorem 3.6.1. The positive requirements were simple - each level on the tree assigned a positive requirement only needed to make at most one enumeration into A, for the sake of making A noncomputable. Therefore, we could use this fact to help us assign the $\mathcal{N}_{e,i}$-strategies on the tree. For instance, we could do the following assignment of strategies in Theorem 3.6.1:

level of the construction tree	strategy assigned
0	handle 1-boxes
1	noncomputable strategy
2	handle 2-boxes
3	noncomputable strategy
4	handle 3-boxes

For instance, the 3-boxes can be handled safely at level 4 because each level (levels 1 and 3) above it running the positive strategy can make at most one enumeration each. Hence the 3-boxes are at most promoted to a 1-box, after which levels 1 and 3 no longer need to act.

We can use this idea to help us out in this current construction. Even though the nodes assigned the incompleteness strategies enumerate more than once into A, we can make them behave just like a noncomputable strategy relative to each $\mathcal{N}_{e,i}$. To illustrate this, we arrange the strategies in this manner:

level of the construction tree	strategy assigned
0	handle 1-boxes
1	incompleteness strategy
2	handle 2-boxes
3	incompleteness strategy
4	handle 3-boxes

If the incompleteness strategy at level 3 enumerates into A, promoting boxes at level 4 in its current run, we make sure that at its next run when it has to enumerate into A again, it will not promote any boxes at level 4 which it has promoted before. Hence levels 1 and 3 will look like they are running a noncomputable strategy when viewed by level 4, even though they are making many enumerations into A. Thus, level 3 can promote boxes at levels $4, 6, 8$,etc but this has no effect on level 1.

To help us implement this idea, for each node σ assigned to an incompleteness strategy, we keep track of the *injury number* of σ. This represents the largest number x such that σ has previously injured a box location $T_x^{W_e}$, and helps σ remember which boxes not to promote in its next run. Note that the injury number does not need to increase until σ makes the next enumeration into A, hence lifting $\gamma(b)$ above the φ-use, and believing the Φ^C-computation and ending its current run. Therefore, in its current run, there is only finitely much restraint it has to obey from below (due to those boxes it has previously promoted), before it is next allowed to believe in a Φ^C-computation. Note also that in this construction, we are exploiting the fact that C need not be low (in fact, cannot be low): we dump markers into C until all blockages on A have been overcome.

3.6.3 Construction tree layout

The construction takes place on a subtree of the full binary tree. Nodes of even length $|\alpha| = 2e$ are assigned the requirement $\mathcal{R}_e$ with the single outcome 0. Nodes of length $|\alpha| = 2\langle e, i\rangle + 1$ are assigned the requirement $\mathcal{N}_e$ if $i = 0$, and the subrequirement $\mathcal{N}_{e,i-1}$ if $i > 0$. Nodes assigned the requirement $\mathcal{N}_e$ have only one outcome 0. The subrequirement $\mathcal{N}_{e,i}$ of $\mathcal{N}_e$ has the two outcomes $\infty <_{left} f$. The infinitary outcome represent phases when $\mathcal{N}_{e,i}$ is imposing no restraint (during A-gaps), while the finitary outcome means that $\mathcal{N}_{e,i}$ is holding some A-restraint.

Let $\alpha <_{left} \beta$ denote that α is strictly to the left of β (i.e. there is some $i < \min\{|\alpha|, |\beta|\}$ such that $\alpha\restriction i = \beta\restriction i$ and $\alpha(i) <_{left} \beta(i)$). We say that α *is a $\mathcal{Q}$-node*, if α is assigned the requirement $\mathcal{Q}$. α is an *incompleteness node*, if α is an $\mathcal{R}_e$-node for some e. We say that α is a *top node*, if α is an $\mathcal{N}_e$-node for some e, and that α is an *i-bottom node of* τ if α is an $\mathcal{N}_{e,i}$-node for some e, and $\tau \subset \alpha$ where τ is an $\mathcal{N}_e$-node. If α is a bottom node, then $\tau(\alpha)$ denotes its top node. α and β are *sibling nodes* if for some i, they are both i-bottom nodes of the same top. If τ is an $\mathcal{N}_e$-node, let $Cone_i^\tau := \{\beta : \beta \supset \tau \ \wedge \ |\beta| < 2\langle e, i+1\rangle\}$, that is, all the nodes lying strictly in between τ and its i-bottom nodes.

Each top $\mathcal{N}_e$-node τ does not really make any guesses as to whether or not $h_e^{W_e}$ is an order. Rather, this is done at each layer below τ. We distribute the A-restraints amongst its bottom nodes, and τ's role is to coordinate the actions of its bottom nodes.

3.6.4 Notations

The functional Γ reducing $\emptyset'$ from $A \oplus C$ will be built globally. If a $\Gamma^{A\oplus C}(x)$ axiom is set at stage s, it will have use denoted by $2\gamma(x,s)$, so that the first $\gamma(x,s)$ many bits of both A and C are involved. We ensure that at all times, if $x < y$ and both axioms are set, then $\gamma(x,s) < \gamma(y,s)$. To ensure this, we always enumerate markers by dumping. That is, each time during the construction when we say that we enumerate $\gamma(x,s)$ into A or C, we mean to say that we enumerate $\gamma(y,s)$ into A or C, for every $y \geq x$ for which the Γ-axioms have been set. We fix an enumeration $\{\emptyset'_s\}_{s\in\mathbb{N}}$ of the halting problem, in which at most one element is enumerated at each stage.

If σ is an $\mathcal{R}_e$-node, it will pick a number z_σ and attempt to make $\Phi_e^C(z_\sigma) \neq D(z_\sigma)$. It does this by picking a number b_σ, and then it will enumerate z_σ into D at a stage s only after it succeeds in ensuring that $\gamma(b_\sigma, s) > \varphi_e(z_\sigma, s)$. At each top $\mathcal{N}_e$-node τ, we measure the length of convergence of $h_e^{W_e}$ at stage s by

$$l_\tau(s) = \max\{y < s \mid (\forall x \leq y)\ h_e^{W_e}(x)[s] \downarrow \geq h_e^{W_e}(x-1)[s] \\ \wedge\ h_e^{W_e}(y)[s] > h_e^{W_e}(y-1)[s]\}.$$

We will build a uniformly W_e-c.e. sequence of sets $\{T_x^\tau\}_{x\in\mathbb{N}}$ at τ. For simplicity we drop the oracle W_e from the notation. At times a bottom node α of τ will enumerate a number r into T_x^τ for some x, with a W_e-use denoted by $t_x^\tau(s)$. In future if W_e doesn't change below this use t_x^τ, then the number r stays in T_x^τ. On the other hand if for some $t > s$, we have $W_{e,t}\restriction t_x^\tau(s) \neq W_{e,s}\restriction t_x^\tau(s)$, then the number r is removed from T_x^τ, i.e. $r \notin T_x^\tau[t]$.

We let r_i^τ record the restraint imposed on A by τ, for the sake of its i-bottom nodes. That is, a single restraint function is used for each entire level. For each incompleteness node $\sigma \supset \tau$, we let I_σ^τ denote the *injury number* that σ has to respect when considering which side (A or C) to enumerate into. That is, σ cannot make any enumeration into $A\restriction j(i)$ for any $i < I_\sigma^\tau$, for the sake of limiting injuries to T_i^τ. If α is an i-bottom node of τ, define the parameter

$$\mathit{marker}_\alpha(s) = \text{least } y < l_\tau(s) \text{ such that } h_e^{W_e}(y)[s] \geq m_e^{i+2} + 1.$$

This marks out the first place y such that $h_e^{W_e}(y)[s] \geq m_e^{i+2}+1$, and $h_e^{W_e}\restriction y+1[s]$ looks like an order. If no such y exists, let $marker_\alpha(s)\uparrow$. The purpose of this definition is to let α wrap its trace axioms around the value $u_e(marker_\alpha)[s]$. If β is a sibling node of α, then for all s, $marker_\alpha(s) = marker_\beta(s)$. We also write $marker_{e,i}(s)$ in place of $marker_\alpha(s)$, and let $marker_{e,-1} = 0$. If W_e has settled on the part which is accessed by α, we will be able to run the strategy at level $|\alpha|$ without interruption. Such a stage s is said to be *α-fixed*, or *(e,i)-fixed*. That is, we have $marker_{e,i}[s]\downarrow$, and $W_{e,s}\restriction u = W_e\restriction u$, where $u = u_e(marker_{e,i})[s]$.

When we *initialize* an incompleteness node σ, we set $z_\sigma, b_\sigma\uparrow$. When we initialize a top node τ, we clear the sequence $\{T_x^\tau\}_{x\in\mathbb{N}}$ of all axioms, and start building it again from scratch. Set $r_i^\tau = 0$ for all i, and set $I_\sigma^\tau = 0$ for each incompleteness node $\sigma \supset \tau$. During the construction, when we *increase* (or *decrease*) a parameter value P to x, we mean that we set $P = x$, unless P is already larger (or smaller) than x, in which case we do nothing.

3.6.5 The construction

At stage $s = 0$, initialize all nodes, and do nothing. Let $s > 0$. We build the stage s approximation to the true path, δ_s of length s inductively. We say that α is visited at stage s, if $\delta_s \supset \alpha$; equivalently we say that s is an α-stage. Assume that $\alpha = \delta_s\restriction d$ has been defined for $d < s$. There are three possibilities for α.

1. *α is an $\mathcal{R}_e$-node* : let $\delta_s(d) = 0$. Pick the first case from the list that applies, and act for α accordingly:

   (a) if b_α and z_α are undefined, pick fresh values for them,

   (b) if $D(z_\alpha) \neq \Phi_e^C(z_\alpha)[s]$, do nothing,

   (c) if $z_\alpha \in D$, pick a fresh value and reassign z_α,

   (d) all the above fails, i.e. $\Phi_e^C(z_\alpha)[s]\downarrow = 0 = D(z_\alpha)$. If $\gamma(b_\alpha, s)\downarrow$, we now have to decide which side (A or C) to enumerate $\gamma(b_\alpha, s)$ into. If

      - $\gamma(b_\alpha, s) > \max\{r_i^\tau \mid \tau \subset \beta^\frown f \subseteq \alpha$ for some top node τ, and some i-bottom node β of $\tau\}$, and
      - $\gamma(b_\alpha, s) > \max\{j(x,s) \mid x < I_\alpha^\tau$ for some top node $\tau \subset \alpha\}$,

both holds then we enumerate z_α into D, and for each top node $\tau \subset \alpha$, we increase the injury number I^τ_α to be 1+ the largest n such that $J^A(n)[s] \downarrow \in T^\tau_n$, with use $j(n,s) > \gamma(b_\alpha, s)$ (if no such n exists, do nothing). Enumerate $\gamma(b_\alpha, s)$ into A. Otherwise if one of the above does not hold, i.e. $\gamma(b_\alpha, s)$ is blocked on the A side, then we enumerate $\gamma(b_\alpha, s)$ into C, and do nothing else.

2. *α is an $\mathcal{N}_e$-node* : let $\delta_s(d) = 0$, and s^- be the previous α-stage (if this is the first α-stage, do nothing). Check if there is a least i such that $marker_{e,i}(s^-) \downarrow$, and $W_{e,s^-} \upharpoonright u_e(marker_{e,i})[s^-] \neq W_{e,s} \upharpoonright u_e(marker_{e,i})[s^-]$. If no such i exists, do nothing. Otherwise consider the least such i - we know that all work previously done at level $\mathcal{N}_{e,i}$ and below has now been undone. So, we can decrease I^α_σ to $marker_{e,i-1}$ for every incompleteness node $\sigma \supset \alpha$.

3. *α is an $\mathcal{N}_{e,i}$-node* : let s^- be the previous α-stage. Check if:

   (a) s^- exists, i.e. we have visited α before.

   (b) $marker_\alpha(s^-)$ and $marker_\alpha(s)$ are both defined and equal.

   (c) $W_{e,s^-} \upharpoonright u_e(marker_\alpha)[s^-] = W_{e,s} \upharpoonright u_e(marker_\alpha)[s^-]$.

   If one of the above 3(a), 3(b) or 3(c) fails, then $h^{W_e}_e$ has not yet stabilized on the part which is required for α; we have not reached an α-fixed stage. In this case drop A-restraint by letting $\delta_s(d) = \infty$, and do nothing else.

   Otherwise if all of 3(a)-(c) holds, then let $\delta_s(d) = f$, and do the following. For each x such that $J^A(x)[s] \downarrow$ and $marker_{e,i-1}(s) \leq x < marker_\alpha(s)$, we enumerate the value $J^A(x)[s]$ into $T^{\tau(\alpha)}_x$ (unless the value is already in there), with W_e-use $t^{\tau(\alpha)}_x(s) = u_e(marker_\alpha)[s]$. Increase the restraint $r^{\tau(\alpha)}_i$ to $j(x,s)$.

This ends the definition of δ_s. At the end of stage s, we initialize all nodes $\beta >_{left} \delta_s$. We take actions for coding. If there is some $x \in \emptyset'_s - \emptyset'_{s-1}$, such that $\Gamma[s]$-axioms currently apply, we enumerate $\gamma(x,s)$ into C. Otherwise, pick the least x for which no $\Gamma[s]$-axioms apply. Set $\Gamma^{A\oplus C}(x)[s] \downarrow = \emptyset'_s(x)$ with fresh use $2\gamma(x,s)$. Go to the next stage.

3.6.6 Verification

The true path of the construction is defined as usual to be the leftmost path visited infinitely often. If τ is a top node and σ is an incompleteness node and $x \in \mathbb{N}$, we say that σ *injures* T^τ_x at stage s, if at stage s, σ enumerates into A below the use of a convergent computation $J^A(x)[s] \downarrow = n$, where $n \in T^\tau_x$. We first show that if τ is on the true path, then each location T^τ_x is injured from above[4] only finitely often:

Lemma 3.6.3. *Let τ be an $\mathcal{N}_e$-node on the true path, and $i \in \mathbb{N}$. Suppose s is an (e, i)-fixed stage such that τ is never initialized after stage s. Then for each $x < marker_{e,i}$, and each $\sigma \in Cone^\tau_i$, σ is allowed to injure T^τ_x at most once after stage s.*

Proof. If σ injures T^τ_x at some stage $t \geq s$, the same action also sets $I^\tau_\sigma > x$. The only way for σ to injure T^τ_x again, is for I^τ_σ to drop to $\leq x$. Since τ is never initialized this must be due to τ decreasing the injury number T^τ_σ under step 2 of the construction. Since t is a τ-stage, this can only happen if $W_{e,t}\restriction u \neq W_e \restriction u$, which is impossible. □

Lemma 3.6.4. *$\Gamma^{A \oplus C}$ is total and equals $\emptyset'$. In particular for each x, $\gamma(x, s)$ eventually settles.*

Proof. We prove the above by induction on x. Fix an x, and let s_0 be a stage after which $\gamma(x - 1)$ is never moved. The only interesting case to consider is when some σ on the true path has picked $b_\sigma = x$. We assume that σ is never initialized after s_0, and that no $\sigma' \subset \sigma$ enumerates after stage s_0, since $b_{\sigma'} < b_\sigma$. If it is the case that σ enumerates $\gamma(x)$ into A after s_0, then we are done. We assume that no enumeration is ever made into A by σ, and that step 1(d) of the construction applies at infinitely many σ-stages. We want to argue that the two conditions in step 1(d) are eventually not satisfied, for a contradiction.

The first condition: fix a top node τ, and an i-bottom β of τ such that $\tau \subset \beta^\frown f \subseteq \sigma$. We argue that $\lim_{s \to \infty} r^\tau_i[s] < \infty$. Since β has true outcome f, there is a β-fixed stage, so eventually at every large enough τ-stage t, we have $\delta_t(|\beta|) = f$. Once we reach such a stage in the construction, the restraint r^τ_i can only be destroyed

[4]We think of the location T^τ_x as being built at the level of the i-bottom nodes, where $x < marker_{e,i}$.

by some node $\sigma' \in Cone_i^\tau$. However r_i^τ only increases if $j(w,t)$ gives a new value for some w, and there are only finitely many w to consider at level $|\beta|$. By Lemma 3.6.3, each of these T_w^τ-boxes are injured only finitely often, so the restraint function r_i^τ is bounded.

Now we look at the second condition: fix a top $\mathcal{N}_e$-node $\tau \subset \sigma$. Since σ never enumerates into A, the injury number I_σ^τ does not increase. Hence, I_σ^τ settles down at a smallest value, say p at some stage t_0. Fix some $w < p$, and we want to show that $j(w,t)$ has only a finite effect on σ. In particular we want to show that the set $\{j(w,t) \mid t \text{ is a } \sigma\text{-stage where step 1(d) applies}\}$ is bounded. If t is a σ-stage such that $J^A(w)[t] \downarrow$ and step 1(d) applies, we would enumerate $\gamma(x,t)$ into C to move $\gamma(x)$ above $j(w,t)$. Since all smaller markers have already settled, it follows that A will never change below $j(w,t)$ after stage t.

Hence the two conditions in step 1(d) will eventually be not satisfied, and σ will eventually find no reason to enumerate $\gamma(x)$ into C - it will have to enumerate $\gamma(x)$ into A instead, a contradiction. $\square$

We first show that A is hyper jump traceable. Fix an $\mathcal{N}_e$-node τ on the true path, such that $h_e^{W_e}$ is an order. If α is a τ-bottom node on the true path, then α has true outcome f, as there will be an α-fixed stage. Let s_0 be the stage where τ is initialized for the last time, hence the true version of $\{T_x^\tau\}_{x\in\mathbb{N}}$ is built after s_0. Fix an x, and let i be such that $m_e^{i+1} < h_e^{W_e}(x) \leq m_e^{i+2}$; this can be done for almost all x, and our task is to show that $|T_x^\tau| \leq m_e^{i+1}$, and if $J^A(x)\downarrow$, then $J^A(x) \in T_x^\tau$. Let x^+ = least such that $h_e^{W_e}(x^+) > m_e^{i+2}$, i.e. $x^+ = \lim_s marker_{e,i}[s]$. Let α be the i-bottom node of τ on the true path. Let s_1 be the least α-fixed stage $> s_0$.

It is not hard to see that $T_x^\tau = \emptyset$ at the beginning of stage s_1: if not then some j-bottom node of τ had made a trace at some stage t, with $s_0 < t < s_1$, with use $u_e(marker_{e,j})[t]$. If W_e changes below $u_e(marker_{e,j})[t]$ after stage t, the trace value would be removed. But if W_e did not change then we must have $j = i$ to match the situation at stage s_1, but this means that $t < s_1$ is (e,i)-fixed, a contradiction. So, $T_x^\tau = \emptyset$ at the beginning of s_1. Anything that we put in T_x^τ during or after stage s_1, will be put in with W_e-use $u_e(x^+)$, and thus will stay in T_x^τ forever. If $J^A(x)\downarrow$, then its value will clearly be placed into T_x^τ after stage s_1, so we certainly have $J^A(x) \in T_x^\tau$. Now we argue that:

Lemma 3.6.5. $|T_x^\tau| \leq m_e^{i+1}$.

Proof. Firstly, observe that there are at most $2^{|\alpha|}$ many τ-stages $t \geq s_1$ such that $\delta_t(|\alpha|) = \infty$, because each sibling node of α can be visited with outcome ∞ at most once, on or after the α-fixed stage s_1. Let $t \geq s_1$ be a stage where $J^A(x)[t]$ is traced into T_x^τ, by some i-bottom node of τ. The same action also increases r_i^τ beyond $j(x,t)$. Which incompleteness node σ can enumerate into $A\restriction j(x,t)$ after stage t? Obviously $\sigma \not<_{left} \tau$, and every node to the right of τ has to obey r_i^τ. If $\sigma \subset \tau$ then by Lemma 3.6.4 it only makes finitely many enumerations into A, so we may assume x is large enough so as not to be affected by σ. This leaves the case when $\sigma \supset \tau$; hence either $\sigma \in Cone_i^\tau$, or else $|\sigma| > |\alpha|$. In the former case there can only be a total of $|Cone_i^\tau|$ many injuries to r_i^τ by Lemma 3.6.3, while the latter case contributes at most $2^{|\alpha|}$ many injuries since r_i^τ is obeyed at $\delta_t(|\alpha|)^\frown f$-stages. Hence there are at most $1 + 2^{|\alpha|} + |Cone_i^\tau| = m_e^{i+1}$ many different values in T_x^τ. $\square$

We show that C is Turing incomplete. Let σ be an $\mathcal{R}_e$-node on the true path, and let $b = \lim b_\sigma$. By Lemma 3.6.4 there is a stage s_0 after which 1(d) never applies at σ-stages. After s_0 the parameter z_σ can get reassigned at most once since 1(d) never applies, so let $z = \lim z_\sigma$. Observe that $D(z) \neq \Phi_e^C(z)$. Hence, A is cuppable, and this concludes the proof of Theorem 3.6.2.

3.7 No c.e. hyper jump traceable set can be promptly simple.

In this section, we show that no c.e. hyper jump traceable set can be promptly simple. Thus, while there can be cuppable hyper jump traceable c.e. sets, none of them can be low cuppable. The construction of a noncomputable c.e. hyper jump traceable set in Section 3.6 was by using a degenerate $0'''$-priority argument, and does not seem to allow prompt enumeration for the positive requirements. In the following theorem we show that this must indeed be the case.

A noncomputable member of $\mathcal{H}$ was constructed without using a direct cost function. $\mathcal{H}$ is therefore the first known example of a subclass of the c.e. K-trivials, which is completely free of the promptly simple sets. Other known subclasses of the c.e. K-trivials, such as the strongly jump traceable, ML-noncuppable, ML-coverable sets, and $\mathcal{C}^\diamond$ for various (Σ^0_3) null classes $\mathcal{C}$, are all shown to be nontrivial by constructing a promptly simple member using cost functions (see Chapter 8 of [Nie09]).

Therefore $\mathcal{H}$ forms a subclass of the cappable c.e. sets. In Theorem 3.9.1 we will extend this result to show that there is a single capping companion for the entire class $\mathcal{H}$.

Theorem 3.7.1. *Suppose A is c.e. and promptly simple. Then, there is a c.e. set C, such that $A \notin SJT(C)$.*

3.7.1 Requirements

We build the c.e. set C, and a Turing functional Ψ to meet the requirements :

$$\mathcal{R}_e \quad : \quad \text{Defeat the } e^{th} \text{ } C\text{-trace. That is, for some } x\text{, either}$$
$$|T^e_x| \geq \Psi^C(x)\text{, or else } J^A(x) \notin T^e_x.$$

Here, we let $\{T^e_x\}_{x\in\mathbb{N}}$ be the e^{th} C-trace, in some effective listing of all traces computing with an oracle. For ease of notations we suppress all mention of C in T^e_x.

3.7.2 Description of strategy

The strategy for this theorem is based on the fact that we could force A to change "promptly", otherwise known as prompt permitting. Suppose we were only given A noncomputable, and we wanted to build a c.e. set C such that $A \notin SJT(C)$. The false proof would go something like this. We want to build a C-order Ψ^C, and defeat every possible C-trace $\{T^C_x\}_{x\in\mathbb{N}}$ via Ψ^C. We take control of $J^A(x)$ for some x, and start by setting $\Psi^C(x) = 1$, and enumerate $J^A(x)[s_0] \downarrow = s_0$ with use u. Once s_0 shows up in T^C_x, we record the string $A_{s_0} \upharpoonright u$. If A changes in future below u, we could set $J^A(x)$ different and be done. There is no guarantee that A will ever change below u, of course, so while waiting for the (potential) change to happen, we have to pick a larger $x' > x$ and repeat. Of course other requirements might have already defined $\Psi^C(y) = 2$, say, for some $x < y < x'$, so we have to enumerate $\psi(x)$ into C

to kill the axiom and set $\Psi^C \upharpoonright [x, x'] = 1$. Eventually A has to change below one of the use u since it is noncomputable, and when it does so we would be done.

Since we know that there is a noncomputable hyper jump traceable set, which part of the above goes wrong? The opponent plays the noncomputable A, which means that he has to change $A \upharpoonright u$ below one of the uses we specify, but he doesn't have to do so *promptly.* In particular, he could put $J^A(x)[s_0]$ into T_x^C with a C-use larger than that of $\psi(x)$, i.e. he wraps his axioms around ours. The opponent then waits for us to pick $x' > x$ and for us to put $\psi(x)$ into C. We needed to do that to ensure that Ψ^C is an order, but we have also unwittingly helped the opponent to clear T_x^C temporarily. The opponent could now seize the opportunity and change $A \upharpoonright u$, and we would be left stranded.

However, if the opponent was put in charge of a promptly simple set A instead, we could force him to change $A \upharpoonright u$ promptly. Only when the opponent fails to respond promptly, do we choose a new $x' > x$, and put $\psi(x)$ into C to clear our Ψ^C axioms (as well as the T_x^C axioms). This is alright because eventually the opponent has to respond with an A-change *before* we clear the C-axioms.

3.7.3 The construction

We arrange our requirements in the order $\mathcal{R}_0 < \mathcal{R}_1 < \cdots$. At each requirement $\mathcal{R}_e$, we will enumerate auxiliary c.e. sets $U_{e,0}, U_{e,1}, \cdots$ to try and force A to change. By a slowdown lemma and the Recursion Theorem, we may assume that if we put numbers $x_0 < x_1 < \cdots$ into an auxiliary set U at stages $s_0 < s_1 < \cdots$ respectively, then A has to promptly permit one of them; namely there is some i such that $A_{s_i} \upharpoonright x_i \neq A_{s_i+1} \upharpoonright x_i$. We may in fact assume that A has to promptly permit infinitely many of the x_i's.

By the Recursion Theorem again, we have an infinite list of indices of functionals we will enumerate during the construction. Let $x(e, s)$ denote the index that $\mathcal{R}_e$ is using at stage s, with use $u(e, s)$. At times $\mathcal{R}_e$ will be initialized; in that case it abandons this index and use and picks a fresh one from the list (not used before). We will let *region*(e, s) record the number n such that $\mathcal{R}_e$ wants $\Psi^C(x(e)) = n$. This also represents the number of elements $\mathcal{R}_e$ wants to force into $T_{x(e)}^e$, before it is satisfied. We let *attempt*(e, s) keep track of the number of elements $\mathcal{R}_e$ has succeeded in forcing into $T_{x(e)}^e$; when *attempt*$(e) =$ *region*(e), its work is done. During the construction we will also enumerate axioms into Ψ^C, maintained globally; again we denote the stage s use of $\Psi^C(x)[s]$ by $\psi(x,s)$.

At stage s of the construction, we pick the least $e < s$ such that $\mathcal{R}_e$ requires attention, i.e. one of the following applies:

(A1) $\mathit{region}(e,s)$ is currently unassigned.

(A2) $\mathit{region}(e,s)\downarrow$, but no axioms in $J^A(x(e,s))[s]$ currently apply.

(A3) $J^A(x(e,s))[s]\downarrow= r$, and $r \in T^e_{x(e,s)}[s]$ and $\mathit{attempt}(e,s) < \mathit{region}(e,s)$.

There are three cases:

- If (A1) applies, we pick a fresh number for $\mathit{region}(e)$, pick a fresh unused index $x(e)$, and set $u(e) = s$ and $\mathit{attempt}(e) = 0$. Set $\Psi^C(x(e,s))[s]\downarrow= \mathit{region}(e,s)$ with a fresh use $\psi(x(e,s),s)$. In fact, to ensure the totality of Ψ^C, we also enumerate axioms with the same C-use for all $y < x(e,s)$ where no axioms currently apply for y. Go to the next stage.

- If (A1) fails but (A2) applies, then we set $J^A(x(e,s))\downarrow= s$ with use $u(e,s)$. Go to the next stage.

- If (A1) and (A2) fails but (A3) holds then we enumerate $u(e,s)$ into $U_{e,\mathit{attempt}(e,s)}$, and set $\mathit{region}(e')\uparrow$ for all $e' > e$.
  Check if A promptly permits $u(e,s)$. If yes, increment $\mathit{attempt}(e)$ by 1, and move on to stage $s+2$. If not, we enumerate $\psi(x(e),s)$ into C to clear the definition of $\Psi^C(x')$ for all $x' \geq x(e)$, and choose a fresh unused index $x(e)$ and set $u(e) = s$ and $\mathit{attempt}(e) = 0$. Also, for all the relevant $x' \leq$ the new $x(e)$, set $\Psi^C(x')[s]\downarrow= \mathit{region}(e,s)$ with fresh $\psi(x',s)$ use. Go to the next stage.

At the start of the construction all *region* parameters are unassigned, so this guarantees that $\mathcal{R}_{s-1}$ requires attention at stage s.

3.7.4 Verification

Lemma 3.7.2. *For each e, $\mathit{region}(e,s)$ eventually settles.*

Proof. Suppose that $r = region(e-1, s_0)$ has settled. The only way for $region(e)$ to be reset after stage s_0, is for $\mathcal{R}_{e-1}$ to receive attention, and $\neg$(A1)$\wedge\neg$(A2)$\wedge$(A3) holds for it. Suppose this happens at infinitely many stages after stage s_0. At each such stage, a number will be enumerated into one of $U_{e-1,0}, \cdots, U_{e-1,r-1}$, and has to be new to the set. Choose the largest $r' < r$ such that $U_{e-1,r'}$ is infinite. Since A promptly permits infinitely many of the numbers in $U_{e-1,r'}$, eventually A promptly permits some number $u(e-1, s')$ enumerated into $U_{e-1,r'}$ at stage s', where the construction will make no further enumeration into any of $U_{e-1,r'+1}, \cdots, U_{e-1,r-1}$. Clearly $r' < r-1$. In fact when $\mathcal{R}_{e-1}$ next receives attention under $\neg$(A1)$\wedge\neg$(A2)$\wedge$(A3) at some stage $s'' > s'$, we must have $r'+1 \leq attempt(e-1, s'') \leq r-1$. This is not possible since we would then enumerate $u(e-1, s'')$ into $U_{e-1,attempt(e-1,s'')}$. □

Lemma 3.7.3. *All requirements are satisfied, and $A \notin SJT(C)$.*

Proof. Firstly, note that Ψ^C is total, nondecreasing and unbounded (by Lemma 3.7.2), and we never add inconsistent Ψ^C axioms. Next, fix an $e \in \mathbb{N}$ and let $r = \lim_{s\to\infty} region(e, s)$ and $x = \lim_{s\to\infty} x(e, s)$. Since we never change the parameter $u(e)$ before A fails to promptly permit, we must have $J^A(x) \downarrow= y$ with a permanent axiom say, after stage s_0. We clearly have $\Psi^C(x) = r$, and if $y \in T_x^e$ then $attempt(e) = r$ must be true when y finally shows up in T_x^e. This corresponds to r different numbers forced into T_x^e; each time a number shows up in T_x^e, the C part of the use that puts it in T_x^e is preserved forever. □

This ends the proof of Theorem 3.7.1. How low can we make the set C in Theorem 3.7.1? Since each $\mathcal{R}_e$ enumerates into C only finitely often, hence C can be made low. On the other hand it is not possible to compute an upper bound for the number of C-enumerations for each $\mathcal{R}_e$, because the opponent could refuse to give us his prompt permission for as many times as he wishes. Thus it is not clear if we can make C strongly jump traceable, or even just superlow.

Question 3.7.4. *Does $\mathcal{H} = \bigcap\{SJT(W) \mid W$ is c.e. and low$\}$?*

This is possible since both classes contain no promptly simple member.

3.8 No c.e. set is strongly jump traceable by every Δ^0_2 set

In this section, we will show that the only c.e. sets which are strongly jump traceable by every Δ^0_2 set, are the computable ones. This property is one which is exclusive to the computable sets - the only way we can jump trace A in this way, is for J^A to be traceable via a constant bound.

Theorem 3.8.1. *For any c.e. set $W >_T \emptyset$, there is a set $A \leq_T \emptyset'$ such that $W \notin SJT(A)$.*

3.8.1 Requirements

We build a Δ^0_2 set A by full approximation, and an A-order Ψ^A. A tree T is a total function from finite binary strings to finite binary strings, such that for all σ, $T(\sigma^\frown 0)$ and $T(\sigma^\frown 1)$ are incompatible extensions of $T(\sigma)$. At each stage s of the construction we define the sequence of trees $T_0[s] \supseteq T_1[s] \supseteq \cdots T_s[s]$, and ensure that for each e, $T_e := \lim_{s\to\infty} T_e[s]$ exists pointwise. In fact, we will ensure that $T_e[s] = T_e$ for some s. To make notations more consistent, for each e, s, σ, we write $T_e(\sigma)[s]$ instead of $T_e[s](\sigma)$ - the value of $T_e[s]$ applied to σ.

For each s, we define the finite string $A_s = T_s(\emptyset)[s]$. Hence by the above, $\lim_{s\to\infty} A_s(x)$ exists for all x, so that by the Limit Lemma, $A \leq_T \emptyset'$. We want to ensure that the following requirements are met:

$$\mathcal{R}_e \;:\; \text{Defeat the } e^{th} \text{ } A\text{-trace. That is, for some } x, \text{ either } |V^A_{e,x}| \geq \Psi^A(x), \text{ or else } J^W(x) \notin V^A_{e,x}.$$

We let $\{V^X_{e,x}\}_{x\in\mathbb{N}}$ be the e^{th} trace computing with an oracle, in some effective enumeration. We append $[s]$ to all parameters to denote the value of the parameter at stage s. To choose a *fresh* number x at stage s, means that we choose $x > s$ and larger than any number previously used or mentioned. We write 1^n to denote the finite string of n many 1's

3.8.2 Description of strategy

We remind the reader of the standard strategy in making $W \notin SJT(A)$. We have to define a single order Ψ^A, and defeat each A-trace $\{V_x^A\}_{x\in\mathbb{N}}$ by meeting the requirements above. Let us consider the things we have to do to defeat a single trace. We pick a follower ξ, and enumerate $J^W(\xi)[s]\downarrow= s$ with use u, and wait for s to be traced in V_ξ^A. When s enters $V_\xi^A[s]$, we want W to change so that we can define $J^W(\xi)$ on a new value. Recall from Theorem 3.7.1 that if W is promptly simple, then we can easily build A to be c.e. and low. Therefore, if the opponent wants to make things difficult for us, he has to make W (which is noncomputable) respond very slowly. In particular, the opponent would do the following. He would trace s into V_ξ^A with an A-use larger than $\psi(\xi)$ (i.e. he wraps his axioms around ours), say at stage t. He knows that he has to change W below u eventually to ensure that W is noncomputable, but he could wait until we enumerate $\psi(\xi)$ into A. He knows that we have to do that, because we have to pick a new $\xi' > \xi$ and start the process above again with ξ'. Only after we enumerate $\psi(\xi)$ into A, would he then change $W\upharpoonright u$.

Note that as discussed in Theorem 3.6.1, if we had to make A c.e., we would not be able to succeed. However, we are allowed to make $A \in \Delta_2^0$, so we could actually restore A-traces back to the state which they were at previously. If the opponent does change $W\upharpoonright u$ eventually, we could restore A back to what it was, A_t, at stage t. Doing so would put s back into the trace V_ξ^A.

The formal construction is given below. Trees are used to keep track of nodes which we might have to return to, if the opponent changes W at a later stage.

3.8.3 Notations

Each requirement $\mathcal{R}_e$ will make several attempts at defeating the e^{th} A-trace, which we call *attacks*. The Recursion Theorem gives us a list of indices $i_0, i_1, \cdots$ which we can use to control $J^W(i_n)$. When $\mathcal{R}_e$ begins its i^{th} attack, it will pick an index from the list, which we denote by ξ_i^e. $\mathcal{R}_e$ will then control $J^W(\xi_i^e)$, with W-use u_i^e. Each of the parameters $\xi_1^e, \xi_2^e, \cdots$ represents a separate attack of $\mathcal{R}_e$. Basically, the construction visits $T_e(0)$ if $\mathcal{R}_e$ is performing the first attack, and visits $T_e(1)$ when it is waiting for a $W \upharpoonright u_1^e$-change, i.e. the appropriate numbers have entered the trace V^{σ}_{e,ξ_1^e} for some $\sigma \supset T_e(0)$. When that happens, we would set $T_e(0) = \sigma$, and the construction will visit $T_e(10)$ to start the second attack, and when the opponent responds again, we will go to $T_e(110)$ to begin the third attack, and so on. Eventually $W \upharpoonright u_i^e$ has to change for some i, and when that happens we can go back to $T_e(1^{i-1}0)$ and enumerate a new computation for $J^W(\xi_i^e)$. Each time $\mathcal{R}_e$ requires attention, we will move $T_r(\emptyset)$ for all $r > e$ above some branch of T_e, so that the requirements $\mathcal{R}_r$ for $r > e$ can start their attacks anew, above the branch of T_e we want restored.

Each requirement $\mathcal{R}_e$ will pick a number $region(e)$, which denotes the number r such that we want to try and make $|V^{\sigma}_{e,\xi_i^e}| \geq r$ (provided that numbers are always traced). Obviously this means that $\mathcal{R}_e$ will have to enumerate the axiom $\Psi^{\sigma}(\xi_i^e) \downarrow = r$ for each i^{th} attack it begins. Care has to be taken to ensure that no incompatible Ψ-axioms are enumerated, and that Ψ^A eventually turns out to be an order. Hence $\mathcal{R}_e$ will enumerate Ψ-axioms for the path $T_e(1^{i-1}0)$, when it is beginning the i^{th} attack. Other requirements $\mathcal{R}_k$ for $k > e$ start above $T_e(1^{i-1}0)$, and enumerate their own Ψ-axioms above $T_e(1^{i-1}0)$. When $\mathcal{R}_e$ needs to begin a new j^{th} attack, it has to start on a new branch $T_e(1^{j-1}0)$, incompatible with $T_e(1^{i-1}0)$.

We build a Turing functional Ψ, which we think of as a c.e. set of axioms. The axioms are of the form $\langle x, y, \sigma \rangle$, which means "given input x, output the number y if $\sigma \subset X$", where X is the oracle. At stage s of the construction, when we say that we set $\Psi(x)[s] \downarrow = y$ with use σ, we mean that we enumerate the following Ψ-axioms: for each $x' \leq x$, enumerate the axiom $\langle x', y, \sigma \rangle$, if for every Ψ-axiom $\langle x', y', \sigma' \rangle$ already present, we have σ' incompatible with σ. This ensures that new axioms are always compatible with existing ones, and that along any path $X \in 2^{\omega}$, the function Ψ^X will be nondecreasing whenever it is defined.

For each e, i, when $\mathcal{R}_e$ begins its i^{th} attack, we pick a fresh index for ξ_i^e, and set $\Psi(\xi_i^e)[s] \downarrow = region(e)$, with use $T_e(1^{i-1}0)$. It will then carry out its i^{th} attack by setting $J^W(\xi_i^e) \downarrow$, and then wait for the value to be traced. Each time a new value appears in the trace (on some use extending $T_e(1^{i-1}0)$), we increment the counter $attempt(e, i)$ by 1. When the counter $attempt(e, i)$ reaches $region(e)$ for *any* attack i, then $\mathcal{R}_e$ is permanently satisfied, and need not be considered any further (unless it is later initialized).

We say that an attack of $\mathcal{R}_e$ is *e-pending* if it is waiting for some number to enter the appropriate trace; at any time at most one attack of $\mathcal{R}_e$ is pending, and any e-pending attack will have priority over all the other attacks of the same requirement. When we *initialize* a requirement $\mathcal{R}_k$, we set $region(k), \xi_i^k, u_i^k$ undefined for all i, set $attempt(k,i) = 0$ for all i, and remove the k-pending status from all the attacks of $\mathcal{R}_k$. We let $Full(\sigma)$ be the full tree above σ, i.e. $Full(\sigma)(\nu) = \sigma^\frown\nu$ for all ν. All parameters retain their assigned values until reassigned or initialized.

3.8.4 The construction

At stage $s = 0$, we make all parameters undefined, and set $T_0[0] = Id$. At stage $s > 0$ we define the trees $T_0[s] \supseteq \cdots \supseteq T_s[s]$. For each $e \leq s$, we do the following. If there is some i such that $attempt(e,i) \geq region(e)$, i.e. the i^{th} attack has succeeded, then let $T_e[s] = T_e[s-1]$, and go to the next e. Otherwise, do the following:

1. If $region(e)$ is currently undefined, pick a fresh follower for it.

2. If there is currently no e-pending attack, we set $T_e[s] = T_e[s-1]$ and begin a new attack by doing the following. Pick the least i such that $\xi_i^e \uparrow$, and start the i^{th} attack. We pick fresh followers for ξ_i^e and u_i^e, and set $\Psi(\xi_i^e)[s] \downarrow = region(e)$ with use $T_e(1^{i-1}0)[s]$. We initialize $\mathcal{R}_k$, and set $T_k[s] = Full(T_e(1^{i-1}0)[s])$ for all $e < k \leq s$. Declare i to be the attack of $\mathcal{R}_e$ that is now e-pending. Go to the next stage.

3. If there is a currently e-pending attack i, pick the action which applies:

   (a) $J^W(\xi_i^e)[s] \uparrow$: we set $J^W(\xi_i^e) \downarrow = s$ with use u_i^e, and set $T_e[s] = T_e[s-1]$. We initialize $\mathcal{R}_k$ and set $T_k[s] = Full(T_e(1^{i-1}0)[s])$ for $e < k \leq s$. Go to the next stage.

   (b) $J^W(\xi_i^e)[s] \downarrow$*, but for some* $\sigma \supset T_e(1^{i-1}0)[s-1]$*, we have* $|V^\sigma_{e,\xi_i^e}[s]| >$ $attempt(e,i)$*. (All searches and computations are limited to a use and a search time of* s*)*: hence there is some extension σ which gives us an increase in the size of the trace. Do the following.

      - Set $T_e(1^{i-1}0^\frown\nu)[s] = \sigma^\frown\nu$ for all ν, and $T_e(\eta)[s] = T_e(\eta)[s-1]$ everywhere else.

- Increase the value of *attempt*(e, i) to $|V^{\sigma}_{e,\xi^e_i}[s]|$.
- If *attempt*(e, i) is now at least as big as *region*(e), we initialize $\mathcal{R}_k$ and set $T_k[s] = \mathit{Full}(T_e(1^{i-1}0)[s])$ for all $e < k \leq s$, and go to the next stage. Otherwise we have to decide which attack is going to be e-pending next: firstly, remove the e-pending status from i. Next, pick the least j, if there is one, such that $\xi^e_j \downarrow$ and $J^W(\xi^e_j) \uparrow$, and declare j to be e-pending.

(c) *Neither (a) nor (b) holds*: Do nothing, let $T_e[s] = T_e[s-1]$, and go to the next e.

3.8.5 Verification

It follows directly from the steps in the construction that at each stage s, the trees $T_0[s] \supseteq \cdots \supseteq T_s[s]$ are all defined. Let $A = \lim_{s\to\infty} T_s(\emptyset)[s] \leq_T \emptyset'$ (by Lemma 3.8.2). It is obvious that if an attack i of $\mathcal{R}_e$ is e-pending, then it remains e-pending until either $\mathcal{R}_e$ is initialized, or *attempt*(e, i) increases. This is the reason why we have the "pending" status - to hold the construction through $T_e(1^{i-1}0)$ until we are able to find a node extending $T_e(1^{i-1}0)$, with a larger trace size. We will first show that all trees $T_e[s]$ will eventually stop changing:

Lemma 3.8.2. *For each e, $\mathcal{R}_e$ is initialized only finitely often, and $T_e[s] = T_e$ for some s.*

Proof. We prove the above simultaneously by induction on e. Assume that s_0 is a stage where the statement holds for $e' < e$. After s_0, $\mathcal{R}_e$ can only be initialized when we are taking actions for $\mathcal{R}_{e-1}$. Since $T_{e-1}[s_0] = T_{e-1}$, it follows that $\mathcal{R}_{e-1}$ does not begin any new attack after s_0. Let $u = \max\{u^{e-1}_i[s_0] : \text{attack } i \text{ of } \mathcal{R}_{e-1} \text{ eventually begins}\}$, and wait until $W_{s_1}\upharpoonright u = W\upharpoonright u$. After stage s_1, $\mathcal{R}_e$ is never initialized.

Next, we argue that T_e exists. Since $\mathcal{R}_e$ is initialized only finitely often, let $r = \lim \mathit{region}(e)$. Since every change in $T_e[s]$ corresponds to an increase in *attempt*(e, i) for some attack i, it follows that if T_e does not exist, then no attack remains e-pending forever, and that all attacks of $\mathcal{R}_e$ are eventually started. We assume, contrary to the statement of the lemma, that this is the case. We let $x = \limsup_{i\in\mathbb{N}} \mathit{attempt}(e, i)$, and we may also assume $x < r$ (in fact, no attack i ever attains *attempt*$(e, i) = r$), otherwise $\mathcal{R}_e$ will be permanently satisfied. Then, we claim that W is computable, for a contradiction. For each i, we can compute $W\upharpoonright u_i^e$ as follow: run the construction, and wait until the first $j \geq i$ such that *attempt*$(e, j) = x$, say at stage $s_2(i)$.

We claim that $W_{s_2(i)} \upharpoonright u_i^e = W \upharpoonright u_i^e$ is correct for almost all i: suppose that $W_{s_2(i)}\upharpoonright u_i^e \neq W\upharpoonright u_i^e$. Therefore $J^W(\xi_j^e)[t] \uparrow$ for some $t > s_2(i)$ after the change in W. This means that attack j must become e-pending eventually, and since no attack can be forever e-pending, we must have *attempt*$(e, j) > x$. By the definition of x, the procedure above to compute W can fail for only finitely many i's. Hence W is computable, and this contradiction shows that T_e must exist. □

Fix a requirement $\mathcal{R}_e$. There is some attack i such that either *attempt*$(e, i) \geq \lim$ *region*(e), or i is forever e-pending. Let $i(e)$ denote this i. Note that $i(e)$ denotes the attack that has a successful run, i.e. either $V^A_{e,\xi^e_{i(e)}}$ fails to trace $J^W(\xi^e_{i(e)})$, or has size larger than *region*(e). The following facts are not hard to verify:

1. For every e, $\xi^e_{i(e)} < \xi^{e+1}_{i(e+1)}$, since the former is always chosen before the latter.
2. For every e, $A \supset T_e(1^{i(e)-1}0)$.
3. For each attack i of $\mathcal{R}_e$ that is eventually started, we have $\Psi^{T_e(1^{i-1}0)}(\xi_i^e) = r$, where $r = \lim$ *region*(e).

Lemma 3.8.3. Ψ^A *is an order, i.e. total, nondecreasing, unbounded.*

Proof. From the facts above, we have for every e, $\Psi^A(\xi^e_{i(e)}) = \lim$ *region*(e). This clearly means that Ψ^A is total, and unbounded. Next, we want to see that Ψ^X is nondecreasing along any path $X \in 2^\omega$. Suppose $\mathcal{R}_e$ enumerates the axiom $\langle x, y, \sigma\rangle$ at stage s; note that this only happens when $\mathcal{R}_e$ is starting a new attack i, hence $T_e(1^{i-1}0)[s] = \sigma \subset X$.

Suppose $\mathcal{R}_k$ enumerates a Ψ-axiom $\langle x', y', \sigma'\rangle$ at some stage $t > s$, where $\sigma' \subset X$. Since σ and σ' are compatible this means that $x' > x$. The only thing we have to ensure, is that $y' \geq y$. If $\mathcal{R}_k$ was initialized between s and t, then $y' > y$ so there is no problem. We may therefore assume that $\mathcal{R}_k$ was not initialized. Hence, $k \not> e$, and if $k = e$ then $\sigma' \supset T_e(1^i)[t] = T_e(1^i)[s]$ has to be incompatible with σ so that is impossible (because we start a new attack of $\mathcal{R}_e$ at stage t). Lastly, if $k < e$ then because T_e is always restarted above a branch of T_k which is k-pending, we must have $\sigma \supset T_k(1^{j-1}0)[s]$ for some pending j-attack of $\mathcal{R}_k$ at stage s. Since a new attack of $\mathcal{R}_k$ is started at stage t, we must have $\sigma' \supset T_k(1^j)[t] = T_e(1^j)[s]$ incompatible with σ, again impossible. $\square$

Lemma 3.8.4. *All requirements are met.*

Proof. Fix an e, and consider the $i(e)^{th}$ attack of $\mathcal{R}_e$. Let $x = \xi^e_{i(e)}$, and $r = \lim region(e) = \Psi^A(x)$. Note that $|V^A_{e,x}| \geq |V^{T_e(1^{i(e)-1}0)}_{e,x}|$, so if $attempt(e, i(e)) \geq r$, then $|V^A_{e,x}| \geq r$ and we are done. Suppose not, i.e. $i(e)$ is forever e-pending. We must have $J^W(x)\downarrow$ and receives the definition at a stage s, when $i(e)$ is finally e-pending. Hence $J^W(x) = s \notin V^{T_e(1^{i(e)-1}0)}_{e,x}[t]$, where we have $|V^{T_e(1^{i(e)-1}0)}_{e,x}[t]| = attempt(e, i(e))[s]$ at the previous stage $t < s$, where we increased $attempt(e, i(e))$. Since $A \supset T_e(1^{i(e)-1}0)$, hence if $s \in V^A_{e,x}$, then some σ where $T_e(1^{i(e)-1}0) \subset \sigma \subset A$, will return a successful search under step 3b in the construction at a later stage past s, which will again increase $attempt(e, i(e))$. This is not possible since $i(e)$ never loses its e-pending status after stage s. $\square$

3.9 The degrees containing hyper jump traceable sets are not downwards dense

In this section, we demonstrate yet another strange behaviour exhibited by the degrees containing hyper jump traceable sets. One might expect that if a class of degrees contain very little information, then every noncomputable c.e. degree would bound a member of that class. For instance, every noncomputable c.e. degree bounds a strongly jump traceable set $A >_T \emptyset$. However, this is not true for the hyper jump traceable sets - they are not downwards dense despite their strong resemblance to the computable sets.

This is due to the fact that we have to consider noncomputable functions, which are slightly more tricky to deal with because they may change their mind in their approximation. Such functions might in reality be eventually constant, but infinitely often will try and fool us into thinking that they are unbounded. An example is the following function f with the approximation f_s:

$x =$	0	1	2	3	4	5
$f_0(x)$	1	2	3	4	5	6
$f_1(x)$	1	1	2	3	4	5
$f_2(x)$	1	1	1	2	3	4
$f_3(x)$	1	1	1	1	2	3

Informally, a hyper jump traceable set A imposes a lot of negative restraint, so it is difficult to get elements into the set. In this respect, A is very close to being computable. Even though this is the case, the manner in which A allows numbers to enter is a little bit more complicated than the other low sets such as K-trivials or strongly jump traceable sets (both of these constructions can be described using cost functions). Roughly speaking, elements can only enter A at certain "gap phases" in which restraints on A are temporarily taken down; these windows of opportunity are synchronized with changes in the approximation to functions such as f above.

As we have already seen, the construction of a noncomputable hyper jump traceable set (Theorem 3.6.2) is not compatible with making it promptly simple (Theorem 3.7.1), because of this reason. Such a construction also does not seem to be compatible with permitting below a given c.e. set $B >_T \emptyset$, again due to the same reason : when permission is given to us by B to change A, we might not be able to do so because some function h^W is currently telling us that it is a true order function (hence imposing a large amount of A-restraint). There is no way for us to figure out if it is lying or not, and when the opportunity for us to effect changes on A passes, the opponent will then reveal the true form of h^W. The resulting A-gap which is opened is of no use to us for B-permission, though we could still use it to make A noncomputable. The failure of this strategy can be turned into a proof of the following theorem.

Theorem 3.9.1. *There is a c.e. set $A >_T \emptyset$ such that for any set X, if $\emptyset <_T X \leq_T A$, then X is not hyper jump traceable.*

An immediate corollary to this theorem, is that there is a single c.e. set which forms a minimal pair with every hyper jump traceable set:

Corollary 3.9.2. *There is a c.e. set $A >_T \emptyset$, such that if X is hyper jump traceable, then A and X forms a minimal pair.*

Therefore, the hyper jump traceable c.e. sets are far away from being promptly simple, in the sense that they can all be capped by a single c.e. set.

3.9.1 Requirements

We build a coinfinite c.e. set A, and ensure that it is noncomputable via the usual simple requirements.

$$\begin{aligned}\mathcal{P}_e &: |W_e| = \infty \Rightarrow A \cap W_e \neq \emptyset,\\ \mathcal{N}_e &: \text{If } \Phi_e^A \text{ is total, then either } \Phi_e^A = X_e \text{ is computable, or else } X_e \text{ is not}\\ & \text{ hyper jump traceable.}\end{aligned}$$

Here, Φ_e denotes the e^{th} Turing reduction. The stage s use of the computation $\Phi_e^A(x)$ is denoted by $\varphi_e(x,s)$. At each requirement $\mathcal{N}_e$, we will build a c.e. set C and a Turing functional h, for the sake of demonstrating that X_e is not hyper jump traceable. We ensure that h^C is nondecreasing, and will ultimately be total and unbounded unless X_e is computable. Consider an effective enumeration $\{T_{0,x}\}_{x\in\mathbb{N}}, \{T_{1,x}\}_{x\in\mathbb{N}}, \cdots$ of all possible traces with an oracle. We split $\mathcal{N}_e$ into the subrequirements $\mathcal{N}_{e,0}, \mathcal{N}_{e,1}$, where $\mathcal{N}_{e,i}$ will try and diagonalize against the i^{th} trace:

$$\mathcal{N}_{e,i} : \text{if } X_e >_T \emptyset, \text{ then for some } x, \text{ either } |T_{i,x}^C| \geq h^C(x), \text{ or } J^{X_e}(x)\downarrow \notin T_{i,x}^C.$$

3.9.2 Description of strategy

We first outline the basic strategy used to satisfy a single $\mathcal{N}_{e,i}$ requirement, which has to diagonalize the i^{th} trace $\{T_{i,x}^C\}$. Suppose $\mathcal{N}_{e,i}$ wants to try and make $|T_{i,x}^C| \geq 2$ for some x, where C is built at the main requirement $\mathcal{N}_e$. The naive try would be to do the following. We pick some x_0, and set $J^{X_e}(x_0)\downarrow= s$ with X_e-use j_0, and wait for the value s to be traced in T_{i,x_0}^C. When s shows up in T_{i,x_0}^C, we would then define $R\restriction j_0 = X_e\restriction j_0$ where R is the procedure we are building to compute X_e. We would then repeat the above with a larger $j_1 > j_0$ and a different $x_1 > x_0$. At the end of the day if $X_e\restriction j_n$ never changes after $R\restriction j_n$ is set, then R correctly computes X_e; on the other hand if $X_e\restriction j_n$ changes on some j_n we could go back and define $J^{X_e}(x_n)\downarrow=$ a new value, and we would be able to meet $\mathcal{N}_{e,i}$.

The above strategy works well when considering $\mathcal{N}_{e,i}$ in isolation, but cannot possibly work when combined with other $\mathcal{N}_{e,j}$'s. This is because we had not made use of the fact that $X_e \leq_T A$ - indeed we are trying to show that there can be no hyper jump traceable set. The problem shows up when the opponent takes a long time to trace $J^{X_e}(x_0)$ into T^C_{i,x_0}; in fact the opponent can delay responding until we define $h^C(y) \downarrow= 3$ for some $y > x_0$, say with C-use $u(y)$. He would then trace $J^{X_e}(x_0)$ into T^C_{i,x_0} with a C-use larger than $u(y)$. When we pick a new $x_1 > y$ and j_1 and repeat the above, we would have to define $h^C(x_1) = 2$; in order to do that we have to enumerate $u(y)$ into C and clear the opponent's trace T^C_{i,x_0} at the same time. Thus even if X_e changes below j_0 in future, we will not be able to benefit from it.

The fortunate thing is that we do not need to consider arbitrary X_e, only those X_e below A. When we set $R\restriction j_n$ we could actually freeze $A\restriction \varphi_e(j_n)$ at the same time, so that $X_e\restriction j_n$ will not change. However we need to arrange for the A-restraint to be dropped infinitely often, since the restraint might go to ∞ in the case when X_e is computable. The following are the steps for doing this, at the n^{th} cycle:

Step 1: set $J^{X_e}(x_n) \downarrow$ with use j_n, and wait for the trace to appear in T^C_{i,x_n}. When it appears, we *open up a gap* by dropping all A-restraint. Go to step 2.

Step 2: we *close the gap* by doing the following. Check if $X_e\restriction j_n$ has changed during the gap open phase. If yes, then success is achieved for $\mathcal{N}_{e,i}$. If not, we compute $R\restriction j_n = X_e\restriction j_n$, and re-impose A-restraint. Start the $n + 1^{th}$ cycle from step 1 with a larger j_{n+1}, x_{n+1}.

Thus if only finitely many gaps are opened, then when the last gap is closed we have the desired X_e-change. If infinitely many gaps are opened and closed, then R computes X_e correctly. On a construction tree, we can arrange things so that restraint is dropped at infinitely many stages. Note the similarity between this strategy, and the one used in Theorem 3.7.1. In both cases we wait for the opponent to respond with a trace on an index x_n. When that happens, we challenge the opponent to change X_e. If the opponent refuses to respond with a X_e-change we will *abandon this index x_n and never return to it.* In Theorem 3.7.1 we never need to return to an abandoned index since a prompt change in X_e is guaranteed. In our case here, we can impose restraint and force X_e to change only during gap phases -

even though the indices which were abandoned previously cannot benefit from this change, we make sure that the one which is currently active can benefit from the change.

We now describe the strategy in the general case. Assume first of all that X_e is c.e.. We start by picking a number k and x_0, and set $h^C(x_0)\downarrow = k$. We now need to have k many consecutive gap phases in which X_e changes. Each run of attempts is called a *cycle*. We will need to have k many attempts $R_1, \cdots, R_k$ at computing X_e, and have restraint functions $r_1, \cdots, r_k$ for each of these procedures.

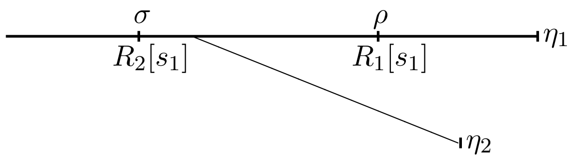
We start a new cycle by picking a fresh follower x. This follower is used throughout the current cycle, and will only be abandoned when the current cycle is ended and a new one begins. We start by running the basic strategy and waiting for it to return. That is, we wait for $J^{X_e}(x)$ to show up in $T^C_{i,x}$. When that happens, we open up a 1-gap by dropping restraint r_1. At the next expansionary stage, we close the 1-gap by the following: if there had been no X_e change below the use, then we extend the procedure R_1 and re-impose the r_1-restraint. We end the current cycle and start a new one. If there had been a change then we run the basic strategy again with r_2, R_2 in place of r_1, R_1, since we are now able to set a new $J^{X_e}(x)$ axiom. This continues on until either we get k consecutive gap phases in this current cycle (and hence $|T^C_{i,x}| \geq k$), or else some k'-gap is closed with no X_e-change, ending the cycle.

If a single subrequirement $\mathcal{N}_{e,i}$ of $\mathcal{N}_e$ opens and closes infinitely many gaps, then X_e would be computable. If k' is the smallest number such that infinitely many k'-gaps are opened and closed, then $R_{k'}$ is total and computes X_e correctly. In this case $\mathcal{N}_{e,i}$ actually abandons infinitely many indices making h^C into a constant function. This is how we exploit such order functions to our advantage - infinitely often it will appear to the opponent that it is a real order.

Now let us consider the general situation when X_e is Δ^0_2. For simplicity we will only describe what goes on in cycles which only require two consecutive gap phases in which X_e changes. We will construct procedures R_1, R_2, and restraints r_1, r_2 just as in the c.e. case. Each time a 1-gap is closed, we have the two cases (as before): either no X_e-change (in which case we extend R_1, r_1 and end the cycle), or else there is an X_e-change (in which case we carry on with a 2-gap). However, when a 2-gap is closed, the opponent has one extra option. He could now recover X_e back to the use of the computation enumerated earlier before the 1-gap opened. Thus, the follower x would now be useless, since we are unable to enumerate a new (third) computation for $J^{X_e}(x)$.

The important point to note is that should this happen (say at stage t), then we will have $X_{e,t} \supset R_1$ again. We could then end the current cycle, and we would have made progress because R_1 is currently correct. Also, R_2 is never extended unless a 2-gap is closed with no X_e-change. Hence $R_2[t] \subseteq R_1[t] \subset X_{e,t}$ is correct as well. Hence if infinitely many cycles are ended with R_2 being extended, then R_2 is total and computes X_e correctly. On the other hand if $\lim |R_2| < \infty$, then almost all cycles will be ended with a recovery of X_e back to a previous configuration. In this case, R_1 will be total and computes X_e correctly.

We illustrate the above with an example. Suppose at stage s_1, we have $R_2[s_1] = \sigma \subset R_1[s_1] = \rho$. We have currently started a new cycle with a computation with use $\eta_1 \supset \rho$. Suppose this computation shows up in the trace, and we then open a 1-gap. When the 1-gap is closed, we found that X_e has changed. We then challenge the opponent with a new computation with use η_2. Note that at this stage we had only opened a 1-gap but not a 2-gap so that η_2 and ρ are incompatible extensions of σ:



When the opponent returns by tracing the new computation, we will open up a 2-gap. Note that during the 2-gap, X_e can change even below σ. When the 2-gap is closed at stage s_2, there are three possibilities:

1. *No X_e-change, that is $X_{e,s_2} \supset \eta_2$.* Then, we extend $R_2[s_2] = \eta_2$, and restart R_1 above η_2. End the cycle.

2. *X_e returns to the previous use, i.e. $X_{e,s_2} \supset \eta_1$.* Then, we extend $R_1[s_1] = \eta_1$. End the cycle. Note that R_2 is unchanged by this action.

3. *Otherwise.* Then, X_{e,s_2} is new and we can enumerate a new computation with that use. The node can now stop acting, having completed the diagonalization successfully.

Note that R_1 and R_2 are always built by extension (unless R_1 is restarted due to R_2 injuring it). Hence one of the two will be total computable, and compute X_e correctly. In general, if n-gaps are opened infinitely often, then one of $R_1, \cdots, R_n$ will be total and compute X_e correctly.

3.9.3 Construction tree layout

The construction takes place on an infinite branching tree. Nodes of even length $|\alpha| = 2e$ are assigned the requirement $\mathcal{P}_e$ with a single outcome 0. Nodes of length $|\alpha| = 2\langle e, i\rangle + 1$ are assigned the requirement $\mathcal{N}_e$ if $i = 0$, and the subrequirement $\mathcal{N}_{e,i-1}$ if $i > 0$. Nodes assigned the requirement $\mathcal{N}_e$ have two outcomes $\infty <_{left} f$. They stand respectively for infinitely many, and finitely many expansionary stages. The subrequirement $\mathcal{N}_{e,i}$ of $\mathcal{N}_e$ has infinitely many outcomes $f >_{left} 1 >_{left} 2 >_{left} \cdots$, with order type ω^*. The rightmost outcome f represents the fact that X_e is noncomputable and hence diagonalization will have to succeed, while the outcome n to the left of f represents the fact that X_e is computable via one of the attempts $R_1, R_2, \cdots, R_n$ at computing it. The reason why we label the outcomes as such is because during the construction, $\mathcal{N}_{e,i}$ will make a few attempts at computing X_e; each time it plays outcome n we make progress on one of the attempts $R_1, R_2, \cdots, R_n$. Note that we also identify outcome f with the natural number 0, so as to keep the notations consistent.

We order nodes on the tree lexicographically: denote $\alpha <_{left} \beta$ to mean that α is strictly to the left of β (i.e. there is some $i < \min\{|\alpha|, |\beta|\}$ such that $\alpha\restriction i = \beta\restriction i$ and $\alpha(i) <_{left} \beta(i)$). We say that α *is a $\mathcal{Q}$-node* if α is assigned the requirement $\mathcal{Q}$. α is a *positive node*, if α is a $\mathcal{P}_e$-node for some e. We say that α is a *top node*, if α is an $\mathcal{N}_e$-node for some e, and that α is an *i-bottom node of* τ if α is an $\mathcal{N}_{e,i}$-node for some e, i, and $\tau \subset \alpha$ where τ is an $\mathcal{N}_e$-node. If α is a bottom node, then $\tau(\alpha)$ denotes the (unique) top node τ such that α is a bottom node of τ. Note that in order to keep the labelling of nodes simple, we had actually allowed some nodes to be given a label for which it never needs to act. Therefore, we need to specify when a bottom node α is an *active* node: if $\alpha \supset \tau(\alpha)^\frown\infty$, and for every $\tau(\alpha)$-bottom node β such that $\beta \subset \alpha$, we also have $\beta^\frown f \subset \alpha$. Bottom nodes which are not active do nothing in the construction, and will always play the outcome f when visited.

3.9.4 Notations

Let α be an active $\mathcal{N}_{e,i}$-node, and $\tau = \tau(\alpha)$. We measure the length of convergence at the top node τ:

$$l_\tau[s] := \max\{y < s \mid (\forall x < y)\Phi_e^A(x)[s] \downarrow\}.$$

α will act each time $l_\tau[s]$ gets long enough, at τ-expansionary stages. At the node τ we build a c.e. set C_τ and a Turing functional h_τ[5]. The C_τ-use of an enumerated $h_\tau(x)$-axiom (if it still applies at stage s) is denoted by $u_\tau(x, s)$.

α works by first picking a number $region_\alpha$. It will also pick a number x_α, and then set $h_\tau^{C_\tau}(x_\alpha) \downarrow= region_\alpha$. In this current run, α will attempt to diagonalize against the i^{th} trace by focussing on $T_{i,x_\alpha}^{C_\tau}$. x_α is picked from an infinite list of indices, given by the Recursion Theorem, which we will use to control $J^{X_e}(x_\alpha)$. α will also pick a number j_α, which denotes the length of X_e such that α will enumerate $J^{X_e}(x_\alpha)$-computations with that much use. The parameter $attempt_\alpha$ will record how successful α is in diagonalization - this keeps track of at least how many elements are currently in $T_{i,x_\alpha}^{C_\tau}$. At the same time, α will be trying to build several effective procedures to compute X_e. More specifically, it will be simultaneously building at any one time, $region_\alpha$ many procedures to compute X_e, and imposing various restraints for each of these procedures. These procedures are denoted by $R_{\alpha,1}, R_{\alpha,2}, \cdots$, which should be viewed as finite strings. Let $r_{\alpha,n}$ denote the A-restraint function imposed for the n^{th} procedure $R_{\alpha,n}$. Finally, we let F_α denote the state of α's strategy. This may be 0 (which means that α has not yet started on its strategy), 1 (which means that α is waiting for a number to appear in the trace), or 2 (which signifies that α has opened a certain A-gap).

A stage s is *τ-expansionary*, if either $s = 0$, or else

1. τ is visited at stage s of the construction,

2. $l_\tau[s] > l_\tau[s^-]$ where s^- is the previous τ-expansionary stage, and

3. $l_\tau[s] > j_\beta$ for every active τ-bottom node β.

[5]We use lowercase h_τ even though it is a Turing functional; this is to respect the fact that $h_\tau^{C_\tau}$ is intended to be an order function.

When we *initialize a bottom node* α, we set $region_\alpha, attempt_\alpha, x_\alpha$ and j_α undefined, and set $F_\alpha = 0$, $r_{\alpha,n} = 0$ and $R_{\alpha,n} = \emptyset$ for all n. When we *initialize a top node* τ, we set $C_\tau = \emptyset$, remove all axioms in h_τ, and initialize all τ-bottom nodes. During the construction, we will enumerate axioms for $J^{X_e}(x)$. The index x is chosen from an infinite list of indices given to us by the Recursion Theorem. When we say at a certain stage s, that $J^{X_e}(x)[s]\downarrow$, we mean that there is an axiom (previously enumerated), that currently applies at stage s.

3.9.5 The construction

At stage $s = 0$, initialize all nodes, and do nothing. Let $s > 0$. We build the stage s approximation to the true path, δ_s of length s inductively. Assume that $\alpha = \delta_s\restriction d$ has been defined for $d < s$. There are three possibilities for α.

1. *α is a $\mathcal{P}_e$-node*: let $\delta_s(d) = 0$, and if $W_{e,s} \cap A_s \neq \emptyset$ do nothing. Otherwise, check if there is some $x \in W_{e,s}$, where $x > 2e$, $x > \max\{t < s \mid \delta_t <_{left} \alpha\}$, and for each bottom node $\beta \subset \alpha$, we have $x > \max\{r_{\beta,n} \mid n <_{left} \alpha(|\beta|)\}$. If such an x exists, enumerate the least such into A and initialize all nodes $\beta \supset \alpha$.

2. *α is an $\mathcal{N}_e$-node*: if s is α-expansionary, let $\delta_s(d) = \infty$. Otherwise let $\delta_s(d) = f$.

3. *α is an $\mathcal{N}_{e,i}$ node*: let $\tau = \tau(\alpha)$. If $attempt_\alpha = region_\alpha$ or α is not active, then we do nothing. Otherwise, take the relevant action below based on F_α:

   (a) $F_\alpha = 0$: pick fresh numbers for $region_\alpha$, x_α and j_α. Set $attempt_\alpha = 0$. For all $x' \leq x_\alpha$ for which no $h_\tau^{C_\tau}$-axioms currently apply, set $h_\tau^{C_\tau}(x')[s]\downarrow=$ $region_\alpha$ with a fresh use $u_\tau(x', s)$. Also set $F_\alpha = 1$.

   (b) $F_\alpha = 1$: there are three possibilities.

      i. $J^{X_e}(x_\alpha)[s]\uparrow$: we set $J^{X_e}(x_\alpha)[s]\downarrow= attempt_\alpha + 1$ with use σ, where $\sigma = X_{e,s}\restriction j_\alpha$. Set $r_{\alpha,attempt_\alpha+1} = \varphi_e(j_\alpha, s)$.

      ii. *$J^{X_e}(x_\alpha)[s]\downarrow$ and appears in $T_{i,x_\alpha}^{C_\tau}$*: set $F_\alpha = 2$, and open up an A-gap by letting $\delta_s(d) = attempt_\alpha + 1$. We also call this action *opening a k-gap*, where $k = attempt_\alpha + 1$.

      iii. *otherwise*: do nothing since we are waiting for $J^{X_e}(x_\alpha)$ to get traced.

(c) $F_\alpha = 2$: let s^- be the previous visit to α. Close the A-gap, and take the appropriate actions listed below. There are three possibilities.

i. $X_{e,s}\restriction j_\alpha = X_{e,s^-}\restriction j_\alpha$: (i.e. X_e recovers). We now have to abandon the current value of x_α. To do that, we first enumerate $u_\tau(x_\alpha, s)$ into C_τ to clear all relevant axioms, and then pick a fresh value for x_α. For all $x' \leq$ the newly chosen x_α for which no $h_\tau^{C_\tau}$-axioms currently apply, set $h_\tau^{C_\tau}(x')[s] \downarrow = region_\alpha$ with a fresh use $u_\tau(x', s)$.

Next, set $r_{\alpha,attempt_\alpha+1} = \varphi_e(j_\alpha, s)$, and set $R_{\alpha,attempt_\alpha+1} = X_{e,s}\restriction j_\alpha$. Reset $attempt_\alpha = 0$, pick a fresh value for j_α, and set $F_\alpha = 1$.

ii. $J^{X_e}(x_\alpha)[s]\downarrow$: (i.e. X_e goes back to a previous configuration). Pick a fresh x_α and adjust $h_\tau^{C_\tau}$ as in 3(c)(i) above. Extend $R_{\alpha,k} = X_{e,s}\restriction j_\alpha$, where $k = J^{X_e}(x_\alpha)[s]$. Also set $r_{\alpha,k'} = \varphi_e(j_\alpha, s)$ for all k' such that $k \leq k' \leq attempt_\alpha + 1$, since such a k'-gap has been recently opened, and the restraint has to be updated. Reset $attempt_\alpha = 0$, pick a fresh value for j_α, and set $F_\alpha = 1$.

iii. *otherwise*: set $r_{\alpha,attempt_\alpha+1} = \varphi_e(j_\alpha, s)$. Increase $attempt_\alpha$ by 1, and set $F_\alpha = 1$. Pick a fresh value for j_α.

In all cases other than 3(b)(ii), the outcome played is f, where all restraints $r_{\alpha,n}$ will be held.

This ends the definition for δ_s. At the end of stage s, initialize all nodes $\beta >_{left} \delta_s$. This completes the construction.

3.9.6 Verification

Define the true path of the construction TP, by the following: for all n, $TP\restriction n$ is visited infinitely often, and $\delta_t <_{left} TP\restriction n$ only finitely often. Note that TP exists: this is because if α is an active bottom node, then the only outcomes it can play when visited at a stage s must be one of $\{f, 1, 2, \cdots, region_\alpha[s]\}$, and $region_\alpha$ is reassigned only if α is initialized. Also, it is obvious that every node α on TP is initialized only finitely often, so we let $true(\alpha)$ be the least stage t such that α is visited at stage t, and α is never initialized after stage t.

Lemma 3.9.3. *A is noncomputable.*

Proof. Firstly observe that if β is a bottom node on TP, then it can increase $r_{\beta,n}$ at a stage s either under step 3(b)(i), or under step 3(c) of the construction. In the latter case it must be that some outcome $\leq_{left} n$ was played at the previous visit to β. In the former case, if $s > true(\beta)$, then nothing happens until the next A-gap is opened by β, where outcome n will be played. Thus each positive node α on the true path has to obey only a finite amount of restraint on A, and so A is noncomputable. $\square$

The next fact to establish is that for all e, requirement $\mathcal{N}_e$ is satisfied. Let τ be an $\mathcal{N}_e$-node on TP, and suppose that $\Phi_e^A = X_e$ is noncomputable. There must clearly be infinitely many τ-expansionary stages. Hence τ has true outcome ∞. We show that every τ-bottom node α on TP must have true outcome f:

Lemma 3.9.4. *Suppose α is a τ-bottom node on TP, with true outcome $o <_{left} f$. Then, X_e is computable (contrary to assumption).*

Proof. Let $s_0 = true(\alpha^\frown o)$. Firstly, observe that after stage s_0, we must always have $attempt_\alpha < o$. Suppose not - then at some stage when an o-gap is closed, we increase $attempt_\alpha$ to the value o under step 3(c)(iii) of the construction. Since outcome $o+1$ cannot be played after stage s_0, it follows that F_α stays forever at 1 and thus outcome o is never played again - a contradiction. Thus, $attempt_\alpha$ is always $< o$.

At each stage $s > s_0$ such that outcome o is played, we have an opening of an o-gap. At the next visit to α, the o-gap opened at stage s will be closed. Thus, we have $j_\alpha \to \infty$. Furthermore each o-gap can only be closed under step 3(c)(i) or (ii). Hence, there is some largest $n \leq o$ such that $|R_{\alpha,n}[t]| \to \infty$. We may thus assume that s_0 is large enough so that no $R_{\alpha,m}$ is reassigned after s_0, for all $m > n$.

Claim 3.9.5. Suppose that $s_1 > s_0$ is a stage in which $R_{\alpha,n}$ is assigned the value σ. Then, $\sigma \subset X_e$.

Proof of claim. Suppose s_1 and σ are as above. We show by an induction, that if an n-gap is opened at any stage $s_2 > s_1$, then we must have

(I1) $X_{e,s_2} \supset \sigma$, and

(I2) any axioms of the form $J^\rho(x_\alpha)[s_2] = k$ enumerated by stage s_2, must satisfy $k \leq n$ and $\rho \supset \sigma$.

This is enough to guarantee that $\sigma \subset X_e$ by (I1), since Φ_e^A is total. At stage s_1 when $R_{\alpha,n}$ is assigned, it must also be that a "cycle" is ended, i.e. $attempt_\alpha$ is set to 0. Thus when the next n-gap is opened, we must have $X_{e,t} \supset \sigma$, which proves the base case for (I1). As for (I2), note that $X_{e,s_1} \restriction |\sigma|$ is preserved from s_1 until the next n-gap is opened, and hence any computation set in the meantime must have use $\rho \supset \sigma$.

Now consider an arbitrary $s_2 > s_1$ such that an n-gap is opened at stage s_2, and assume (I1) and (I2) holds. Let $s_3 > s_2$ be the next time an n-gap is opened, our task is to show firstly that $\sigma \subset X_{e,s_3}$.

This current cycle must end at some stage t, where $s_2 < t < s_3$, with the closing of a k'-gap, for some $k' \geq n$ and $attempt_\alpha$ reset to 0. Consider the action taken when the cycle ends. Either step 3(c)(i) or (ii) applies to close the k'-gap. If 3(c)(i) applies, then $k' = n$ because $R_{\alpha,n+1}, R_{\alpha,n+2}, \cdots$ are not reassigned anymore. But then the restraint $r_{\alpha,n}$ is updated and protects $X_{e,t} \restriction |\sigma|$ until the next n-gap is opened at s_3, and hence $\sigma \subset X_{e,s_3}$. Suppose that 3(c)(ii) applies at stage t. Thus we return to a previous configuration, and $J^\rho(x_\alpha)[t] \downarrow = k$ where $\rho \subset X_{e,t}$. Clearly $k \leq n$ because again, $R_{\alpha,n+1}, R_{\alpha,n+2}, \cdots$ are not reassigned anymore. Hence $J^\rho(x_\alpha)[s_2] = J^\rho(x_\alpha)[t] = k$ must have been enumerated before stage s_2, and from (I2), we know that $\rho \supset \sigma$. The restraint $r_{\alpha,n}$ is updated at the end of the cycle at stage t, which preserves $X_{e,t} \supset \sigma$ until stage s_3. In either case we have $\sigma \subset X_{e,s_3}$.

Next, we have to show (I2). Note that any such computation has to be set at some stage between t and s_3, where the corresponding use ρ must be the current approximation of X_e. However after stage t, the segment $X_{e,t} \restriction |\sigma|$ is protected until s_3, so clearly $\rho \supset \sigma$. □

To compute $X_e(p)$, wait until the least stage $t > s_0$ such that $|R_{\alpha,n}[t]| > p$. Then, $R_{\alpha,n}(p)[t] = X_e(p)$. □

Lemma 3.9.6. *$h_\tau^{C_\tau}$ is total, nondecreasing and unbounded, and for each τ-bottom node α on TP, $h_\tau^{C_\tau}(\lim x_\alpha) \downarrow = \lim region_\alpha$.*

Proof. For each τ-bottom node α on TP, we have x_α and $region_\alpha$ eventually settles at some values x and r respectively. It is clear also that $h_\tau^{C_\tau}(x)\downarrow= r$, because after α sets the $h_\tau^{C_\tau}(x)$-axiom, no other τ-bottom node $\beta \subset \alpha$ can enumerate below $u_\tau(x)$ lest β opens some gap in future, which would cause α to be initialized. Also $\beta \not\supset \alpha$ and $\beta \not>_{left} \alpha$ because otherwise β will set its C_τ-use $u_\tau(x_\beta)$ after α does. Lastly $\beta \neq \alpha$ since x_α has settled.

No inconsistent h_τ-axioms are ever enumerated. $h_\tau^{C_\tau}$ is total because there are infinitely many active τ-bottom nodes along TP, by Lemma 3.9.4. It is nondecreasing because axioms set under 3(a) are done so with a fresh *region*-value, while in 3(c)(i) and (ii), old axioms are first cancelled before new ones replace them. $\square$

Lemma 3.9.7. *X_e is not hyper jump traceable.*

Proof. Let α be an $\mathcal{N}_{e,i}$-node on TP, and let $x = \lim x_\alpha$ and $r = \lim region_\alpha$. Let $a = \lim attempt_\alpha$, which also has to settle since only finitely many gaps are opened by α. We can easily see that $|T_{i,x}^{C_\tau}| \geq a$ because each time we increase $attempt_\alpha$ after $true(\alpha)$ we must have a new value appearing in $T_{i,x_\alpha}^{C_\tau}$, whose axiom will never be removed after it shows up (unless removed by α itself). Hence if $a = r$, then we are done, so suppose that $a < r$. After α increases $attempt_\alpha$ for the last time it will be put back to state $F_\alpha = 1$. It is clear that $J^{X_e}(x)\downarrow$, since $X_e \upharpoonright j_\alpha$ is forever preserved. It also follows that $J^{X_e}(x) \notin T_{i,x}^{C_\tau}$ since the value a has already been attained by $attempt_\alpha$. $\square$

Many questions regarding the class $\mathcal{H}$ remain open. For instance, it is not known if $\mathcal{H}$ forms an ideal. Since the relation SJT is not a true relativization, one cannot directly relativize the proof that the strongly jump traceable c.e. sets are closed under $\oplus$. A more direct approach is needed, and we expect that the box promotion methods in Cholak, Downey and Greenberg [CDG08] would be useful. It is also not known if every hyper jump traceable set is bounded by a c.e. one, or if there is a c.e. hyper jump traceable set which is low$_2$ cuppable.

Chapter 4

On very high degrees

Published in the Journal of Symbolic Logic (2008), **73(1)**, pp. 309-342.

4.1 Introduction

The motivation for this work comes from four sources :

1. The study of the relationship between measure and degree.
2. The study of algorithmic randomness, in particular that of Kolmogorov complexity.
3. Questions arising from the study of pseudojump operators.
4. To develop techniques used in the construction of a proper subclass of the uniformly almost everywhere dominating degrees.

We prove the following two results, where the motivation, notation and terminology will be discussed below.

Theorem 4.2.1. *There is a c.e. minimal pair of superhigh degree. That is, there are superhigh c.e. sets A and B, such that $\forall D(D \leq_T A$ and $D \leq_T B \Rightarrow D \equiv_T \emptyset)$.*

Theorem 4.3.1. *There is a cappable c.e. ultrahigh set. That is, there is an ultrahigh c.e. set A, and a noncomputable c.e. set B, such that $\forall D(D \leq_T A$ and $D \leq_T B \Rightarrow D \equiv_T \emptyset)$.*

Our first motivation comes from the study of the relationship between measure and degree. A is said to be almost everywhere (a.e.) dominating, if for almost all $X \in 2^\omega$, and all $g \leq_T X$, there is $f \leq_T A$ such that f dominates g. A is said to be uniformly almost everywhere (u.a.e.) dominating, if there is $f \leq_T A$, such that for almost all $X \in 2^\omega$, and all $g \leq_T X$, we have f dominating g. Kurtz [Kur81] showed that $\emptyset'$ was u.a.e. dominating. Motivated by Kurtz, Dobrinen and Simpson [DS04] were led to study the class of a.e. dominating and u.a.e. dominating reals. They conjectured that the a.e. dominating reals and the u.a.e. dominating reals coincide, and that the degrees of such reals were precisely those above $\mathbf{0'}$. In [CGM06], Cholak, Greenberg and Miller gave a direct construction of an incomplete c.e. set which is u.a.e. dominating, while Barmpalias and Montalbán [BM07] constructed a u.a.e. dominating degree which is half of a minimal pair. It turns out that the a.e. dominating degrees and the u.a.e. dominating degrees do coincide [KH07, KHMS06b], but such degrees are not always complete. However, it follows from a result of Martin [Mar66] that the u.a.e. dominating degrees resemble $\emptyset'$ in that they are all high. The intuition is that not only are they high, but they resemble $\emptyset'$ very strongly.

Simpson [Sim07] continued the theme of showing that the u.a.e. dominating reals resemble $\emptyset'$, by showing that every u.a.e. dominating degree is superhigh (i.e. $A' \equiv_{tt} \emptyset''$). In a fascinating connection between the theory of algorithmic randomness and effective measure theory, it was recently shown that u.a.e. dominating reals can be defined in terms of relativized Kolmogorov complexity. As we will see, this concerns the class of reals A such that $\emptyset'$ is K-trivial relative to A. Thus we see that the u.a.e. dominating reals arises quite naturally in two different ways.

The class of K-trivial reals was first introduced in [Sol75]. They are defined as those reals A such that the K-complexity of each initial segment of A is as low as it can be. That is, a real A is K-trivial, if

$$\exists c \forall n (K(A \upharpoonright n) \leq K(n) + c).$$

Every initial segment of A contains no more information than just its length. Such reals might seem very similar to the computable ones. Indeed, Chaitin [Cha76] showed that if C was used in place of K, the only C-trivial reals are the computable ones. Solovay however, was the first to construct a noncomputable K-trivial, showing that the property of K-triviality was different from being computable.

The K-trivials have aroused great interest in recent years, and are related to various other classes defined independently. How does all these relate to u.a.e. dominating? We say that a Δ^0_2 set A is almost complete (or $\emptyset'$-trivializing), if $\emptyset'$ is K-trivial relative to it, namely

$$\exists c \forall n (K^A(\emptyset' \upharpoonright n) \leq K^A(n) + c).$$

In upcoming work, Kjos-Hanssen, Miller and Solomon showed that for $A \leq_T \emptyset'$, A has u.a.e. dominating degree iff A is $\emptyset'$-trivializing. Thus we find that the natural class of u.a.e. dominating reals coincide with the $\emptyset'$-trivializing ones. Progress in understanding the relation of u.a.e. dominating reals to the degrees has been slow, and little is known [BM07, CGM06, Sim07]. We aim to contribute to the understanding of this class and its relationship with the Turing degrees.

In spite of the results of Cholak, Greenberg and Miller [CGM06], and Barmpalias and Montalbán [BM07], there is no known construction of a $\emptyset'$-trivializing real that combines with upper cone avoidance[1]. The only result related to cone avoidance is due to Nies and Shore who gave a direct construction of a c.e. $\emptyset'$-trivializing real A, avoiding the upper cone of a K-trivial B.

The connection between K-triviality and u.a.e. domination allows for another construction of a u.a.e. dominating real using the pseudojump technique of Jockusch and Shore. A pseudojump operator V is one that takes each set X to the set $V(X) := X \oplus W^X$ c.e. in and above X (for a fixed c.e. set W). We also say that A completes the pseudojump operator V, if $V(A) \equiv_T \emptyset'$. Jockusch and Shore [JS83] proved the pseudojump inversion theorem, which states that every pseudojump operator has a c.e. noncomputable completion. Nies showed that in fact every pseudojump operator has a ML-random completion. If we apply Nies' result to the construction of a K-trivial, we get a ML-random, $\emptyset'$-trivializing degree.

There are a number of general questions about pseudojump operations and their ability to combine with various degree-theoretic constraints. In [CDJL05], Coles, Downey, Jockusch and LaForte proved that there is a pseudojump operator V nontrivial over the c.e. sets (i.e. for all c.e. sets W, we have $V(W) >_T W$), such that V

[1]That is, given any noncomputable set B, there is a $\emptyset'$-trivializing real A such that $A \not\geq_T B$.

does not avoid upper cones. They asked the question if V can be strengthened to be nontrivial over all sets, as any natural operator arising from relativizing constructions of c.e. sets must be. They also asked if it is true that every operator has a cappable completion. We might hope that perhaps the construction of a K-trivial might give an operator which cannot avoid cones. In Theorem 4.3.1, we show that the operator arising from relativizing the construction of a strongly jump traceable (to be defined soon) has a cappable completion. The results of Barmpalias and Montalbán [BM07], and Cholak, Greenberg and Miller [CGM06] are immediate corollaries to our theorem.

The questions regarding completions of pseudojump operators, and the results of [BM07] and [CGM06] inspire us to study a proper subclass of the $\emptyset'$-trivializing reals. If we apply the pseudojump inversion theorem to the construction of a strongly jump traceable c.e. set, we get an incomplete c.e. set A, such that $\emptyset'$ is strongly jump traceable relative to it. Since $A \leq_T \emptyset'$, this is the same as $\emptyset' \in SJT(A)$.

Definition 4.1.1. A c.e. set A is said to be *ultrahigh* if $\emptyset' \in SJT(A)$.

Using the relativized version of the results in [CDG08], this new class of reals is seen to be a proper subclass of the c.e. $\emptyset'$-trivializing reals. We remark that the ultrahigh c.e. sets resemble the Halting problem very closely. We do not know if there is a characterization in terms of dominating functions, as in the case of the high c.e. degrees and the $\emptyset'$-trivializing c.e. degrees.

The proofs of Main Theorems 4.2.1 and 4.3.1 not only involve an analysis of the growth rates of the order functions, but also require a careful scheduling procedure which decides when numbers are allowed to be enumerated and when the lengths of agreement are allowed to rise. We believe this may be of independent technical interest.

In Section 4.2 we will construct a minimal pair of c.e. superhigh sets. Shore has also proved the same result in unpublished work, where he has a different way to handle the thickness requirements of the construction of a minimal pair of high c.e. sets. In the proof of Theorem 4.2.1, we will code Tot directly into the jump of the constructed set, such that the trace can be recovered in a tt way. We remark that both methods essentially run along the same lines; both methods will require keeping track of whether each minimal pair requirement is currently holding A-restraint, or holding B-restraint, and will involve coding the true path of the construction into the jump of the sets we are constructing. However, the presentation of the proof below was chosen, because it illustrates clearly the scheduling procedure of when lengths of agreements are allowed to rise - this is a crucial ingredient in the proof of Theorem 4.3.1, where we will combine the minimal pair requirements, with the requirements constructing an ultrahigh set.

This result says that while the questions surrounding minimal pairs of $\emptyset'$-trivializing reals and minimal pairs of ultrahigh reals remain open, their answers will depend crucially upon the growth rates of the order functions. In Section 4.3 we will construct a c.e. ultrahigh set which is half of a minimal pair.

Lastly, we mention that ultrahighness is a strong highness concept. A strong highness notion defines a class of sets which exhibit properties close to $\emptyset'$, and one usually obtains such a class by relativizing a lowness notion and then applying the pseudojump completion. For instance, the high c.e. sets are the sets A where $\emptyset'$ is low relative to A. As we have seen, the Δ^0_2 u.a.e.d. sets can also be expressed as the sets A where $\emptyset'$ is K-trivial (a lowness notion) relative to A. In Section 6.3 we will introduce various weak reducibilities obtained by the relativizations of lowness notions. The resulting highness dual notions will be discussed there.

4.2 A Minimal Pair of Superhigh Sets

In this section, we will proof the following theorem.

Theorem 4.2.1. *There is a c.e. minimal pair of superhigh degree. That is, there are superhigh c.e. sets A and B, such that $\forall D(D \leq_T A \ \wedge\ D \leq_T B \Rightarrow D \equiv_T \emptyset)$.*

4.2.1 Requirements

We build c.e. sets A and B satisfying the following requirements :

$$\begin{aligned}
\mathcal{N}_e &: \text{ If } \Phi_e(A) = \Phi_e(B) = h \text{ is total, then } h \text{ is computable,}\\
\mathcal{P}^A_e &: e \in Tot \Leftrightarrow A' \vDash \sigma_e \text{ (for some truth table } \sigma_e),\\
\mathcal{P}^B_e &: e \in Tot \Leftrightarrow B' \vDash \tau_e \text{ (for some truth table } \tau_e).
\end{aligned}$$

Here, we let Φ_e denote the e^{th} Turing reduction, and $Tot = \{e \in \mathbb{N} \mid q_e \text{ is total}\}$, where q_e is the e^{th} partial computable function of a single variable. We will ensure that the sequences $\{\sigma_e\}_{e\in\mathbb{N}}$ and $\{\tau_e\}_{e\in\mathbb{N}}$ are computable.

We adopt the convention of using uppercase Greek letters for functionals, and lowercase Greek letters for their use. The use of any convergent computation at a stage s is assumed to be bounded by s. We append $[s]$ to parameters, functionals or their use (e.g. $\Phi(A; x)[s]$) to describe their values at a stage s.

4.2.2 Strategy of a Single Requirement

There are two different types of requirements in this construction. The negative requirements $\mathcal{N}_e$ try to make A and B a minimal pair by keeping numbers out of A and B. The positive requirements $\mathcal{P}_e^A$ and $\mathcal{P}_e^B$ tries to make $Tot \leq_{tt} A'$ or $Tot \leq_{tt} B'$ by attempting to control the configuration of an initial segment of A' or B'. As e goes in and out of $Tot[s]$, we will need to put numbers into A or B to force changes in A' or B'. The main conflicts we need to consider, are when some $\mathcal{P}_e^A$ wants to make a change in A below the use of a computation that some negative requirements might want to preserve.

We will firstly remind the reader of the strategy used to satisfy a single negative (minimal pair) requirement $\mathcal{N}_e$: We will define a (partial) computable function h_e that computes the common value of $\Phi_e(A) = \Phi_e(B)$ (if they are equal). Whenever we observe $\Phi_e(A; x)[s] \downarrow = \Phi_e(B; x)[s] \downarrow$, we will set $h_e(x) = \Phi_e(A; x)[s] = \Phi_e(B; x)[s]$, and preserve *either* of $A \upharpoonright \varphi_e(A; x)[s]$ or $B \upharpoonright \varphi_e(B; x)[s]$. This allows us to have a period of time in which numbers are allowed to freely enter, say A (for the sake of the A-positive requirements) while we preserve $B \upharpoonright \varphi_e(B; x)[s]$. When the destroyed A-computation recovers at some stage $s' > s$, we have $\Phi_e(A; x)[s'] = \Phi_e(B; x)[s'] = \Phi_e(B; x)[s] = h_e(x)$ and so the common value at each recovery stage is forced to be the same (so that $h_e(x) = \Phi_e(A; x) = \Phi_e(B; x)$). At stage s' we could now allow numbers to enter B while restraining A to give the numbers a chance to enter B (for the sake of the B-positive requirements).

Before we go any further, we would like to highlight the difference between a requirement making A high, and a requirement making A superhigh. If we were just trying to make A high, we would attempt to define a reduction $Tot = \Gamma^{A'}$. For each

e, we have an associated $\gamma(e)$ use targeted at A', which we could control since we are building A. As e enters and leaves Tot (membership of Tot is a Σ_2^0/Π_2^0 fact), we have to put a stream of numbers into A to flip $A'(\gamma(e))$ back and forth. At times a negative requirement might block A, and prevent us from changing $A'(\gamma(e))$. When that happens we have to abandon the current value of $\gamma(e)$, and pick another one. The point is that as long as we limit the amount of negative restraint on A, this $\gamma(e)$ value eventually settles; that is all that really matters. In terms of the thickness requirements, this translates to the fact that we are allowed to miss finitely many numbers in the e^{th} column before we get to a stage where the requirement is never injured.

The reader will remember how this combines with the minimal pair requirements to produce a minimal pair of high c.e. sets - each high coding requirement experiences only a finite amount of A or B restraint. How different are our requirements in this case? We have to now code Tot into A' and B', while putting a computable bound on the use. In other words, we have to count in advance, for each superhigh coding requirement, the number of times a negative requirement of a stronger priority will block its actions. If we directly adopt the strategy above we would be in trouble, for we cannot count in advance how many times a minimal pair requirement will choose to increase its A or B restraint before it hits an x where $\Phi^A(x) \neq \Phi^B(x)$. Extra care has to be taken to fix this problem - we require a careful scheduling of when a minimal pair requirement is allowed to increase its length of agreement (and hence increase the restraint it imposes). How this is arranged, and the impact it has on the rest of the construction, will be explained later.

Consider a single A-positive requirement, $\mathcal{P}_e^A$. We describe briefly exactly how we intend to carry out the coding. We fix in advance the index of two Turing functionals $\eta_f < \eta_\infty$ which we are enumerating, and the membership $Tot(e)$ will eventually be decided by looking at the configuration $A'(\eta_f)A'(\eta_\infty)$. Suppose at stage s, $Tot(e)[s] = 0$. We would then put η_f into $A'[s]$ with use $u(\eta_f, s)$ by enumerating the axiom $\langle A_s \restriction 1 + u(\eta_f, s), \eta_f \rangle$ into Φ_{η_f}. Note that $u(\eta_f, s)$ is chosen larger than s, and therefore its entry into A later will only destroy those $\mathcal{N}$-computations which converge after stage s.

Suppose that after stage s, we never see an increase in $dom(q_e)$. Then, $Tot(e) = 0$

and $A'(\eta_f) = 1$. On the other hand if e enters $Tot[s']$ at a later stage $s' > s$, we could put $u(\eta_f, s)$ into A to take η_f out of A'. We would then put η_∞ into $A'[s']$ with use $u(\eta_\infty, s')$. If e enters and leaves Tot infinitely often, then $Tot(e) = 1$ and $A'(\eta_f)A'(\eta_\infty) = 01$, since once η_∞ is put into A', it is never removed (we are not considering any injury to $\mathcal{P}_e^A$ for the time being).

4.2.3 Interaction Among Strategies

We begin by considering a single A-positive requirement $\mathcal{P}$ working below a negative requirement $\mathcal{N}$. Suppose that at stage s, $\mathcal{P}$ puts η_f into A' with the use $u(\eta_f, s) > s$. At the next stage $s' > s$ where $dom(q)$ increases, $\mathcal{P}$ would want to take η_f out of A' and put η_∞ in. However, $u(\eta_f, s)$ might be less than the use of some computation $\Phi(A; x)[s']$ which had converged in the meantime (after stage s). If there had already been an enumeration into B below the use of $\Phi(B; x)$, then we would not be able to take η_f out of A' until the B-computation $\Phi(B; x)$ recovers. Unfortunately, if $\Phi(B; x)$ never recovers after stage s', we would not be able to make $A'(\eta_f) = 0$.

The solution to this is the usual - put the requirements on a Π_2-guessing tree (which is possible, since the predicate "$e \in Tot$" is a Π_2^0 fact). There are now two versions of the requirement $\mathcal{P}$: Firstly, $\mathcal{P}^\infty$ which guesses that the hypothesis in the $\mathcal{N}$-requirement is true and hence will only act at those stages where both sides $\Phi(A; x)[s] = \Phi(B; x)[s]$ are convergent. The other version is $\mathcal{P}^f$, which guesses that $\Phi(A) \neq \Phi(B)$ and will only get to act when one of $\Phi(A; x)[s]$ or $\Phi(B; x)[s]$ is allowed to be injured, for some $x \in dom(h)$.

Now, $\mathcal{P}^\infty$ would handle the functionals with indices $\eta_f^\infty < \eta_\infty^\infty$, while the other version $\mathcal{P}^f$ of $\mathcal{P}$ have the functionals $\eta_f^f < \eta_\infty^f$. We set things up so that $\eta_f^f < \eta_\infty^f < \eta_f^\infty < \eta_\infty^\infty$, and use A' on these four values as the truth table. At each $\mathcal{N}$-expansionary stage, where the $\mathcal{N}$-hypothesis has been further verified, $\mathcal{P}^\infty$ would run its (modified) basic strategy :

1. If $dom(q)$ has increased since $\mathcal{P}^\infty$'s last action, restore the configuration $A'(\eta_f^f)A'(\eta_\infty^f)A'(\eta_f^\infty)A'(\eta_\infty^\infty) = 0001$.

2. If $dom(q)$ has not increased since $\mathcal{P}^\infty$'s last action, we force the configuration $A'(\eta_f^f)A'(\eta_\infty^f)A'(\eta_f^\infty) = 001$.

The reader should note that either of the actions taken above would result in an enumeration of a historical use into A (because of the first two bits of the truth table), which might be below the use of Φ^A-computations. This is alright since $\mathcal{N}$ only needs to preserve one of the two sides of the newly converged computations.

At those stages which are not $\mathcal{N}$-expansionary, that is, we are waiting for one of the two computations $\Phi^A(\max dom(h))$ or $\Phi^B(\max dom(h))$ to recover, and at the same time preserving the other one, $\mathcal{P}^f$ would be able to have a chance to run its basic strategy:

1. If $dom(q)$ has increased since $\mathcal{P}^f$'s last action, force the configuration $A'(\eta^f_f)A'(\eta^f_\infty) =$ 01.

2. If $dom(q)$ has not increased since $\mathcal{P}^f$'s last action, set $A'(\eta^f_f) = 1$.

In 1. above, $\mathcal{P}^f$ would have to make an enumeration of a historical use $u(\eta^f_f)$ into A, which might be less than the A-restraint that $\mathcal{N}$ is currently putting up. This situation will arise from the following sequence of events: At the last $\mathcal{N}$-expansionary stage s, we had enumerated numbers into B instead of A. Thus even though $\mathcal{P}^\infty$ has had a chance to act at stage s, it did not do so in order to allow numbers into B. This is bad, for now $\mathcal{N}$ will increase its A restraint above $u(\eta^f_f)$. If $\mathcal{N}$ never sees a recovery on the B side, $\mathcal{P}^f$ would be stuck.

Note that if we were not required to make the reduction tt, we could simply let $\mathcal{P}^f$ move on to another index $\eta' > \eta^f_f$ and repeat. Unfortunately this is illegal in our case - we really have to make do with what we are given.

In order to overcome this difficulty, we will further split $\mathcal{P}^f$ into two versions, $\mathcal{P}^{f_A}$ and $\mathcal{P}^{f_B}$. Hence, the requirement $\mathcal{P}$ has now three different versions - $\mathcal{P}^\infty$ as above, and $\mathcal{P}^{f_A}, \mathcal{P}^{f_B}$ which respectively get to act at stages where $\mathcal{N}$ is holding A and B restraint. $\mathcal{P}^{f_A}$ now get to work with the indices η^A_f and η^A_∞, and $\mathcal{P}^{f_B}$ will work with the indices η^B_f and η^B_∞. We will code the totality of q into the configuration $A'(\eta^A_f)A'(\eta^A_\infty)A'(\eta^B_f)A'(\eta^B_\infty)A'(\eta^\infty_f)A'(\eta^\infty_\infty)$, a truth table of size 6. Depending on whether or not $e \in Tot$ when $\mathcal{P}^\infty, \mathcal{P}^{f_B}, \mathcal{P}^{f_A}$ are visited, each requirement above respectively tries to restore the configuration 000001 or $00001w$, $0001w'$ or $001w''$, and $01w'''$ or $1w''''$. For more details on the truth table, we refer the reader to Section 4.2.6.

To ensure that this strategy works, one will also need to carefully schedule when we allow the length of agreement for $\mathcal{N}$ to rise; more precisely we need to arrange when we extend $dom(h)$, which is the function computing the common value of both sides of the computations measured at $\mathcal{N}$.

Suppose $\mathcal{P}^{f_A}$ had already defined $u(\eta_f^A)$ when it acted at some stage where $\mathcal{N}$ was waiting for the recovery of $\Phi^B(x)$, where $x = \max dom(h)$. Suppose recovery occurs at the next $\mathcal{N}$-expansionary stage t, where we also have $\Phi^A(x+1)[t] \downarrow = \Phi^B(x+1)[t] \downarrow$, and we extend $dom(h)$ to include $x+1$. Although it is the case that $u(\eta_f^A, t) > \varphi(A; x)$, but $\Phi^A(x+1)$ can very well converge with a use larger than $u(\eta_f^A, t)$. This would be a problem if $\mathcal{P}^{f_A}$ wants to act before we have a chance to clear $u(\eta_f^A, t) < \varphi(A; x+1)[t]$.

However, we can see that at stage t, even though the length of agreement between Φ^A and Φ^B has increased, there is really no hurry to define $h(x+1)$ at stage t. The correctness and totality of h only matters if the $\mathcal{N}$-hypothesis is correct, in which case $\mathcal{P}^\infty$ would definitely get a chance to enumerate $u(\eta_f^A, t)$ into A at some time in the future. After $\mathcal{P}^\infty$ places $u(\eta_f^A, t)$ into A (and destroys the $\Phi^A(x+1)[t]$ computation at stage t), we could then wait until the next $\mathcal{N}$-expansionary stage t'' when $\Phi^A(x+1)[t''] \downarrow = \Phi^B(x+1)[t''] \downarrow$ again, and see if the situation at stage t occurs again at stage t'' (i.e. $u(\eta_f^A, t'') < \varphi(A; x+1)[t'']$). The point is that if the $\mathcal{N}$-hypothesis is true, there must be an $\mathcal{N}$-expansionary stage v such that $\Phi^A(x+1)[v] \downarrow = \Phi^B(x+1)[v] \downarrow$, and the computation $\Phi^A(x+1)[v]$ is *believable*, that is, $u(\eta_f^A, v) \not< \varphi(A; x+1)[v]$. We can then extend the definition of h when $\Phi^A(x+1)[v]$ (and $\Phi^B(x+1)[v]$) become believable. Controlling the definition of h in this manner allows us to ensure that we never accept an agreement in the computations $\Phi^A(x+1) = \Phi^B(x+1)$, until we are certain that any followers below both uses will only get enumerated during $\mathcal{N}$-expansionary stages.

The steps taken by a general X-positive requirement $\mathcal{P}$ below a number of negative requirements is essentially the same. If $\mathcal{P}$ is arranged to be of a lower priority than k many $\mathcal{N}$-requirements, then there will be 3^k many different versions of $\mathcal{P}$. Each version of $\mathcal{P}$ acts at stages where its guess about the states of the $\mathcal{N}$-requirements above are correct.

We remark that the reader should really think of the positive requirements as acting on “boxes”. Each of the 3^k many different versions of $\mathcal{P}$ will lie on the same level of the construction tree. Each of these nodes are assigned a different location of the jump $X'(\eta)$, i.e. a box. Thus, the truth table at this level is just the configuration of the row of 3^k boxes. Setting $X'(\eta) = 0$ is known as “emptying the box $X'(\eta)$”, achieved by making an enumeration into X, while setting $X'(\eta) = 1$ is known as “filling the box $X'(\eta)$”, achieved by enumerating a new axiom into that part of the jump. Thus a box has to be emptied before it can be filled with a new axiom.

When a version of $\mathcal{P}$ acts, it will empty all boxes assigned to the other versions of $\mathcal{P}$ to its right. It will either fill or empty its own box $X'(\eta)$ depending on the situation of Tot, thus setting up the truth table to look how it wants (see Section 4.2.6). This helps to give us a visual image of what is to come - as we will see, the main difficulty in Theorem 4.3.1, is that the ultrahigh coding requirements require us not only to place a bound on the number of boxes used, but all the positive requirements at the same level have to “share boxes”, i.e. each version of the same $\mathcal{P}$ no longer has the luxury of working on its own boxes; two or possibly more of the positive requirements at the same level have to work on the same box.

On a final note, we remark that in the proof, one actually codes the true path into A' and B', because the configuration of each truth table recorded in the jump during the construction not only specifies the membership of Tot, but also records which outcomes were played infinitely often during the construction. This is a crucial refinement of the priority tree used to produce a minimal pair of high sets - in addition to guessing the outcomes and doing the coding based on these guesses, we also have to code in the state of the $\mathcal{N}$ nodes (i.e. whether they are holding A or B restraints).

4.2.4 Construction Tree Layout

The construction takes place on the full binary tree. Nodes of length $|\alpha| = 4e + 1$ are assigned the requirement $\mathcal{P}_e^A$, while nodes of length $|\alpha| = 4e+3$ are assigned the requirement $\mathcal{P}_e^B$. The outcomes are labelled ∞ corresponding to a q_e-expansionary stage, and f corresponding to a non-expansionary stage for q_e. We place ∞ to the left of f.

Nodes α of length $|\alpha| = 2e$ are assigned the requirement $\mathcal{N}_e$. Instead of having two separate finite outcomes f_A and f_B, which are to be played when α is holding A

and B restraints respectively, we will identify both outcomes together, call it f. We will however, need to introduce a separate variable $state(\alpha)$ (to be defined below) to record whether α is currently holding A or B restraint. Again we arrange the infinite outcome ∞ (which stands for infinitely many $\mathcal{N}$-expansionary stages) to the left of the finite outcome f. This outcome stands for the fact that $\mathcal{N}$ settles on a final restraint, and where the final restraint is on (A or B) will be recorded in $state(\alpha)$. The construction tree grows downwards, i.e. we say that α *is above* β, if $\alpha \subset \beta$.

We say that $\alpha <_{left} \beta$, if there is some $i < \min\{|\alpha|, |\beta|\}$, such that $\alpha \restriction i = \beta \restriction i$, $\alpha(i) = \infty$, and $\beta(i) = f$. That is, α is to the left of β on the construction tree.

A node α is said to be a $\mathcal{Q}$-node, if it is assigned the requirement $\mathcal{Q}$. α is a negative node, if α is a $\mathcal{N}_e$-node for some e. The node α is an X-positive node if α is a $\mathcal{P}_e^X$-node for some e. At each stage s during the construction, we will define the approximation to the true path, δ_s of length s. A node α is visited at stage s, if $\delta_s \supset \alpha$.

4.2.5 Notations

The symbol X is to be used as a set variable, which will refer to either A or B. Let X^c be B if $X = A$, and vice versa. At each $\mathcal{N}_e$-node α, we will define a partial computable function h_α. If $\Phi_e^A = \Phi_e^B$ is total, we will ensure that h_α is total and that $h_\alpha = \Phi_e^A = \Phi_e^B$. The function h_α initially starts off as $\emptyset$, and from time to time we will increase the domain of h_α, denoted by $dom(h_\alpha)$. We will not do this at every α-expansionary stage however, and will hold back until the relevant computations become believable.

For each $\mathcal{N}_e$-node α we use the notation $R(\alpha, s)$ to record whether α is holding A or B restraint at non-expansionary stages s; $R(\alpha, s) = X$ indicates that α is holding X-restraint at stage s. At each stage s of the construction, numbers will be enumerated into either A or B but not both. We call s an *X-stage*, if numbers are enumerated into X during construction stage s.

If δ is a node on the construction tree, there may be several negative nodes (say for instance $\tau_0, \cdots, \tau_k$ such that $\tau_i^\frown f \subset \delta$, for $i \leq k$). At each stage s, each of the τ_i's might be trying to preserve an A or B computation. To keep a record of this fact, we define for each node δ and stage s, the string

$$state(\delta, s) = X_0 X_1 \cdots X_{|\delta|-1},$$

where for all $i < |\delta|$, the value $X_i \in \{A, B, \infty, f\}$ is determined by the following: If $\delta\upharpoonright i$ is a positive node, let $X_i = \delta(i)$. If $\delta\upharpoonright i$ is a negative node such that $\delta(i) = \infty$, then let $X_i = \infty$. Otherwise we let $X_i = R(\delta\upharpoonright i, s)$.

The introduction of the $state(\alpha)$ variable require us to define new orderings $<_A$ and $<_B$, used to determine priority amongst the different possible *state*-values. We let $\infty <_A f <_A B <_A A$ and also define $\infty <_B f <_B A <_B B$. We extend $<_A$ and $<_B$ lexicographically to orderings $<_A$ and $<_B$ on $\{A, B, \infty, f\}^{<\infty}$. The orderings are defined this way to be consistent with the style of the ordering $<_{left}$ of the nodes on the construction tree - $\gamma <_A \sigma$ means that γ is lexicographically left of σ, i.e. of a higher $<_A$ priority.

At each odd level n of the construction, we have *boxes* η^γ_∞ and η^γ_f for each $\gamma \in \{A, B, f, \infty\}^n$. These are actually Turing functionals which we define during the construction, and we let u^γ_x be the use of the box η^γ_x (henceforth, x is one of ∞ or f). Note that for each fixed n, the axioms for the different boxes η^γ_x form a uniformly c.e. set; hence we are able to compute an index for each box η^γ_x, which we also denote as η^γ_x. There are 2.4^n many boxes at level n, indexed by $\gamma \in \{A, B, f, \infty\}^n$. In addition, we call η^γ_x an *A-box* if $|\gamma| = 4e + 1$ for some e, and call it a *B-box* if $|\gamma| = 4e + 3$ for some e. To *empty* the X-box η^γ_x means that we enumerate the use u^γ_x into X. To *fill* the X-box η^γ_x at stage s with use u, means that we enumerate the axiom $\langle X_s\upharpoonright u + 1, 0\rangle$ into η^γ_x, and set the use $u^\gamma_x = u$.

At an A-stage s if α of length n is visited (say α is A-positive), it will work on the boxes $\eta^{state(\alpha)}_f, \eta^{state(\alpha)}_\infty$. By this, we mean that α will fill the $\eta^{state(\alpha)}_\infty$ box and clear the $\eta^{state(\alpha)}_f$ box if it is an α-expansionary stage, and fill the $\eta^{state(\alpha)}_f$ box if it is not an α-expansionary stage. In either case α will also clear all γ-boxes of a lower $<_A$-priority than (or to the right of) the current boxes, i.e. $\gamma >_A state(\alpha)$. The variable $state(\alpha)$ should be viewed as a pointer, which points at the two boxes where α is currently working on. The truth table at level n will eventually be specified by the α on the true path, as well as the final value of $state(\alpha)$.

At level n, $state(\alpha)$ will point at different boxes at various stages of the construction, depending on the state of the negative nodes above it. However, at no time will a $\mathit{state}(\alpha)$ point at a box which another α' at the same level has used before; this means that the boxes at level n "are not shared" amongst the different α. As we will see, in the proof of Theorem 4.3.1, different α will have to share boxes, and two α at the same level might have to point at the same box.

All variables and parameters retain their assigned values until the next assignation. If the context is clear we omit the stage number from the parameters.

Definition 4.2.2. For a negative node α, we say that a computation $\Phi^A(n)[s]$ with A-use w is *α-believable* at stage s, if

1. for all A-positive nodes β with $\beta^\frown\infty \subseteq \alpha$, the box $\eta_f^{\mathit{state}(\beta)}$ is either empty, or has use $> w$,
2. for all $\gamma >_A \mathit{state}(\alpha)^\frown B$, both A-boxes $\eta_f^\gamma, \eta_\infty^\gamma$ are empty or have use $> w$.

Condition 1. ensures that there are no pending changes below the use w due to incorrectly filled boxes of higher priority, while condition 2. ensures that the A-boxes of lower $<_A$-priority will not be blocked by an increased α-restraint - these boxes can always be cleared when α is visited at an A-stage. A similar definition follows for B-computations, with B and $<_B$ in place of A and $<_A$.

At each $\mathcal{N}_e$-node α, we define the *length of agreement* between the e^{th} reductions of A and B, based on believable computations:

$$l(\alpha,s) = \max\{y < s \mid (\forall x < y)\ (\Phi_e^A(x)[s]\downarrow = \Phi_e^B(x)[s]\downarrow \\ \text{are both } \alpha\text{-believable computations})\}.$$

We say that a stage s is α-expansionary, if α is visited at stage s, and $l(\alpha,s) \geq |\mathit{dom}(h_\alpha)[s]|$. That is, we require only that $l(\alpha,s)$ is equal to $\mathit{dom}(h_\alpha)[s]$ to be expansionary, and not strictly greater. This is to ensure that certain boxes can always be emptied, and that correct computations eventually become α-believable.

For a $\mathcal{P}_e^X$-node α, we define $l(\alpha,s)$, the *length of convergence of* q_e as:

$$l(\alpha,s) = \max\{y < s \mid (\forall x < y)\ (q_e(x)[s]\downarrow)\}.$$

In this case, we say that s is α-expansionary, if α is visited at stage s and $l(\alpha,s) > l(\alpha,s^-)$ where $s^- < s$ is the largest stage such that α was visited at stage s^-.

4.2.6 The Truth Table

We define the truth table σ_e; a similar definition holds for τ_e with B and $4e+3$ in place of A and $4e+1$. We first label all the $\{A, B, f, \infty\}$-sequences of length $4e+1$ by $\gamma_1 <_A \cdots <_A \gamma_n$. The truth table σ_e is then:

$A'(\eta_f^{\gamma_n})$	$A'(\eta_\infty^{\gamma_n})$	$A'(\eta_f^{\gamma_{n-1}})$	$A'(\eta_\infty^{\gamma_{n-1}})$	$\cdots$	$A'(\eta_f^{\gamma_1})$	$A'(\eta_\infty^{\gamma_1})$
0	1	?	?	$\cdots$	?	?
0	0	0	1	$\cdots$	?	?
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	0	0	0	$\cdots$	0	1

Thus, reading the truth table from left to right, we would see a string of zeroes, followed by a first entry with a 1. Everything that follows the first 1 is unhelpful garbage, and is accessed only when we visit the left of the true path, or when the states along the true path have not yet settled. To decide if $e \in Tot$ we see if this entry with the first 1 belongs to an η_f^γ box, or an η_∞^γ box. If it is the former then $e \in Tot$, otherwise $e \notin Tot$.

4.2.7 The Construction

Each stage s of the construction will either be an A-stage enumerating into A, or a B-stage enumerating into B. When we pick a *fresh* number at a stage s, we mean that we pick a number $> s$ and $>$ any number used or mentioned so far. At stage $s > 0$, do the following in the order given:

1. Inductively define the stage s approximation to the true path, δ_s of length s. Suppose that $\delta_s \upharpoonright d$ has been defined for $d < s$. If s is $\delta_s \upharpoonright d$-expansionary, let $\delta_s(d) = \infty$. Otherwise, let $\delta_s(d) = f$.

2. Declare that s is an X-stage, where X is to be decided below. If $s = 0$ or $s = 1$, let $X = A$. Otherwise, look for the longest $\beta \subset \delta_s$ such that β has been visited prior to stage s. Let $s^- < s$ be the most recent stage such that $\delta_{s^-} \supset \beta$. If s^- is an A-stage, let $X = B$, otherwise let $X = A$.

3. Take actions for each expansionary negative node on δ_s. For each negative $\mathcal{N}_e$-node β such that $\beta^\frown\infty \subseteq \delta_s$,
   (a) Set $R(\beta, s+1) = X^c$ (since the X-side of the β-computations will be destroyed at this stage). Here X^c denotes $\mathbb{N} - X$.
   (b) For each $n < l(\beta, s)$ we set $h_\beta(n) \downarrow = \Phi_e^A(n)[s]$, if not already defined.
4. Injure the negative nodes to the right of δ_s.
   (a) For each negative $\beta >_{left} \delta_s$, set $h_\beta = \emptyset$ and set $R(\beta, s+1) = A$.
5. Clear all X-boxes of lower $<_X$-priority than the current state.
   (a) For each $\gamma >_X state(\delta_s)$, we clear the X-boxes $\eta_f^\gamma, \eta_\infty^\gamma$.
6. Take actions for each X-positive node on δ_s.
   (a) For each X-positive node β such that $\beta^\frown\infty \subseteq \delta_s$, clear the box $\eta_f^{state(\beta)}$.
   (b) For each X-positive node $\beta \subset \delta_s$, fill the box $\eta_x^{state(\beta)}$ with fresh use $u > s$, where $x = \delta_s(|\beta|)$.

4.2.8 Verification

Let $TP = \liminf_s \delta_s$ be the true path of the construction under $<_{left}$, i.e. the leftmost path visited infinitely often. At each stage s, numbers (possibly none) are enumerated into either A or B but not both.

For any node α, the state of the nodes above α does not change unless the construction visits left of α. Hence if α is on the true path, then $state(\alpha)$ eventually settles at some value γ, pointing to the two boxes η_f^γ and η_∞^γ. We also point out that during the construction, any box filled at stage s is filled with a use $> s$. Hence any convergent computation $\Phi^X(n)[s] \downarrow$ at stage s, can only be destroyed later due to the enumeration of an X-box use, which was filled prior to stage s.

We begin with a preliminary lemma:

Lemma 4.2.3. *Suppose α is visited infinitely often. Then for each $X \in \{A, B\}$, there are infinitely many X-stages s such that $\delta_s \supset \alpha$.*

Proof. If α is visited at an A-stage s, then the next visit to α would be a B-stage, unless some $\beta \supset \alpha$ causes us to choose $X = A$ at step 2 of the construction. Once all such β are visited at least once after stage s, we would have to choose $X = B$ at step 2. □

Hence any α on the true path will be visited at infinitely many A-stages, as well as at infinitely many B-stages. This gives every positive node on the true path infinitely many chances to act.

Lemma 4.2.4. *Along the true path of construction, all the negative requirements are satisfied.*

Proof. Let α be an $\mathcal{N}_e$-node on true path, such that $\Phi_e^A = \Phi_e^B$ is total. Let s_0 be a stage large enough such that the construction never visits left of α after that. After s_0, h_α is never injured and so it is computable.

Claim 4.2.4.1. *In between α-expansionary stages, at least one side of the Φ_e-computations is preserved. That is, if $s_1 \geq s_0$ is an α-expansionary X-stage such that $X^c_{s_1} \restriction m \neq X^c_{s_2} \restriction m$ for some $s_2 > s_1$, where $m =$ the use of $\Phi_e^{X^c} \restriction dom(h_\alpha)[s_1]$, then there must be an α-expansionary stage u, such that $s_1 < u < s_2$.*

Proof of claim. Suppose on the contrary there are no such u. Since s is an X-stage, the number $k < m$ has to enter X^c at some stage t, where $s_1 < t < s_2$. By assumption that there is no such u, we have $\delta_t >_{left} \alpha^\frown\infty$. The number k has to be enumerated under step 5(a) or 6(a) of the construction at stage t. Since $R(\alpha, t) = X^c$ it follows that $state(\delta_t, t) >_{X^c} state(\alpha, s_1)^\frown X$. However, due to the α-believability of the $\Phi_e^{X^c}$-computations at stage s_1, neither of 5(a) nor 6(a) is possible. □

Claim 4.2.4.2. *There are infinitely many α-expansionary stages.*

Proof of claim. The idea is that we will show that any correct computation has to eventually become α-believable, using Lemma 4.2.3. Suppose there are only finitely many expansionary stages, and let s_1 be the last α-expansionary stage. Each time the construction visits left of α, we will set $h_\alpha = \emptyset$, so we certainly have $s_1 \geq s_0$ as well as $h_\alpha = h_{\alpha,s_1} \neq \emptyset$. Suppose that s_1 is an A-stage, a similar argument can be run if s_1 is a B-stage instead.

Let $s_2 > s_1$ be a stage such that $A_{s_2}\restriction m' = A\restriction m'$ where $m' =$ use of $\Phi_e^A\restriction dom(h_\alpha)$. By Claim 4.2.4.1, it follows that $B_{s_1}\restriction m = B\restriction m$ where $m =$ use of $\Phi_e^B\restriction dom(h_\alpha)[s_1]$. Since B has not changed below m, and $state(\alpha)$ never changes after stage s_1, it follows that the $\Phi^B \restriction m$ computations must be still believable at stage s_2. Hence, if s_2 is not expansionary, it must be because the computation $\Phi_e^A(j)[s_2]$ is not α-believable, for some $j \in dom(h_\alpha)$. This means that at stage s_2, one or both of the following is true:

1. There is an A-positive node β such that $\beta^\frown\infty \subseteq \alpha$, such that the box $\eta_f^{state(\beta)}$ has use $< m'$.

2. For some $\gamma >_A state(\alpha)^\frown B$, the box η_x^γ has use $< m'$.

For (2), note that at the last α-expansionary A-stage s_1, all such boxes are cleared. Hence the box η_x^γ has to be filled after stage s_1. However note that $R(\alpha)$ is set to B at stage s_1 and can only change if we visit left of α, or have an α-expansionary stage, neither of which is possible after stage s_1. So, it follows that $\gamma >_A state(\alpha)$ because only the nodes $\beta >_{left} \alpha$ can fill such a box. This case, together with (1) is impossible because α will eventually be visited again at an A-stage, and enumerate into A below m', a contradiction.

Thus, the Φ_e^A-computations at stage s_2 must also be α-believable, hence s_2 is α-expansionary. □

Note that Claim 4.2.4.2 is not enough to show that h_α is total, as $l(\alpha, s)$ need not be strictly increasing at expansionary stages. The fact that h_α is total follows from the fact that any correct computation must eventually become α-believable:

Claim 4.2.4.3. *h_α is total.*

Proof of claim. Let p be any number, and we want to show that $h_\alpha(p)$ eventually receives a definition after stage s_0. Consider a stage $s_3 > s_0$ such that $A_{s_3}\restriction m''$ and $B_{s_3}\restriction m''$ are correct up to m'', where m'' = larger of the two uses $\Phi_e^A\restriction p+1$ and $\Phi_e^B\restriction p+1$. Now s_3 has to be α-expansionary with $l(\alpha, s_3) > p$, otherwise there is some X-box with pending action below m''. Such a box will be cleared at the next α-expansionary X-stage, causing a change in $X\restriction m''$, a contradiction. □

Thus h_α is total, and computes correctly since one side of $\Phi_e^A = \Phi_e^B$ is always preserved between expansionary stages. $\square$

Lemma 4.2.5. *Along the true path of construction, all the positive requirements are satisfied.*

Proof. We consider the A-positive requirements, a similar argument follows for the B-positive requirements. Let α be a $\mathcal{P}_e^A$-node on the true path, and σ_e the truth table in Section 4.2.6. We want to show that q_e is total iff $A' \vDash \sigma_e$. Let s_0 be a stage after which we never visit left of α, also let $\gamma_0 = \mathit{state}(\alpha, s_0)$, the final state value.

It is clear that every A-box η_x^γ for every $\gamma >_A \gamma_0$ cannot be permanently defined, because we will clear such a box at each visit to α at an A-stage larger than s_0. Therefore we only need to show that

- q_e is total $\Rightarrow \eta_\infty^{\gamma_0}$ is permanently defined, while $\eta_f^{\gamma_0}$ is never permanently defined.
- q_e is not total $\Rightarrow \eta_f^{\gamma_0}$ is permanently defined.

We begin by assuming that q_e is total, hence there are infinitely many α-expansionary A-stages. At these stages, $\eta_f^{\gamma_0}$ has to be emptied, so it is never permanently defined. Suppose that $\eta_\infty^{\gamma_0}$ is filled at the α-expansionary A-stage $s_1 > s_0$, with use $u_\infty^{\gamma_0}$. We must show that it is never emptied after that; suppose on the contrary that some number $k \leq u_\infty^{\gamma_0}$ is enumerated at the A stage $t > s_1$. Suppose k is the use u_x^γ. The possibilities for γ are:

1. $\gamma \supseteq \gamma_0{}^\frown\infty$: Since uses are always chosen fresh, it follows that the box η_x^γ must have been filled at some α-expansionary A-stage $s_2 < s_1$, in which the box $\eta_\infty^{\gamma_0}$ must also be filled with use $u_\infty^{\gamma_0}[s_2] < u_x^\gamma[s_2]$, if it is not already occupied. Hence we have to empty the box $\eta_\infty^{\gamma_0}$ after s_2, before it can be filled at stage s_1, but this action would also empty the box η_x^γ, contrary to assumption.

2. $\gamma >_A \gamma_0{}^\frown\infty$: Such a box would be emptied at s_1, and even if they were filled later, the use would have to be larger than $u_\infty^{\gamma_0}$.

3. $\gamma <_A \gamma_0$: If this applies, then the construction has to visit left of α at stage t.

4. $\gamma \subseteq \gamma_0$: Then, k is the use of some box $\eta_f^{\mathit{state}(\beta)}[t]$ for some β with $\beta^\frown\infty \subseteq \alpha^\frown\infty$. As above, such a box would have to be emptied at stage s_1.

So, $\eta_{\infty}^{\gamma_0}$ has to remain permanently defined. If q_e is not total, a similar argument applies to show that $\eta_f^{\gamma_0}$ will be permanently defined. $\square$

4.3 A Cappable Ultrahigh Set

In this section, we construct an ultrahigh c.e. set A, which is half of a minimal pair:

Theorem 4.3.1. *There is a cappable c.e. ultrahigh set. That is, there is an ultrahigh c.e. set A, and a noncomputable c.e. set B, such that $\forall D(D \leq_T A \ \wedge \ D \leq_T B \Rightarrow D \equiv_T \emptyset)$.*

4.3.1 Requirements

We build the c.e. sets A and B with B coinfinite, satisfying the following requirements.

$$\begin{aligned}
\mathcal{N}_e &: \text{ If } \Phi_e^A = \Phi_e^B = h \text{ is total, then } h \text{ is computable,}\\
\mathcal{P}_e^A &: \text{ If } \Phi_e^A \text{ is an order, make } \emptyset' \ A\text{-jump traceable via } \Phi_e^A,\\
\mathcal{P}_e^B &: \text{ If } |W_e| = \infty, \text{ make } B \cap W_e \neq \emptyset.
\end{aligned}$$

Here, we let Φ_e denote the e^{th} Turing reduction, and $J^{\emptyset'}(k)$ denote the value of the universal $\emptyset'$-partial computable function $\Phi_k^{\emptyset'}(k)$. If Φ_e^A is an order, the requirement $\mathcal{P}_e^A$ will build an A-u.c.e. sequence $\{V_k^A\}_{k\in\mathbb{N}}$ such that for all k, $|V_k^A| \leq \Phi_e^A(k)$ and $J^{\emptyset'}(k) \in V_k^A$. To do this, $\mathcal{P}_e^A$ will divide the task into infinitely many substrategies, or modules. The k^{th} module will be responsible for building V_k^A. The requirement $\mathcal{P}_e^A$ will be split into infinitely many subrequirements $\mathcal{P}_{e,0}^A, \mathcal{P}_{e,1}^A, \cdots$. Each subrequirement will be responsible for ensuring the success of finitely many of these modules. The modules to which each subrequirement $\mathcal{P}_{e,i}^A$ is allocated will change from time to time, during the construction, and depends on the values of $\Phi_e^A[s]$.

4.3.2 The Atomic Strategy

The negative requirements $\mathcal{N}_e$ are as before - at each node α assigned a negative requirement $\mathcal{N}_e$, we will build a partial computable function h_α, that records the common value of $\Phi_e^A = \Phi_e^B$ observed at α-expansionary stages. We ensure that in between α-expansionary stages, at least one side of the Φ_e-computations is preserved.

We describe the strategy to make A ultrahigh. For simplicity, let us first consider the situation in which we want to trace $J^{\emptyset'}(k)$ with bound $f(k)$ for some computable order f. That is, we only need $\emptyset'$ to be jump traceable relative to A via a single bound f. We are building an A-uniformly c.e. sequence $\{V_k^A\}_{k\in\mathbb{N}}$ so that for all k, we have $J^{\emptyset'}(k) \in V_k^A$, and $|V_k^A| \leq f(k)$. In actual fact, we are enumerating an A-computable functional $\Psi^A(k,n)$ such that for every k, the range $\{\Psi^A(k,n) \mid n \in \mathbb{N}\} = V_k^A$. Therefore, if we enumerate a certain number p into V_k^A with use u, at a later stage we are allowed to change our mind. That is, we can *remove p from V_k^A*, at a cost of putting u into A. If we were instead building a plain u.c.e. sequence (without oracle A), we would be unable to change our mind in this way; any enumerated number has to increase the size of the trace permanently.

The freedom for us to change our mind could be exploited in the following way: Whenever the opponent plays $J^{\emptyset'}(k)[s]\downarrow$, we could trace the value $J^{\emptyset'}(k)[s]$ into V_k^A. When the opponent next changes the value of $J^{\emptyset'}(k)[t]$ and shows us a new value (note that he could in fact do this infinitely often, so that $J^{\emptyset'}(k)\uparrow$), we could remove the earlier trace and replace it with the new one.

Things are not so simple, of course, for if we could always do this, then $|V_k^A| = 1$ for all k, so that A is already Turing complete. We will have to remember that there are negative requirements which might be imposing A-restraint, at the time when our opponent shows us a new value to be traced. Therefore, at times we might be forced to leave a wrong trace in V_k^A, and continue with the strategy using a new location. We return to the "box" intuition used in Theorem 4.2.1. This time round, we think of each traced $J^{\emptyset'}(k)$ value as occupying a "box" location. Hence each V_k^A is allowed $f(k)$ many locations, or boxes which we may fill with potential candidates for $J^{\emptyset'}(k)$. If the positive strategy fills a box with an incorrect value, and at a later stage, the negative requirements above it decide to drop the A-restraint, we could go back and clear the box by removing the wrong trace. In this way we are able to re-use the box in future.

We will discuss the problems faced, and the solutions, for two typical scenarios in the construction : Firstly we describe how a negative requirement might survive the infinite positive action of some A-positive requirement living above it (of higher priority). Secondly, we will talk about how an A-positive requirement living below (of lower priority than) a negative requirement survives the A-restraint put on it. As in Theorem 4.2.1, the construction tree grows downwards.

Scenario 1 : *$\mathcal{N}$ living below $\mathcal{P}^A$.*

We consider a requirement $\mathcal{P}^A$, devoted to tracing $J^{\emptyset'}(k)$ into V_k^A. Since $\mathcal{P}^A$ might make infinitely many enumerations into A, we have to arrange for it to be equipped with two outcomes : Firstly, we need to have the infinitary outcome ∞, which represents the situation in which the opponent shows us infinitely many different values to be traced. The other outcome is finite f, which says that the opponent eventually makes up his mind on a certain value (i.e. $J^{\emptyset'}(k) \downarrow$). In this case, $\mathcal{P}^A$ would trace the final value and never remove it from the appropriate box.

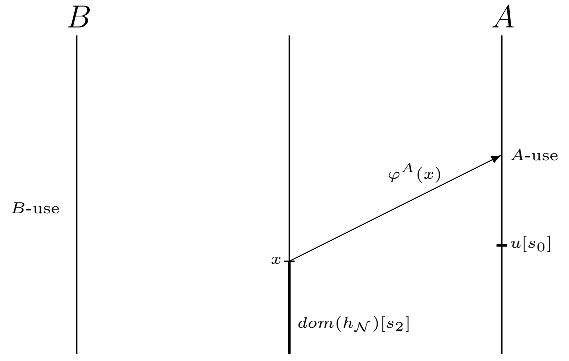
The purpose of equipping $\mathcal{P}^A$ with two outcomes is so that we can have two versions $\mathcal{N}$ and $\widehat{\mathcal{N}}$ of the same negative requirement below $\mathcal{P}^A$, guessing the outcomes ∞ and f respectively. Thus, $\mathcal{N}$ will never believe in a Φ^A-computation until $\mathcal{P}^A$ clears the relevant boxes with use below the Φ^A-computations. This is alright, because correct Φ^A-computations must eventually become believable. On the other hand $\widehat{\mathcal{N}}$ will always believe in new Φ^A-computations whenever they show up - it acts on the guess that $\mathcal{P}^A$ never needs to clear any of its boxes.

Scenario 2 : *$\mathcal{P}^A$ living below $\mathcal{N}$.*

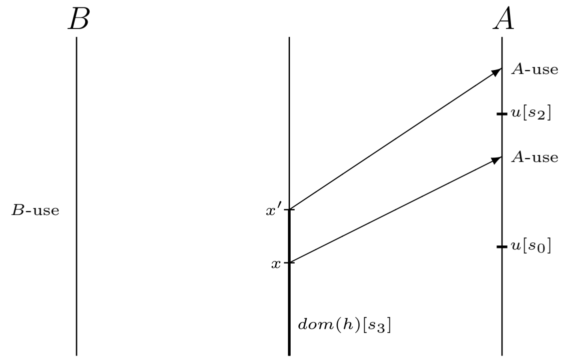
There are two versions of the positive requirement - $\mathcal{P}^A$ which guesses that there are infinitely many $\mathcal{N}$-expansionary stages, and $\widehat{\mathcal{P}^A}$ which guesses otherwise.

At stage s_0, suppose that $\widehat{\mathcal{P}^A}$ gets to act while $\mathcal{N}$ is waiting for the recovery of an A-computation $\Phi^A(x)$ for some $x \in dom(h)$. Suppose further that at stage s_0, $\widehat{\mathcal{P}^A}$ traces the current value $J^{\emptyset'}(k)[s_0]$ into V_k^A with use $u[s_0]$. At some later stage $s_1 > s_0$, we might get recovery of $\Phi^A(x)$ (with an A-use above $u[s_0]$). It might well be the case that at stage s_1, enumeration is made into B and not A (we have to allow for the B-restraint to be dropped infinitely often). When $\widehat{\mathcal{P}^A}$ gets to act later at stage $s_2 > s_1$, it would not be able to clear the box should the opponent challenge it to, since $\mathcal{N}$ has dropped B-restraint and is currently holding A-restraint above

$u[s_0]$. The following diagram describes the situation at stage s_2 :



What $\widehat{\mathcal{P}^A}$ needs to do now is to trace the new value into a second box, say with use $u[s_2] > \varphi^A(x)[s_2]$. Suppose the situation repeats again, namely, at a later stage the B-computation $\Phi^B(x)$ recover, and we have a length of agreement larger than some new $x' > x$, with $\varphi^A(x') > u[s_2]$. $\mathcal{N}$ then drops B restraint again and when $\widehat{\mathcal{P}^A}$ next gets to act at stage s_3, it finds itself in the following situation :



This would be bad, for $\widehat{\mathcal{P}^A}$ would have to move on and occupy yet another box. The solution to this, as in Theorem 4.2.1, is to delay extending $dom(h)$ until the Φ^A-computations become *believable.* When $\mathcal{N}$ first saw the length of agreement increase beyond x, it should not have expanded $dom(h)$ to include x', for $u[s_2]$ - the use of the second box - was actually below $\varphi^A(x')$. $\mathcal{N}$ could actually wait until the next expansionary A-stage, where both of the boxes can be cleared. The point is, once again, that if $\Phi^A(x') = \Phi^B(x')$, then eventually $\mathcal{N}$ would be able to place x' into $dom(h)$.

The reader might ask, what exactly is the difference between this (ultrahigh) strategy and the previous (superhigh) strategy, for the same method was used to decide when we allow $dom(h)$ to rise. The above discussion shows that the requirement with the two different versions $\mathcal{P}^A$ and $\widehat{\mathcal{P}^A}$ at that level, requires 2 boxes to run its strategy. Generally if a positive requirement lives on level m of the construction tree with 2^m different versions, it will need 2^m many boxes to handle all the possible combinations of A/B-restraint each negative requirement above it is currently holding. If the tracing order $f(n)$ has an exponential growth rate of $\sim 2^n$, the above strategy would work fine, because we could alternate the positive and negative strategies in the priority order $\mathcal{P}^A < \mathcal{N} < \mathcal{P}^A < \mathcal{N} < \cdots$. At the $2k^{th}$ level, the positive requirement will be responsible for tracing $J^{\emptyset'}(k)$, and have have 2^{2k} many different versions. Each of the 2^{2k} many different versions of the positive requirement will be allocated a different box, and each version will work on tracing the jump value into its own box when it is visited in the construction. So we will need altogether something like $2^{2k} \sim f(k)$ many boxes at level $2k$. The reader will note that this is actually the situation in Theorem 4.2.1 - the proof in Theorem 4.2.1 can be easily seen to be one which also produces a minimal pair A, B, such that $\emptyset'$ is jump traceable relative to both A and B via an order function of exponential growth rate.

The problem now of course, is that f is given to us - it is an arbitrarily slow-growing order. The main difference is that we cannot arrange our requirements as in the case when $f(n) \sim 2^n$. The cost of having an extra negative requirement above a particular $\mathcal{P}^A$ would have the effect of doubling the number of boxes it needs, because of the extra information we have to keep track of. We would have to arrange the requirements in the following order instead:

$$\underbrace{\mathcal{P}^A < \cdots < \mathcal{P}^A}_{\tilde{f}(1) \text{ many}} < \mathcal{P}^B < \mathcal{N}$$

$$< \underbrace{\mathcal{P}^A < \cdots < \mathcal{P}^A}_{\tilde{f}(2)-\tilde{f}(1) \text{ many}} < \mathcal{P}^B < \mathcal{N}$$

$$< \underbrace{\mathcal{P}^A < \cdots < \mathcal{P}^A}_{\tilde{f}(3)-\tilde{f}(2) \text{ many}} < \cdots ,$$

where $\tilde{f}(n) := \min\{k \mid f(k) > 2^n\}$. We will now discuss how to ensure that this arrangement succeeds - the trick involves allowing negative restraint to *transfer sideways*, and making positive requirements at the same level *share boxes*.

Let us consider the simple case when $\tilde{f}(1) = 0$, hence the first positive requirement appears at level 2 as shown below, with four different versions :

$$\begin{tikzcd}
 & & & \mathcal{P}^B \arrow[dll, no head] \arrow[drr, no head] & & & \\
 & \mathcal{N} \arrow[dl, no head] \arrow[dr, no head] & & & & \widehat{\mathcal{N}} \arrow[dl, no head] \arrow[dr, no head] & \\
\mathcal{P}^A & & \mathcal{P}^A & & \mathcal{P}^A & & \mathcal{P}^A
\end{tikzcd}$$

At this level, we will be allowed only two boxes for $\mathcal{P}^A$ (instead of four, as in the strategy we have discussed above). Hence each version of $\mathcal{P}^A$ no longer has the luxury of having its own box - all of its siblings have to share the two boxes available to them. If f grows very slowly, then there might be many levels (corresponding to each k such that $f(k) < 2$) below level 2 which are allowed two boxes. Box-sharing is therefore a necessary feature in this construction (compare it to the previous construction); we need it to resolve the complications introduced by an f which we do not have control of, as the box size is not always allowed to increase as we move down the construction tree.

The observation we need to make is that both versions $\mathcal{N}$ and $\widehat{\mathcal{N}}$ of the same negative requirement are actually trying to do the same thing, namely they are both measuring the same length of agreement. The only difference between the two, is that they would each believe in different computations, i.e. for instance the A-computation $\Phi^A(x)[s]$ converges, and $\widehat{\mathcal{N}}$ believes in it but not $\mathcal{N}$. Also assume that $\mathcal{N}$ and $\widehat{\mathcal{N}}$ are building computable functions h and $\tilde{h}$ respectively. The plan is to allow B-restraint to transfer sideways from the left ($\mathcal{N}$) to the right ($\widehat{\mathcal{N}}$).

Suppose $\mathcal{N}$ was visited, and the two versions of $\mathcal{P}^A$ below $\mathcal{N}$ fills up the two boxes allocated to level 2. When $\widehat{\mathcal{N}}$ is next visited, the versions of $\mathcal{P}^A$ below $\widehat{\mathcal{N}}$ would not have enough boxes left to use, if $\mathcal{N}$ is holding A-restraint (i.e. waiting for B-recovery). However, $\widehat{\mathcal{N}}$ can wait until all the B-computations in $dom(h)$ have recovered, before it decides to do anything. When this happens, $\widehat{\mathcal{N}}$ can restraint B on the use of these computations. Since $\mathcal{N}$ no longer cares what we put into A (as long as we hold the B-restraint and continue preserving the common value recorded by h), $\mathcal{N}$ could drop its A-restraint, and the versions of $\mathcal{P}^A$ below $\widehat{\mathcal{N}}$ can recycle the boxes used earlier. In general, a negative node α on the construction tree will wait for the lengths of agreement of all the nodes to its left and at the same level to recover, before α plays its expansionary stage. Once that happens, α would then drop A restraint to allow boxes below it to become accessible once again. The downside is of course, we would increase the B-restraint by a huge amount each time. This "transferred" B-restraint has to be obeyed even at α-expansionary B-stages, and will increase only at each visit to the left of α. Since the amount of transferred B-restraint is finite if α is on the true path, the B-positive requirements would be satisfied.

In this theorem we only make B noncomputable. Suppose we wanted to make B ultrahigh, using the method described above. The problem is that now, the B-positive requirements also has a number of pre-determined B-boxes at each level. In the strategy above, we had allowed the restraint to be transferred from the A side to the B side, so that certain A-boxes can become accessible again. However by doing so we might block some B-boxes and cause them to become unusable. Therefore, the strategy above does not immediately produce an ultrahigh, or even just a superhigh B. Indeed, it is not known if there is a minimal pair of A and B, such that $\emptyset'$ is jump traceable relative to both A and B, via an arbitrary order f.

4.3.3 Implementation on the tree

In the previous section we had described how to make a minimal pair A, B such that $\emptyset'$ is A-jump traceable via a single arbitrary computable order, and B is noncomputable. In general a positive requirement living on level k will only need to use 2^n many boxes, where n is the number of levels $< k$ of the construction tree devoted to negative requirements.

The above assumptions were too simple for two reasons: firstly in the actual construction, we have to deal with infinitely many functions Φ_e^A. Secondly, the values of Φ_e^A can only be approximated during the construction; however A will change during the construction. These changes will occur during the construction due to the other requirements, and we cannot hope to block changes in A to force Φ_e^A to be a computable order. Therefore it is possible that "blocks" get shuffled around; we have to constantly adjust the size of these blocks.

Suppose τ is assigned the requirement $\mathcal{P}_e^A$. τ would measure the length of convergence $l(\tau, s)$ of Φ_e^A, that is, the initial segment of $\Phi_e^A[s]$ such that $\Phi_e^A[s]\upharpoonright l(\tau, s)$ looks like it is an order. τ will divide its job amongst infinitely many sub-requirements

$\{\mathcal{P}^A_{e,i}\}_{i\in\mathbb{N}}$, which are all spread out at different levels below τ. Let

$$m^\tau_i = |j \in \mathbb{N} : \tau < \mathcal{N}_j < \mathcal{P}^A_{e,i}|,$$

which is the number of levels j of the construction tree devoted to a negative requirement, between τ and its i^{th} sub-requirement.

Suppose α is a node on the construction tree assigned the requirement $\mathcal{P}^A_{e,i}$. Instead of tracing $J^{\emptyset'}(k)$ for a single k at level $|\alpha|$, we let α (and all the other nodes on the same level) handle several of these k-modules, for all $k \in L(|\alpha|, s)$, such that

$$L(|\alpha|, s) := \{k < l(\tau, s) \mid 2^{m^\tau_i} < \Phi^A_e(k)[s] \leq 2^{m^\tau_{i+1}}\}.$$

We call each of these groups of modules a *block*. The implementation of this on the tree would be difficult if we had to dynamically change the assignment of requirements to the nodes extending τ. Instead, we will choose a presentation which fixes the labels on the construction tree, albeit at the cost of working with an ω-branching construction tree. To wit, the entire block will be assigned to a single node α. Suppose that $k_0, \cdots, k_m$ are the modules in the same block. Then α will be responsible for tracing all of $\{k_i\}^{\emptyset'}(k_i)$ for $i \leq m$. α will require two outcomes (k, ∞) and (k, f) for each k-module it is looking after, hence requiring altogether $2^{|L(|\alpha|,s)|}$ many outcomes, which we could fix in advance if not for the fact that $L(|\alpha|, s)$ itself might change as the construction proceeds.

When τ first detects that the length of convergence is long enough so that $\Phi^A_e(l(\tau, s))[s] > 2^{m^\tau_{i+1}}$, it will make all the nodes at level $|\alpha|$ start their atomic strategies. These nodes will start the process of tracing all the $J^{\emptyset'}(k)$ for $k \in L(|\alpha|, s)$. Note that all the traces will be made with A-use larger than s. In future if $L(|\alpha|, t) \neq L(|\alpha|, s)$, it must be the case that there is a change in A below s. Hence the next time τ sets its i^{th} sub-requirement to work with the new $L(|\alpha|, t)$, all of the earlier traces (which we no longer want) would have been automatically removed; no intervention by τ is needed for this. It might also be possible that $L(|\alpha|)$ changes infinitely often (as in the case when Φ^A_e is not an order), so we will have to arrange a leftmost outcome 0 for α. Each time before α can start work with a new set of τ-modules $L(|\alpha|)$, we will make α play outcome 0 once. This ensures that the true path is well-defined, in the case when α never receives a permanent set of instructions.

We need to equip α with many outcomes; in fact we need outcomes which describe whether or not $\{k_0\}^{\emptyset'}(k_0), \cdots, \{k_m\}^{\emptyset'}(k_m)$ are currently believed to be convergent. Thus, the outcomes of α will be finite strings of length $m+1$ with alphabeth $\{\infty, f\}$. We want the outcome $x_0 \cdots x_m$ to be the true outcome of α if and only if for every $i \leq m$, $\{k_i\}^{\emptyset'}(k_i) \downarrow \Leftrightarrow x_i = f$. Suppose that $\{k_0\}^{\emptyset'}(k_0) \uparrow$ and $\{k_1\}^{\emptyset'}(k_1) \uparrow$. Then, we want to arrange for $\infty\infty$ to be played infinitely often. If we simply play the natural outcome which codes the current state of convergence of $\{k_0\}^{\emptyset'}(k_0)[s], \cdots, \{k_m\}^{\emptyset'}(k_m)[s]$, then the trouble will be that $\{k_0\}^{\emptyset'}(k_0)$ and $\{k_1\}^{\emptyset'}(k_1)$ take turns to show us a new value. Then, we will cycle through the outcomes $f\infty$ and ∞f, when in fact we really want to record this as the outcome $\infty\infty$.

In order to reflect the outcomes correctly, at each α-stage, we will play the outcome in the following way. We want to make it as difficult as possible to output outcome f for any of the k_i's. After all if it is truly the case that $\{k_i\}^{\emptyset'}(k_i) \downarrow$, we can afford to delay for as long as we want before playing an outcome f for k_i (and hence begin the tracing of the correct $\{k_i\}^{\emptyset'}(k_i)$). Therefore at an α-stage s, we only output outcome f for k_i if $\{k_i\}^{\emptyset'}(k_i) \downarrow [s]$ has been stable for a long time (since some stage t), and for every other k_j such that $\{k_j\}^{\emptyset'}(k_j)[t] \downarrow$ and is different from $\{k_j\}^{\emptyset'}(k_j)[s]$, we would have output ∞ for all of these k_j's before s. It is clear that this prevents the situation above where we cycle through $f\infty$ and ∞f, and furthermore if $\{k_i\}^{\emptyset'}(k_i)$ truly does converge, then we will eventually always output outcome f for k_i.

4.3.4 Global Considerations

Suppose V_k^A was allowed trace locations denoted by $t_k(0), t_k(1), \cdots$, at level l. We had to allow for different nodes on level l access to the same trace location $t_k(m)$. This creates a curious situation when considering the interaction between two such positive levels $l < l'$. This is best illustrated by an example: take a positive node α on level l, and let $\beta_0 \supset \alpha^\frown\infty$ and $\beta_1 \supset \alpha^\frown f$ both be positive nodes on level l'. At some stage s, we might visit β_0, which decides to play outcome f. We will then make a trace $t_{k'}(m')$ for some m'. Later on, we might then visit α, where we also make a trace $t_k(m)$. Note that the use for $t_k(m) >$ use for $t_{k'}(m')$, since the latter was traced earlier. If finally β_1 is visited and sees a new $\{k'\}^{\emptyset'}(k')$-value, it will (and indeed must) record this new value at the same trace location $t_{k'}(m')$, because we really do not want to use a new valuable trace location simply because we were blocked by another ultrahigh strategy. Hence, β_1 would have to clear $t_{k'}(m')$ by enumerating the use, which in turn clears the location $t_k(m)$ as well. This creates the situation where a lower priority requirement β_1 takes an action which "injures" a higher priority requirement α. Once again we compare this with the case of a high or superhigh construction; this situation described does not happen there because at stage t when β_1 decides to trace the new $\{k'\}^{\emptyset'}(k')$-value, it will simply use another trace location to record this value (instead of clearing the old one).

We note that this situation is easily resolved by the tree mechanism, and in fact poses no real problem at all. The above situation of β_1 injuring α may repeat infinitely often, but *only if α has infinitely many expansionary stages.* In this case, α gets its trace cleared infinitely often by β_1, but it would not matter since $\{k\}^{\emptyset'}(k) \uparrow$. On the other hand if $\{k\}^{\emptyset'}(k) \downarrow$, then there will only be finitely many α-expansionary stages. Eventually at one of the $\alpha^\frown f$-stages, we would make a $t_k(m)$ definition before any $t_{k'}(m')$ definition is made, after which the action of β_1 will no longer affect α.

Such an arrangement might also affect nodes of higher priority, but which are further down the construction tree. Suppose now we consider α_0 to the left of α_1, which are both on the same level l. Suppose we visit $\alpha_0{}^\frown f$ and we make a trace $t_k(m)$. If later on we visit α_1 but we see a new $\{k\}^{\emptyset'}(k)$-value, we will need to clear the same trace $t_k(m)$ with a use set by α_0 earlier. This will cause nodes extending $\alpha_0{}^\frown f$ to be injured; again this is due to a lower priority α_1. In particular, this might cause some negative node working on level $j > l$ to be initialized. In this case, when we next visit α_0, we will allow $\alpha_0{}^\frown\infty$ to be visited once (regardless of the current $\{k\}^{\emptyset'}(k)$-status). This gives the version of the negative node on level j extending $\alpha^\frown\infty$ a chance to act.

An alternative presentation of the proof would have been the use of a pinball machine; indeed a pinball machine eliminates the above situation where lower priority nodes injure higher priority ones. However there are advantages of the tree method over the pinball machine in this case. This will be made clear in the following section. Basically the tree can handle the dynamic variations in "block sizes" easily, while in a pinball machine we would have to dynamically generate or modify the pinball machine when the distance between the gates (negative requirements) increases.

4.3.5 Construction Tree Layout

The construction takes place on a subtree of $\omega^{<\omega}$. Nodes of length $3e$ are assigned the requirement $\mathcal{N}_e$ with the outcomes ∞ for infinitely many expansionary stages, and f for finitely many expansionary stages, with $\infty <_{left} f$.

Nodes of length $3\langle e, i\rangle + 1$ are assigned the requirement $\mathcal{P}_e^A$ if $i = 0$, and the subrequirement $\mathcal{P}_{e,i-1}^A$ if $i > 0$. The possible outcomes of $\mathcal{P}_e^A$ are $\infty <_{left} f$, which stands respectively for infinite (finite) action taken. The outcomes of $\mathcal{P}_{e,i}^A$ are $0 <_{left} 1 <_{left} 2 <_{left} \cdots$. Outcome 0 to the extreme left means no action, i.e. the atomic strategies have not started. The other outcomes each represent the code number of a finite sequence of pairs $(n_0, x_0), (n_1, x_1), \cdots, (n_j, x_j)$ where the n_i's are distinct natural numbers, and $x_i \in \{\infty, f\}$, in some effective coding $\langle\cdot\rangle$ with range $\mathbb{N} \setminus \{0\}$. The important point here is to ensure that if σ and η are two such sequences, then $\langle \sigma, (n, \infty), \eta\rangle < \langle \sigma, (n, f), \eta\rangle$.

We say that m specifies the pair (k, x), if m is the code number of a sequence of pairs in which (k, x) appears. An outcome specifying (k, ∞) is played when α is charged with the task of tracing $J^{\emptyset'}(k)$, and a new value has just been shown by the opponent. An outcome specifying (k, f) on the other hand, will be played when α is assigned to trace $J^{\emptyset'}(k)$, but no clearing of boxes are pending.

Finally, nodes of length $3e + 2$ are assigned the requirement $\mathcal{P}_e^B$, with outcome s for success (in meeting the simplicity requirement), placed to the left of w, for waiting.

Let $\alpha <_{left} \beta$ denote that node α is strictly to the left of node β, extended lexicographically from the ordering among the outcomes. We say that α is a *$\mathcal{Q}$-node* if α is assigned the requirement $\mathcal{Q}$. A *negative node* is an $\mathcal{N}_e$-node for some e. We say that α is a *mother node*, if α is a $\mathcal{P}_e^A$-node for some e, and that α is an *A-positive* node if α is a $\mathcal{P}_{e,i}^A$-node for some e, i. A *B-positive* node is a $\mathcal{P}_e^B$-node for some e.

A node τ is said to be the *mother* node of α, if $\tau \subset \alpha$, and τ is a $\mathcal{P}_e^A$-node and α is a $\mathcal{P}_{e,i}^A$ for some e, i. In this case we also call α an *i-daughter* node of τ. If α is an A-positive node, then $\tau(\alpha)$ denotes its unique mother node. α and β are *sibling* nodes, if $\tau(\alpha) = \tau(\beta)$, and $|\alpha| = |\beta|$.

If α is an A-positive node, we let $Left(\alpha)$ denote all the sibling nodes of α, strictly to the left of α. If α is any other type of nodes, $Left(\alpha)$ simply denotes the set of all nodes $\beta <_{left} \alpha$ such that $|\beta| = |\alpha|$. We let *Right*(α) be defined similarly with the inequality reversed.

4.3.6 Notations

Suppose that τ is a $\mathcal{P}_e^A$-node. The k^{th} module of τ, which wants to trace $J^{\emptyset'}(k)$, is also called the (τ, k)*-module*. We divide a cofinite segment of $\mathbb{N}$ into infinitely many partitions $\{M_i^\tau\}_{i\in\mathbb{N}}$, where

$$M_i^\tau = \{x : 2^{\langle e,i+1\rangle - \langle e,0\rangle} < x \leq 2^{\langle e,i+2\rangle - \langle e,0\rangle}\}.$$

Basically $\langle e, i+1\rangle - \langle e, 0\rangle = m_i^\tau$, i.e. the number of levels j between τ and its i^{th} subrequirement containing negative nodes. An i-daughter node α of τ will be allowed $2^{m_i^\tau}$ many boxes at its level. Hence α will be entrusted with the task of ensuring the success of the (τ, k)-module for every k such that $\Phi_e^A(k) \in M_i^\tau$.

τ's job is to coordinate the actions of all of its daughter nodes, and build the A-u.c.e. sequence $\{V_k^\tau\}_{k\in\mathbb{N}}$ (note that oracle A is suppressed from the notation V_k^τ). In actual fact we are constructing, at the node τ, the Turing functional $\Psi^A(k, n)$ so that for every k, *Range*$(\Psi^A(k, \cdot)) = V_k^\tau$. We refer to each value $\Psi^A(k, n)$ as a *box*, which we will fill with a number. Fixing an effective coding $\langle\cdot\rangle$ of all finite $\{A, B\}$-sequences, we will call $\Psi^X(k, \langle\sigma\rangle)$ a *σ-box of V_k^τ*. That is, we think of each set V_k^τ as a row of boxes, where each box is labelled by a finite string $\sigma \in \{A, B\}^{<\omega}$. The purpose of this definition, is that during the period of time where α (and all its sibling nodes at the same level) is running the (τ, k)-module, it will be allowed to place numbers in, as well as take numbers out of the σ-boxes of V_k^τ, for every $\sigma \in \{A, B\}^{<\omega}$ of length m_i^τ. Hence at level $|\alpha|$ we have access to $2^{m_i^\tau}$ many boxes.

At a particular stage s, when we say that we *fill a σ-box of V_k^τ with use u*, we mean that we enumerate the axiom $\langle\langle k, \langle\sigma\rangle\rangle, y, A_s\upharpoonright u + 1\rangle$ into Ψ. The s^{th} stage use of the computation $\Psi^A(k, n)[s]$ is denoted by $u_{k,n}^\tau[s]$. To *clear*, or to *empty the σ-box of V_k^τ at a stage s*, means that we enumerate $u_{k,\langle\sigma\rangle}^\tau[s]$ into A. As usual we associate finite strings with their code numbers, and write $u_{k,\sigma}^\tau$ in place of $u_{k,\langle\sigma\rangle}^\tau$. During a stage s of the construction in which we enumerate into A, if an A-positive node α is visited and we filled some box for the sake of α, we will say that the box is *filled by* α at stage s. This remains true until the box is next emptied.

Define the *length of convergence* for Φ_e^A at stage s, to be

$$l(e,s) = \max\{y < s \mid (\forall x \leq y)\ \Phi_e^A(x)[s] \downarrow \geq \Phi_e^A(x-1)[s]\}.$$

Since τ is measuring $l(e,s)$, we also write $l(\tau,s)$ in place of $l(e,s)$. A stage s is *τ-expansionary,* if either $s = 0$, or else τ is visited at stage s of the construction, and $l(\tau,s) > l(\tau,s^-)$ where $s^- < s$ is the largest τ-expansionary stage smaller than s.

α has a list of instructions at stage s, defined as:

$$L(\alpha,s) = \{k < l(\tau,s) \mid \Phi_e^A(k)[s] \in M_i^\tau\},$$

which is a list of τ-modules that α has to run. This list will change as the construction proceeds, and when more of Φ_e^A is revealed. All sibling nodes of α will have the same list of instructions at all times, so they will all run the same modules, until τ says otherwise.

We will say that α *is active* at stage s, if τ allows α to start running the modules in the instruction list $L(\alpha)$. That is, α is active at s, if $\alpha \supset \tau^\frown\infty$ and we have the following :

- (AC.1): $\Phi_e^A(l(\tau))[s] \notin M_i^\tau$.

  [Hence we do not expect new numbers to appear in the instruction list $L(\alpha)$, unless there is an A-change.]

- (AC.2): $L(\alpha,s) \neq \emptyset$.

  [Otherwise α has nothing to do.]

- (AC.3): There is some largest $t < s$ where α is visited at both stages t and s, $L(\alpha,t) = L(\alpha,s)$, and for every $k \leq 1 + \max L(\alpha,s)$, the computation $\Phi_e^A(k)[s]$ has persisted for at least two visits to α (i.e. $A\restriction\varphi_e(A;k)[t] = A\restriction\varphi_e(A;k)[s]$).

  [To ensure that the true path is well-defined, that is, we ensure that α switches from being active to inactive each time there is a change in instructions.]

We also say that α is *permanently active* at stage s, if it is active at every visit to α after stage s, i.e. it never gets interrupted in running the modules.

Let β be an $\mathcal{N}_e$-node. At β we define a partial computable function h_β to record the common value of $\Phi_e^A = \Phi_e^B$. As in the previous proof, we do not increase $dom(h_\beta)$ at all β-expansionary stages, instead we hold back until an appropriate time. We also let $R(\beta, s) = X$ if β is holding X-restraint at stage s. Each stage s of the construction is either an A-stage where numbers are enumerated into A, or a B-stage in which numbers are enumerated into B.

Suppose τ is a mother node, and $\delta \supset \tau$ is any node. Let $\beta_0 \subset \cdots \subset \beta_{k-1} \subset \delta$ be precisely all the negative nodes lying on the path between τ and δ. As in the previous proof, we need to record at each stage s, whether each of these β_i is holding A or B restraint. We are allowed less storage space now - we will only record the outcomes of these negative nodes: let

$$state(\tau, \delta, s) = X_0 X_1 \cdots X_{k-1},$$

where for all $i < k$, the value $X_i \in \{A, B\}$ is determined by the following: if $\beta_i^\frown\infty \subseteq \delta$ then let $X_i = B$, otherwise we let $X_i = R(\beta_i, s)$.

If δ is a daughter node of τ, we abbreviate $state(\tau, \delta, s)$ by $state(\delta, s)$. If δ is a $\mathcal{P}_{e,i}^A$-node, then $|state(\delta, s)| = m_i^\tau$. As in the previous construction, $state(\delta, s)$ functions as a pointer telling δ to run all of its modules using the $state(\delta, s)$-box. It might be that several of δ's sibling nodes are also pointing simultaneously at the same box - unlike in the previous proof - and we have to ensure that δ's progress will not be undone by another of its sibling node. It also follows trivially from the definition that if $\delta_1 \subseteq \delta_2$, then $state(\tau, \delta_1) \subseteq state(\tau, \delta_2)$.

We fix a partial ordering $<_A$ on the set of all finite $\{A, B\}$-sequences, which will be used to determine priority amongst the different $state$ values. Let $\rho <_A \sigma$, iff there is some least $i < \min\{|\rho|, |\sigma|\}$, such that $\rho \restriction i = \sigma \restriction i$, and $\rho(i) = B \neq \sigma(i)$.

An A-positive node α will start work only if it is active. The variable $state(\alpha)$ points at a particular box in the k^{th} row of boxes (for each k in the instruction list $L(\alpha)$). α will fill the box in the k^{th} row each time it plays an outcome specifying (k, f), and will clear the box in the k^{th} row each time the outcome specifies (k, ∞). As the situation in the negative nodes above α changes, α may move on to other boxes in the same row; however α only has access to $2^{m_i^\tau}$ boxes in each row.

All sibling nodes on the same level will share boxes; all of them will have access to all the boxes available to that level. Because of this, and because of the transfer of restraints, we might actually return to a box of higher $<_A$-priority, as we move from the left to the right of the construction tree. We have to be a bit careful with notations, in particular when considering α-believability of A-computations. There are now more cases to consider:

Definition 4.3.2. For a negative node α, we say that a computation $\Phi^A(n)[s]$ with A-use w is *α-believable* at stage s, if

1. for all A-positive nodes $\beta \subset \alpha$, and each k such that $\alpha(|\beta|)$ specifies (k, ∞), the $state(\beta, s)$-box of $V_k^{\tau(\beta)}$ is either empty, or has use $> w$,
2. for all A-positive nodes β which has been visited prior to stage s, such that $\tau(\beta)^\frown\infty \subset \beta^\frown 0 \subseteq \alpha$, and all $k \in L(\beta, s)$, the $state(\beta, s)$-box of $V_k^{\tau(\beta)}$ is either empty, or has use $> w$,
3. for all mother nodes $\tau \subset \alpha$, all strings $\gamma >_A state(\tau, \alpha, s)^\frown B$ and all x, the γ-box of V_x^τ is either empty, or has use $> w$,
4. for all A-positive nodes $\beta \subset \alpha$, and all A-positive nodes $\sigma >_{left} \beta$, with $\tau(\sigma) \subset \alpha$, and all strings $\gamma \supseteq state(\tau(\sigma), \beta, s)$, and all x, if the γ-box of $V_x^{\tau(\sigma)}$ is currently full and was filled by σ, then the use is $> w$.

We now give an explanation for having each of the conditions 1-4 above. Condition 1 ensures that there are no pending changes due to incorrectly filled boxes from above. For condition 2, we note that if β has true outcome 0, then all the relevant boxes will have to be eventually cleared, so we don't want to believe in an α-computation until they are cleared. Condition 3 ensures that boxes of a lower $<_A$-priority will not be blocked by an increased restraint. Condition 4 is present because at each visit to β on the true path, we want to clear all the boxes filled by some $\sigma >_{left} \beta$ (even though these boxes might be of a higher $<_A$-priority).

Suppose $\alpha <_{left} \alpha'$ are negative nodes at the same level. As discussed previously, the two nodes are measuring the same length of convergence. Thus, our plan was to let α' wait until all the B-computations below $|dom(h_\alpha)|$ recover, before we allow α' to play an expansionary stage. However, α and α' may actually believe in different computations. In particular there might be some A-computation $\Phi^A(x)$ for some $x \in dom(h_\alpha)$ which α believed in (and hence placed in $dom(h_\alpha)$), but which α' does not believe in. It would be unreasonable for us to insist that α' wait for the recovery of such computations - instead we allow α' to neglect these computations when considering α'-expansionary stages. Doing so might cause h_α to record the wrong value, but as we will see, things will work out fine - h_α can only be wrong at finitely many values if α is to be on the true path.

Apart from considering lengths of agreement, we define a new parameter called the *length of believability*. The intention for this is to let α' assess the believability of $dom(h_\alpha)$: if α is an $\mathcal{N}_e$-node, we define the length of believability to be

$$l_b(\alpha, s) = \max\{y < s \mid (\forall x < y)\ (\Phi_e^A(x)[s]\downarrow, \text{ and is } \alpha\text{-believable})\}.$$

We also define the *length of agreement* measured at α to be:

$$l(\alpha, s) = \max\{y < s \mid (\forall x < y)\ (\Phi_e^A(x)[s]\downarrow = \Phi_e^B(x)[s]\downarrow), \\ \text{and } l_b(\alpha, s) \geq y\}.$$

We say that a stage s is *α-expansionary*, if α is visited at stage s, and both of the following hold :

1. $l(\alpha, s) \geq |dom(h_\alpha)[s]|$,
2. For each $\beta \in Left(\alpha)$ such that $R(\beta, s) = A$, and $l_b(\alpha, s) \geq |dom(h_\beta)[s]|$, we require that $l(\alpha, s) \geq |dom(h_\beta)[s]|$.

4.3.7 The Construction

Each stage s will either be an A-stage with enumerations into A, or a B-stage with enumerations into B. When we pick a *fresh* number at a stage s, we mean that we pick a number $> s$ and $>$ any number used or mentioned so far. At stage $s > 0$, we inductively define the stage s approximation to the true path, δ_s of length s. Suppose that $\alpha = \delta_s \upharpoonright d$ has been defined for $d < s$. If α is an $\mathcal{N}_e$-node or a $\mathcal{P}_e^A$-node, then we let $\delta_s(d) = \infty$ if s is α-expansionary, and $\delta_s(d) = f$ otherwise. If α is a $\mathcal{P}_e^B$-node, then we check if $B_s \cap W_{e,s} = \emptyset$. If so, we let $\delta_s(d) = w$, else we let $\delta_s(d) = s$.

Finally, suppose that α is a $\mathcal{P}^A_{e,i}$-node. If α is currently not active, we let $\delta_s(d) = 0$. Otherwise, for each $k \in L(\alpha, s)$, we have to let $\delta_s(d)$ specify either (k, ∞) or (k, f). For each $k \in L(\alpha, s)$, we check if the following holds (we will explain the choices for OUT.1-3 in the paragraph that follows):

- (OUT.1): The current box that α is pointing at in the k^{th} row (i.e. the $state(\alpha, s)$-box of $V_k^{\tau(\alpha)}$) is either unoccupied, or it is occupied by the correct value (i.e. contains $J^{\emptyset'}(k)[s]$).
- (OUT.2): $J^{\emptyset'}(k)[s] \downarrow$, and we require that the computation has persisted for at least two visits to α. We let $t < s$ be the least stage such that the computation $J^{\emptyset'}(k)[s] \downarrow$ applies at stage t (i.e. $J^{\emptyset'}(k)[t] \downarrow$ with $\emptyset'_t \upharpoonright j(k)[t] = \emptyset'_s \upharpoonright j(k)[t]$). Hence, α is visited at some stage u, $t \leq u < s$.
- (OUT.3): We also check that for every $k' \in L(\alpha, s) - \{k\}$, such that $J^{\emptyset'}(k')[t] \downarrow$ and $\emptyset'_t \upharpoonright j(k')[t] \neq \emptyset'_s \upharpoonright j(k')[t]$, there is a stage u such that $t < u < s$, α is visited at stage u, and $\emptyset'_t \upharpoonright j(k')[t] \neq \emptyset'_u \upharpoonright j(k')[t]$.

If all of the above hold, we let $\delta_s(d)$ specify (k, f), otherwise we let $\delta_s(d)$ specify (k, ∞). That is, we let $\delta_s(d) = \langle (k_0, x_0), \cdots, (k_p, x_p) \rangle$, where k_i's are precisely all the distinct elements of $L(\alpha, s)$, and the x_i's are specified above.

As promised, we will justify the choices for the conditions OUT.1-3 above - these are implemented purely for technical reasons. OUT.1 ensures that if the current box is occupied by the wrong value, then we immediately force outcome (k, ∞) - this ensures that the box will be cleared at the next possible chance. The combination of OUT.2 and 3 ensures that the true path is consistent with the "truth of outcome". In particular, consider the following scenario: suppose that $\{k_0, k_1\} \subset L(\alpha, s)$. It might be the case that both $J^{\emptyset'}(k_0) \uparrow$ and $J^{\emptyset'}(k_1) \uparrow$, in which case the correct outcome which we expect is $(k_0, \infty), (k_1, \infty)$. Because we combined both modules at the single node α, we have to ensure that this outcome is played infinitely often. This might be a problem if $J^{\emptyset'}(k_0)$ and $J^{\emptyset'}(k_1)$ take turns to show us a new value.

The solution to this is to ensure the following. If it is the case that $J^{\emptyset'}(k) \downarrow$ and has settled by stage t, then we will delay playing the outcome (k, f) until all the other modules $k' \in L(\alpha)$ which have not yet settled on $J^{\emptyset'}(k')$, have had a chance to play (k', ∞).

This completes the definition of δ_s. We now give the actions to be taken at stage s, in the following order:

1. Declare that s is an X-stage, where X is to be decided below. If $s = 0$ or $s = 1$, let $X = A$. Otherwise, look for the longest $\beta \subset \delta_s$ such that β has been visited prior to stage s. Let $s^- < s$ be the most recent stage such that $\delta_{s^-} \supset \beta$. If s^- is an A-stage, let $X = B$, otherwise let $X = A$.

2. Take actions for each expansionary negative node β. For each $\mathcal{N}_e$-node β such that $\beta^\frown\infty \subseteq \delta_s$, we do the following :

   (a) Set $R(\beta, s+1) = X^c$.

   (b) For all $x < l(\beta, s)$, we set $h_\beta(x) \downarrow = \Phi_e^A(x)[s]$, unless it is already defined.

   (c) Take actions for every β' whose restraint is transferred to β; this step of the construction is known as *transferring β'-restraint to β*: for every $\beta' \in Left(\beta)$ such that $R(\beta', s) = A$ and $l_b(\beta, s) \geq |dom(h_{\beta'})[s]|$, we do the following :

      i. Set $R(\beta', s+1) = B$, since this transferred B-restraint will be held until the next β'-expansionary stage.

      ii. Injure every negative node $\xi \supset \beta'^\frown f$, i.e. we set $h_\xi = \emptyset$, since this is a "pseudo" β'-expansionary stage.

3. Injure the negative nodes on the right, as well as the negative nodes on the left which have believed in wrong computations.

   (a) On the right: for each negative $\beta >_{left} \delta_s$ such that $|\beta| < s$, we set $h_\beta = \emptyset$, and set $R(\beta, s+1) = R(\delta_s \restriction |\beta|, s+1)$. This ensures we do not use a box of a stronger $<_A$-priority when we visit right of δ_s later.

   (b) On the left: for each negative $\beta <_{left} \delta_s$, $|\beta| < s$ such that $R(\beta, s) = A$ and $l_b(\delta_s \restriction |\beta|, s) < |dom(h_{\beta,s})|$, we set $h_\beta = \emptyset$. That is, β on the left has previously believed and set up restraint to preserve A-computations that are no longer believable. We injure β even though it is to the left, and we will have to show that this happens only finitely often if β is on the true path.

4. Injure all the mother nodes to the right.

   (a) For each mother node $\tau >_{left} \delta_s$, we set $V_k^\tau = \emptyset$ for all k, and set $u_{k,n}^\tau \uparrow$ for all k, n.

5. If $X = A$, we empty all the boxes which need to be cleared.

   (a) Action for nodes along δ_s: for each A-positive node $\beta \subset \delta_s$, and k such that $\delta_s(|\beta|)$ specifies (k, ∞), we clear the $state(\beta, s)$-box of $V_k^{\tau(\beta)}$, if it is not already empty.

   (b) Clear all boxes of lower $<_A$-priority: for each mother node $\tau \subset \delta_s$, a number x and a string $\gamma >_A state(\tau, \delta_s, s)$, we clear the γ-box of V_x^τ, if not already empty.

   (c) Clear all boxes filled by some $\sigma >_{left} \delta_s$, regardless of the $<_A$-priority: for all A-positive nodes β and σ such that $\beta \subset \delta_s$ and $\sigma >_{left} \beta$ with $\tau(\sigma) \subset \delta_s$, and some string $\gamma \supseteq state(\tau(\sigma), \beta, s)$, some number x, such that the γ-box of $V_x^{\tau(\sigma)}$ is currently full and was filled by σ, we clear the box.

6. If $X = A$, we will also fill all the boxes needing to be filled, unless the decision to fill a particular box made at the beginning of the stage, is now deemed unwise due to enumerations made in step 5.

   (a) For each A-positive $\beta \subset \delta_s$, and k such that $\delta_s(|\beta|)$ specifies (k, f), we place $J^{\emptyset'}(k)[s]$ into the $state(\beta, s)$-box of $V_k^{\tau(\beta)}$ with a fresh use, unless the box is already full.

   An exception to the rule is the following: a node $\beta \subset \delta_s$ might want to fill a box of $V_k^{\tau(\beta)}$ (hence β is active when we are defining δ_s above), but the actions in step 5 might have caused an enumeration into A below some Φ^A-use, and β is no longer active after step 5 is done. In this case, β would *not* fill the box.

   That is, we would do the above for each A-positive $\beta \subset \delta_s$, unless in step 5, we had made an enumeration into A below the use of $\Phi_r^A(n)[s]$ (where β is a $\mathcal{P}_{r,j}^A$-node) for some $n \leq 1 + \max L(\beta, s)$.

7. If $X = B$, we will give the B-positive nodes a chance to act.

   (a) For each $\mathcal{P}_e^B$-node β such that $\beta^\frown w \subseteq \delta_s$, enumerate into B, the least number x (if there is one) satisfying the following:

   - $x > 2e$ and $x \in W_{e,s}$,
   - $x > \max\{t < s \mid \delta_t <_{left} \beta\}$, and
   - for every $\mathcal{N}_{e'}$-node σ such that $\sigma <_{left} \beta$ or $\sigma^\frown f \subseteq \beta$ for some e', and for each $n \in \cup_{t \leq s} dom(h_\sigma)[t]$, we also require that $x > \varphi_{e'}(B; n)[s]$.

   That is, once a negative node $\sigma <_{left} \beta$ has a B-recovery, β will not be able to enumerate below the recovered use. This is to ensure that B-positive requirements obey the transferred B-restraint at all times.

4.3.8 Verification

We say that a node α plays an outcome x at a stage s, if $\delta_s \supset \alpha^\frown x$. It is easy to see that there is a leftmost path visited infinitely often: if α is A-positive and is never permanently active, then 0 is the leftmost outcome it plays infinitely often; on the other hand if α does become permanently active, then $L(\alpha)$ eventually settles and α will have to play one of the $2^{|L(\alpha)|}$ many possible outcomes. Let TP be the true path of the construction. We list a few facts regarding the construction:

1. Suppose α is visited infinitely often. Then for each $X \in \{A, B\}$, there are infinitely many X-stages s such that $\delta_s \supset \alpha$ (see lemma 4.2.3).

2. For each negative α, $dom(h_\alpha)$ is changed only at α-expansionary stages (where it is increased), or when it is *reset* (i.e. set to $\emptyset$).

3. For a negative α, there are three actions in the construction which can cause a change in $R(\alpha)$ (and therefore the *state* variable):

   - when we visit left of α,
   - during an α-expansionary stage, where we set $R(\alpha)$ based on whether the current stage is an A or B-stage,
   - when we visit right of α, and transfer α-restraint to the right; in this case we change $R(\alpha)$ from A to B.

4. Because of fact 3, if $\tau \subset \alpha$ are along the true path, then $state(\tau, \alpha)$ eventually settles. We denote the limit value of $state(\tau, \alpha)$ by $State^{\tau}_{\alpha}$.

For a node α on the true path, we let $True(\alpha)$ be the least stage s such that α is visited at stage s, the construction never visits left of α after stage s, and for every $\tau \subset \alpha$, $state(\tau, \alpha, s)$ has settled. Note that if $\alpha \subseteq \beta$, then $True(\alpha) \leq True(\beta)$.

We begin with a few preliminary lemmas. Firstly, we argue that for any segment $\tau \subset \alpha$ on the true path, once $state(\tau, \alpha)$ has settled, no box of a stronger $<_A$-priority than $state(\tau, \alpha)$ can be accessed:

Lemma 4.3.3. *Let $\tau \subset \alpha$ be on the true path, where τ is a mother node. Then, for every $t \geq True(\alpha)$, we have*

$$state(\delta_t\restriction|\tau|, \delta_t\restriction|\alpha|, t) \geq_A State^{\tau}_{\alpha}.$$

Proof. Let $s_0 = True(\alpha)$. Let ξ be a negative node such that $\tau \subset \xi^\frown f \subseteq \alpha$. We claim that if there is a ξ'-expansionary stage $t > s_0$ for some $\xi' \in Right(\xi)$, then $R(\xi)$ will be set to B (if ξ is not already holding B-restraint), and $R(\xi)$ will stay at B forever: otherwise h_ξ will be reset, and for every $\eta \in Left(\xi)$, either h_η will be reset, or $R(\eta)$ will be set to B (and stay that way). This is impossible because the next visit to ξ after t will have to be ξ-expansionary, by definition.

To prove the lemma, we suppose that there is a $t > s_0$ such that $state(\delta_t\restriction|\tau|, \delta_t\restriction|\alpha|, t) <_A State^{\tau}_{\alpha}$, and derive a contradiction. We have

$$\begin{aligned} state(\delta_t\restriction|\tau|, \delta_t\restriction|\alpha|, t) &= \sigma B \sigma' \\ State^{\tau}_{\alpha} &= \sigma A \sigma'' \end{aligned}$$

for some strings σ, σ' and σ''. The $(|\sigma|+1)^{th}$ digits in $state(\delta_t\restriction|\tau|, \delta_t\restriction|\alpha|, t)$ and $State^{\tau}_{\alpha}$ must correspond to negative nodes ξ_0 and ξ_1 respectively with $\xi_0 \geq_{left} \xi_1$. Clearly $\tau \subset \xi_1{}^\frown f \subseteq \alpha$, and also that $R(\xi_1) = A$ after stage s_0.

By the claim at the beginning of this proof, stage t cannot be a ξ_0-expansionary stage. It follows that $R(\xi_0, t) = B$. However when α was visited at stage $s_0 < t$, step 3(a) of the construction would have set $R(\xi') = A$ for every $\xi' \in Right(\xi_1)$. In fact, step 3(a) was included in the construction to ensure that the situation discussed here does not arise - we want to prevent a box of a stronger $<_A$-priority from being accessed later. Clearly $R(\xi')$ for all $\xi' \in Right(\xi_1)$ will have to be A after stage s_0, a contradiction. $\square$

Next, we turn our attention to the positive nodes. For the next two lemmas 4.3.4 and 4.3.5, we let τ be a $\mathcal{P}_e^A$-node and α be an i-daughter node of τ such that $\tau^\frown\infty \subset \alpha$, with both lying on the true path. Let $L := \lim_{s\to\infty} L(\alpha, s)$, which may or may not exist. It is not hard to see that

1. Φ_e^A is an order $\Rightarrow L$ exists and $L = \{k \mid \Phi_e^A(k) \in M_i^\tau\}$.

2. α becomes permanently active iff L exists, $L \neq \emptyset$ and for all $k \leq 1 + \max L$, $\Phi_e^A(k)\downarrow$.

3. If α never becomes permanently active, then 0 is the true outcome of α.

In the next lemma, we want to show that the true outcome is consistent with the "truth". This becomes a concern because we are combining several modules, each with their separate outcomes, in a single node. In (ii) we show that if $J^{\emptyset'}(k)\downarrow$, then not only will the true outcome specify (k, f), but there will also be only finitely many outcomes (k, ∞) being played at that level. This ensures that boxes filled by a positive node on the true path, do not become emptied while the construction was visiting right of α.

Lemma 4.3.4. *Suppose that α becomes permanently active.*

(i) Let $L = \{k_0, \cdots, k_p\}$. The true outcome of α is $\langle (k_0, x_0), \cdots, (k_p, x_p)\rangle$ where for each $i = 0, \cdots, p$, $x_i = \infty$ iff $J^{\emptyset'}(k_i)\uparrow$.

(ii) For each $k \in L$ such that $J^{\emptyset'}(k)\downarrow$, there can only be finitely many stages t such that $\delta_t \supset \tau^\frown\infty$ and $\delta_t(|\alpha|)$ specifies (k, ∞).

Proof. (i) We partition L into the two parts, $L_\infty := \{k \in L \mid J^{\emptyset'}(k)\uparrow\}$, and $L_f := L - L_\infty$. We let $s_0 \geq True(\alpha)$ be an A-stage large enough, such that α is visited at s_0, and at every stage $t \geq s_0$,

- $L(\alpha, t) = L$,

- α is active at stage t (if it is visited),

- for each $k \in L_f$, $J^{\emptyset'}(k)[t] \downarrow$ on the correct use, and

- for each $k \in L_f$, the $State^{\tau}_{\alpha}$-box of V^{τ}_{k} is either unoccupied, or else occupied by $J^{\emptyset'}(k)$. This is possible because if the $State^{\tau}_{\alpha}$-box is occupied by a number other than $J^{\emptyset'}(k)$, α will play an outcome specifying (k, ∞) every time it is visited, until the box is cleared of the erroneous value.

At every visit to α after s_0, α will play one of the outcomes from the list $\{\langle (k_0, a_0), \cdots, (k_p, a_p)\rangle \mid a_i = \infty \text{ or } f \text{ for all } i\}$. To prove that the true outcome of α is $\langle (k_0, x_0), \cdots, (k_p, x_p)\rangle$, we will do it in two parts. Firstly, we show that for each $k \in L_f$, there are only finitely many stages such that α plays an outcome specifying (k, ∞). Secondly, we will show that there are infinitely many stages such that α plays an outcome specifying *all of* $\{(k, \infty) \mid k \in L_\infty\}$.

To prove the first part, we fix a $k \in L_f$. Let $s_1 \leq s_0$ be the least such that $J^{\emptyset'}(k)[s_1] \downarrow$ on the correct use. For each $k' \in L_\infty$ such that $J^{\emptyset'}(k')[s_1] \downarrow$, there must be a change in $\emptyset'$ below the use after stage s_1. Wait until all the changes occur, for all these $k' \in L_\infty$, and we let $s_2 > s_0$ be the *second* (or more) visit to α after the last change. We claim that after stage s_2, any outcome played by α has to specify (k, f): Pick $k' \in L - \{k\}$, such that $J^{\emptyset'}(k')[s_1] \downarrow$ with $\emptyset'_{s_1} \upharpoonright j(k')[s_1] \neq \emptyset'_{s_2} \upharpoonright j(k')[s_1]$. We want to show that α is visited at some stage u between $s_1 < u < s_2$, where the change has already occurred by stage u. If $k' \in L_f$ then $u = s_0$ since any such change has to occur by stage s_0; on the other hand if $k' \in L_\infty$ then such a u must exist, by the choice of s_2.

To prove the second part, we let $s_3 > s_0$ be a stage where α is visited. We show that there is a stage $s_4 \geq s_3$ such that α plays an outcome specifying *all of* $\{(k, \infty) \mid k \in L_\infty\}$. We may assume that $\tilde{L} := \{k \in L_\infty \mid J^{\emptyset'}(k)[s_3] \downarrow\} \neq \emptyset$ (otherwise we can let $s_4 = s_3$). For each $k \in \tilde{L}$, there is a least stage $s_4^k > s_3$ such that α is visited, and $\emptyset'_{s_3} \upharpoonright j(k)[s_3] \neq \emptyset'_{s_4^k} \upharpoonright j(k)[s_3]$. Let $\tilde{k}$ be some member of $\tilde{L}$ with the largest $s_4^{\tilde{k}}$, and let $s_4 = s_4^{\tilde{k}}$. Therefore, if $s_4^- < s_4$ is the previous visit to α, then $\emptyset'_{s_3} \upharpoonright j(\tilde{k})[s_3] = \emptyset'_{s_4^-} \upharpoonright j(\tilde{k})[s_3] \neq \emptyset'_{s_4} \upharpoonright j(\tilde{k})[s_3]$, i.e. the oracle will change below the use for $\tilde{k}$, but only after stage s_4^-.

We argue that this choice of s_4 works, in particular we fix a $k \in L_\infty$ such that $J^{\emptyset'}(k)[s_4] \downarrow$, and we want to show that the outcome played by α at stage s_4 specifies

(k,∞). Assume the computation $J^{\emptyset'}(k)[s_4]\downarrow$ has persisted for two visits. Hence $k\neq\tilde{k}$, and therefore condition (OUT.3) in the construction is not met when deciding the outcome for k. Hence (k,∞) is played at stage s_4.

(ii) This is proved in a similar way as in (i). Fix a $k\in L$ such that $J^{\emptyset'}(k)\downarrow$, and let s_1,s_2 be as in (i). The stage s_2 works for (i), but in this case we have to wait until every node $\alpha'\in \mathit{Right}(\alpha)$ which has been visited prior to stage s_2, is visited again at least one more time (if ever), say by stage $s_5>s_2$. Now the argument in (i) can be used to show that at every stage $t>s_5$ such that $\delta_t\supset\tau^\frown\infty$, we either have $\delta_t(|\alpha|)=0$, or else $\delta_t(|\alpha|)$ specifies (k,f). □

The next lemma tells us that each time α plays outcome 0 - which will be the case every time A changes below the use of some computation in $L(\alpha)$ - certain boxes will also be cleared. (i) states that if k is not eventually in $L(\alpha)$, then all the γ-boxes for $|\gamma|=m_i^\tau$, which are incorrect locations for the k^{th} row, will be cleared eventually. (ii) covers the case when $k\in\lim L(\alpha)$, but yet α has true outcome 0. This might be possible if we have infinitely many uses for $\Phi_e^A(k)[t]$, so that $\Phi_e^A(k)\uparrow$, but each convergent value $\Phi_e^A(k)[t]\in M_i^\tau$. Each time α plays outcome 0, the box might be full, unlike in (i), because some sibling node of α might fill the box after each change in the use for $\Phi_e^A(k)[t]$. However, if α is to have true outcome 0, then this box cannot remain permanently filled - it has to be cleared before the next visit to $\alpha^\frown 0$.

Lemma 4.3.5. *Let τ and α be as above.*

(i) If α plays outcome 0 at a stage s, then for every number $k\notin L(\alpha,s)$ and string γ of length m_i^τ, the γ-box of V_k^τ has to be empty at stage s.

(ii) If α is visited at stage s, then for every k and string γ of length m_i^τ, if the γ-box of V_k^τ gets filled by the actions at stage s (if not already full), then the box will have to be emptied at least once before α can next play outcome 0.

Proof. (i) Suppose the contrary, let $k\notin L(\alpha,s)$ and γ of length $|\gamma|=|\mathit{State}_\alpha^\tau|$ be such that the box is full with use $u_{k,\gamma}^\tau[s]\downarrow$. The box must have been filled at some A-stage $t<s$, in which $\delta_t(|\alpha|)$ was active at stage t and $k\in L(\alpha,t)$. The fact that $k\notin L(\alpha,s)$, means that A has to change below the use of the $\Phi_e^A(k)[t]$-computation, i.e. change below t. Furthermore this change cannot happen at stage t itself -

otherwise we would have not have filled the box at stage t - hence the change occurs after $u^\tau_{k,\gamma}[s] > t$ was set, a contradiction. In fact, we don't need the assumption that α plays outcome 0, but this will always be true if k goes out of $L(\alpha)$.

(ii) Let α be visited at stage s, and $u^\tau_{k,\gamma}\downarrow$ at the end of stage s. This box is filled at some τ-expansionary stage $t \leq s$, in which $k \in L(\alpha, t)$. Before α can next play outcome 0, there must be a change below some computation in $L(\alpha, t)$, i.e. below t. As in (i), this change has to occur after $u^\tau_{k,\gamma} > t$ was set, a contradiction. □

In the spirit of minimal pair arguments, one would want to show that at each negative node α, at least one side of computations is preserved between α-expansionary stages. In the next lemma we show that if t_0 is an α-expansionary B-stage, then the A-computations will be preserved until one of the three things happen :

1. we come to the next α-expansionary stage,
2. h_α gets reset because some $\alpha' \in Right(\alpha)$ does not believe in the A-computations which α had believed in. This happens only finitely often, as shown in part (ii).
3. the B-side of the computations recover, and the A-restraint is dropped while the construction was at the right of α. α doesn't care if this happens, since the value is now preserved on the B-side.

Lemma 4.3.6. *Let α be an $\mathcal{N}_e$-node on the true path.*

(i) Suppose $t_0 \geq True(\alpha)$ is an α-expansionary X-stage, such that $X^c_{t_0}\upharpoonright m \neq X^c_{t_1}\upharpoonright m$ for some $t_1 > t_0$, where $m =$ use of $\Phi^{X^c}_e \upharpoonright dom(h_\alpha)[t_0]$ and $R(\alpha, t_1) = X^c$. Then, there must be a stage u such that $t_0 < u < t_1$, where either u is α-expansionary, or else h_α is reset at stage u.

(ii) h_α is reset only finitely often.

Proof. (i) Let $s_0 = True(\alpha)$, and t_0, t_1 and m be as in the statement of the lemma. Suppose on the contrary there is no such u. The number $k < m$ has to enter at an X^c-stage t, where $t_0 < t < t_1$. By the assumption that there is no such u, we must have $\delta_t >_{left} \alpha^\frown\infty$.

The case $X = A$ is easy, so we consider $X = B$. We will show that the enumeration of $k < m$ into A at stage t contradicts α-believability of the use m at stage t_0. Since uses are chosen fresh, we may assume that $k = u^{\tau}_{x,\gamma}$ was set before stage t_0, and that $\tau \subset \alpha$. Since $R(\alpha, t_1) = A$, it follows that $R(\alpha, u) = A$ at every stage $u = t_0 + 1, \cdots, t_1$. In fact, the following is true:

Claim 4.3.6.1. *For every $\alpha' \in Right(\alpha)$, and $t_0 < u < t_1$, we must have $R(\alpha', u) = A$, and that u is not α'-expansionary.*

Proof of claim. Since t_0 was α-expansionary, step 3(a) of the construction would have ensured that $R(\alpha') = A$ for all $\alpha' \in Right(\alpha)$. This stays that way until some α'-expansionary stage u, in which we will either transfer α-restraint by flipping $R(\alpha)$ over to B, or we will reset h_α, both of which are impossible by assumption. □

By Claim 4.3.6.1 we have $state(\tau, \delta_t, t) \supseteq state(\tau, \delta_t \restriction |\alpha|, t)^\frown A$, and by Lemma 4.3.3, we in fact have $state(\tau, \delta_t, t) >_A State^{\tau}_{\alpha}{}^\frown B$. At stage t, $k = u^{\tau}_{x,\gamma}$ was enumerated under step 5 of the construction. There are three cases:

1. Step 5(b): if this was the action putting k into A, then $\gamma >_A State^{\tau}_{\alpha}{}^\frown B$, contradicting α-believability of use m at t_0.

2. Step 5(a): we either have $\delta_t \supset \alpha^\frown f$, or $\delta_t >_{left} \alpha$. The former can immediately be seen to contradict α-believability, so we assume at stage t, the latter holds. Let β be the node which wants to clear $u^{\tau}_{x,\gamma}$, where $\gamma = state(\beta, t)$. If $|\beta| > |\alpha|$, then clearly $\gamma >_A State^{\tau}_{\alpha}{}^\frown B$, contradicting α-believability just as in 1. above. Suppose that $|\beta| < |\alpha|$, and so it follows that the use m was α-believable at stage t_0, but is not $\delta_t \restriction |\alpha|$-believable at stage t, i.e. $l_b(\delta_t \restriction |\alpha|, t) < |dom(h_\alpha)[t]|$. In this case the construction would promptly reset h_α at stage t, contrary to assumption.

3. Step 5(c): let $\beta \subset \delta_t$ be the node which wants to clear a box filled by $\sigma >_{left} \beta$. Clearly $\beta \subset \alpha$ can immediately be seen to contradict α-believability, so suppose $\beta \not\subset \alpha$. If $|\beta| > |\alpha|$ we would also have $\gamma >_A State^{\tau}_{\alpha}{}^\frown B$ just as in 1. above, while $|\beta| < |\alpha|$ means that $l_b(\delta_t \restriction |\alpha|, t) < |dom(h_\alpha)[t]|$ just as in 2. above.

All three cases are impossible, hence the enumeration of k cannot happen.

(ii) We note that a consequence of Lemma 4.3.4(ii), is that there is a stage s_1 large enough, such that for every A-positive node $\beta \subset \alpha$, and every k such that $\alpha(|\beta|)$ specifies (k, f), and every $\beta' \in \mathit{Right}(\beta)$ with $\beta' \supset \tau(\beta)^\frown\infty$, we must have β' playing (k, f) each time it is visited after stage s_1 (unless β' is visited for the very first time). That is, only the outcome (k, f) can be played at level $|\beta|$, unless we are visiting the node for the very first time (and thus the node is not active). We let $s_1 \geq s_0$ be large enough so that this holds, and argue that h_α cannot be reset after stage s_1.

The only way for h_α to be reset after stage s_1, would be for α to have believed in some computation which is placed in $dom(h_\alpha)$, but which later some $\alpha' \in \mathit{Right}(\alpha)$ refuses to believe in. We have to go through each case 1-4 of Definition 4.3.2 and show that they still apply.

Let $s_2 > s_1$ be a stage where h_α is reset. Each time h_α is reset after stage s_1, there must be an α-expansionary stage before h_α can next be reset, so we may as well assume that $s_1 < s_{exp} < s_2$ where s_{exp} is the largest α-expansionary stage less than s_2. Since $R(\alpha, s_2) = A$ (otherwise we would leave h_α alone at s_2), hence s_{exp} has to be a B-stage setting $R(\alpha) = A$. Letting $m =$ use of $\Phi_e^A \upharpoonright dom(h_\alpha)[s_{exp}]$, by part (i) we know that the A-side has to be preserved, i.e. $A_{s_{exp}} \upharpoonright m = A_{s_2} \upharpoonright m$, and therefore any witness to non-believability at stage s_2, has to be some $u^\tau_{x,\gamma} < m$, which was set before stage s_{exp}.

We let $\alpha' = \delta_{s_2} \upharpoonright |\alpha| \in \mathit{Right}(\alpha)$, i.e. α' is the node where $l_b(\alpha', s_2) < |dom(h_\alpha)|$ and refuses to believe in some computation in $dom(h_\alpha)$ with use $< m$. Corresponding to conditions 1-4 of Definition 4.3.2, we have one of the following:

1. For some A-positive node $\beta' \subset \alpha'$, we have $\gamma = \mathit{state}(\beta', s_2)$ and $\alpha'(|\beta'|)$ specifies (x, ∞): the first thing to note is that $\gamma \geq_A \mathit{State}^{\tau(\beta')}_{\alpha \upharpoonright |\beta'|}$. If they are not equal then we have $\gamma >_A \mathit{State}^{\tau(\beta')}_\alpha{}^\frown B$, which is impossible because of the α-believability of use m at stage s_{exp}. Hence, $\gamma = \mathit{State}^{\tau(\beta')}_{\alpha \upharpoonright |\beta'|}$. The question is, what is the true outcome $\alpha(|\beta'|)$? We know $\alpha(|\beta'|)$ cannot specify (k, ∞) because of α-believability of m. On the other hand, the true outcome $\alpha(|\beta'|)$ cannot specify (k, f) by the choice of s_1, otherwise (k, ∞) cannot be played at level $|\beta'|$ anymore. This leaves us with the fact that 0 is the true outcome of $\alpha \upharpoonright |\beta'|$. By α-believability of m again, it must be the case that $k \notin L(\beta', s_{exp})$,

in which case by Lemma 4.3.5(i), the use $u^{\tau}_{x,\gamma}$ has to be set after stage s_{exp}, a contradiction.

2. For some A-positive node β' which has been visited prior to s_2, we have $\beta'^\frown 0 \subseteq \alpha'$, $x \in L(\beta', s_2)$ and $\gamma = state(\beta', s_2)$: a contradiction is derived in a similar way as in 1. above.

3. For some mother node $\tau \subset \alpha'$, we have $\gamma >_A state(\tau, \alpha', s_2)^\frown B$: this is not possible because $\gamma >_A State^{\tau}_{\alpha}{}^\frown B$, and because of α-believability once again.

4. For some A-positive nodes β' and σ such that $\beta' \subset \alpha'$ and $\sigma >_{left} \beta'$, we have $\tau(\sigma) \subset \alpha$, and $\gamma \supseteq state(\tau(\sigma), \beta', s_2)$: we have $state(\tau(\sigma), \beta', s_2) \geq_A State^{\tau(\sigma)}_{\alpha \upharpoonright |\beta'|}$, and hence the inequality can be split into two cases: $=$ and $>_A$, both cases are treated in the same way as above.

Thus all 4 cases are not possible, so the use m has to be α'-believable at stage s_2. This shows that h_α can be reset only finitely often. $\square$

Lemma 4.3.7. *Along the true path of the construction, all the negative requirements are satisfied.*

Proof. Let α be an $\mathcal{N}_e$-node on the true path such that $\Phi^A_e = \Phi^B_e$ is total. It is easy to see that if α is visited at some stage s after which h_α is never reset, then necessarily $s \geq True(\alpha)$: this is because if some negative β such that $\beta^\frown f \subseteq \alpha$ has β-restraint being transferred, then the same step of the construction will also reset h_α. We first claim:

Claim 4.3.7.1. *There are infinitely many α-expansionary stages.*

Proof of claim. Suppose that on the contrary, there are finitely many α-expansionary stages - let s_{exp} be the last α-expansionary stage. Obviously h_α cannot be reset after s_{exp}, and hence by the observation above we have $s_{exp} \geq True(\alpha)$.

Since α-expansionary stages only occur if all the computations of $Left(\alpha)$ recover, we let $l = \max |dom(h_\beta)[s_{exp}]|$ for $\beta = \alpha$ or $\beta \in Left(\alpha)$. Let $s_2 > s_{exp}$ be a stage large enough, so that both A_{s_2} and B_{s_2} are correct up to $m =$ use of $\Phi^A_e \upharpoonright l$ and $\Phi^B_e \upharpoonright l$. The only reason why s_2 is not α-expansionary, is because the use m is not

α-believable at stage s_2. Hence, one of the cases 1-4 of Definition 4.3.2 must not apply.

Cases 1 and 4 give straightforward contradictions to the correctness of $A_{s_2}\upharpoonright m$. Suppose that case 2 fails, i.e. for some β with $\beta^\frown 0 \subseteq \alpha$, and some $k \in L(\beta, s_2)$, we have $u^{\tau(\beta)}_{k,state(\beta,s_2)}[s_2] < m$. In this case even though β has true outcome 0, it might be the case that boxes in the k^{th}-row always get filled by some overly zealous sibling node of β. Despite this, it follows by Lemma 4.3.5(ii) that the use $u^{\tau(\beta)}_{k,state(\beta,s_2)}[s_2]$ has to be cleared before $\beta^\frown 0$ can next be visited after s_2, another contradiction to the correctness of $A_{s_2}\upharpoonright m$. Lastly, suppose that case 3 fails, i.e. $u^{\tau}_{x,\gamma}[s_2] < m$ for some $\tau \subset \alpha$, x and $\gamma >_A State^{\tau}_{\alpha}{}^\frown B$. If $R(\alpha, s_2) = B$, then the construction would enumerate $u^{\tau}_{x,\gamma}[s_2] < m$ into A at s_2. So, $R(\alpha, s_2) = A$ but in that case s_{exp} has to be an B-stage. Since the A-computations have to be preserved after s_{exp}, it follows that $u^{\tau}_{x,\gamma}[s_2]$ has to be set before s_1, and thus would cause the use m not to be α-believable at stage s_1 - a contradiction again. □

By Lemma 4.3.6(ii), h_α is reset only finitely often, hence h_α is computable. As in the previous theorem, the fact that there are infinitely many α-expansionary stages does not immediately imply that h_α is total. One has to use Claim 4.3.7.1 to show that every use which appears non-believable must be cleared at α-expansionary stages. Similar to Claim 4.2.4.3, one can then show that h_α is total, i.e. every number is eventually put in $dom(h_\alpha)$.

Finally, it is not hard to see that h_α records the correct value. The only thing new in this construction, is that α-restraint might be transferred while the construction was visiting right of α (i.e. $R(\alpha)$ flips from A to B). However, this happens only if the B-computations measured by α recover, and the B-positive nodes to the right of $\alpha^\frown\infty$ always respect this use the instant it converges. Therefore, all negative requirements are satisfied. □

Lemma 4.3.8. *Along the true path of the construction, all the positive requirements are satisfied.*

Proof. Let τ be a $\mathcal{P}^A_e$-node on the true path, such that Φ^A_e is an order. Clearly ∞ is the true outcome of τ; we need to show that $\{V^{\tau}_k\}_{k\in\mathbb{N}}$ traces $J^{\emptyset'}$ correctly. We fix a number k. Let i be the least such that $\Phi^A_e(k) \in M^{\tau}_i$, and α be the version of $\mathcal{P}^A_{e,i}$

on the true path, with true outcome o. Hence α will eventually be responsible for tracing $J^{\emptyset'}(k)$ in V_k^τ.

We first of all show that $|V_k^\tau| < \Phi_e^A(k)$. A box in the k^{th} row cannot be permanently set by a node β, where $|\beta| \neq |\alpha|$: suppose such a β fills a box in the k^{th} row at stage s, and $k \in L(\beta, s)$. We know k eventually has to leave $L(\beta)$, so it is not hard to see that at the first visit to $TP\upharpoonright|\beta|$ (the true version of β) after k leaves $L(\beta)$, we must have $TP\upharpoonright|\beta|$ playing outcome 0. By Lemma 4.3.5(i), this means that all boxes on the k^{th} row filled at level $|\beta|$ has to be empty at that visit.

Hence any box of V_k^τ can only be filled permanently by a node at level $|\alpha|$. However such nodes will only fill a γ-box for $|\gamma| = m_i^\tau$. There are only $2^{m_i^\tau} < \Phi_e^A(k)$ many choices for γ.

Next, suppose that $J^{\emptyset'}(k)\downarrow$. We want to show that $J^{\emptyset'}(k) \in V_k^\tau$. We know α will become permanently active and has true outcome specifying (k, f). Let $s_0 \geq True(\alpha^\frown o)$ be large enough for our purpose. At stage s_0, α would put $J^{\emptyset'}(k)$ into the $State_\alpha^\tau$-box of V_k^τ (unless it is occupied, in which case it has to be occupied by the right value). The only thing that would stop us, is that step 5 of the construction had made an enumeration below some computation in $L(\alpha, s_0)$, in which case α would not be active at the next visit and plays outcome 0 left of its true outcome - impossible. At any rate the $State_\alpha^\tau$-box on the k^{th} row has to contain the correct value at the end of stage s_0, and we want to show that this box is never emptied after s_0.

Suppose on the contrary, that at stage $t > s_0$ of the construction, an enumeration $p \leq u^\tau_{k,State_\alpha^\tau}[s_0]$ was made. Now, p must be the use of some box, which is set at some stage $t' \leq s_0$. We may in fact assume that $True(\alpha^\frown o) < t'$ (by letting s_0 wait long enough), and the use p was set by node β at stage t'. There are now three cases, depending on the position of β; we want to get a contradiction in all three cases:

1. $\beta \subseteq \alpha$: then p is the use of the $State_\beta^{\tau(\beta)}$-box in some k'-row. Since β is along the true path, the only way for us to clear the use p at stage t, would be through Step 5a. Therefore the true outcome of β has to specify (k', ∞), and thus p actually has to be cleared at stage s_0, before α sets the use $u^\tau_{k,State_\alpha^\tau}$. This is impossible.

2. $\beta >_{left} \alpha$: step 5(c) of the construction was specially included, with the purpose of clearing all such use, which were set by nodes to the right of the true path.

3. $\beta \supset \alpha$: at stage t' when we set the use p, we would do it *after* α sets the use $u^{\tau}_{k,State^{\tau}_{\alpha}}[t']$. Since $u^{\tau}_{k,State^{\tau}_{\alpha}}[t'] < p \leq u^{\tau}_{k,State^{\tau}_{\alpha}}[s_0]$, it follows that the use p has to be cleared before stage s_0, another contradiction.

Hence such a use p cannot exist, so the $State^{\tau}_{\alpha}$-box, once filled by α, remains full. This shows A is ultrahigh. Next, we want to show that the capping companion B is noncomputable. B is certainly coinfinite, and since $dom(h_\sigma)$ only increases at σ-expansionary stages for an A-positive σ, it follows that the restraint in step 7(a) of the construction is finite for a B-positive node on the true path. □

This ends the proof of Theorem 4.3.1.

4.4 Further Questions

Several natural questions present themselves. For instance, in Theorem 4.3.1, if the B-positive requirements make infinitely many enumerations, such as in making B ultrahigh, u.a.e.d , or even just high, the method of transferring B-restraint no longer works. This is because if $\widehat{\mathcal{N}}$ to the right of $\mathcal{N}$ drops the A-restraint while holding B restraint for $\mathcal{N}$, it might do so while believing incorrectly (infinitely often) in certain B-computations.

1. Is there a minimal pair of c.e. ultrahigh, or a minimal pair of c.e. u.a.e.d? Generally given any arbitrary computable order f, is there a minimal pair of superhigh c.e. sets, via f?

2. Is there an c.e. set which is noncuppable and ultrahigh?

3. Is there a characterization of ultrahighness in terms of dominant functions?

Chapter 5

A pair of ultrahigh tt-degrees which are complements

The work in this chapter is joint with Jiang Liu.

5.1 Introduction

It is well known that producing minimal pairs or meets in a degree structure generated by a strong reducibility is usually easier than the corresponding actions in the Turing degrees; for instance in the tt-degrees the totality of the reductions concerned allows us to convert an infinite injury process into a finite injury-type argument. This has been exploited in literature to prove many results in the c.e. tt-degrees (regarding meets) which fail in the classical Turing degrees (and even in the wtt-degrees), precisely because of the following fact: in a tt-reduction, we must not only decide the use of the computation in advance (as in wtt-reductions), but we also have to promise the result of all possible configurations of the oracle, even before anything happens. Some examples demonstrating this phenomenon include a minimal tt-degree, the failure of Lachlan's nondiamond theorem [CFLW, JM85], and the fact that every degree is branching.

The arithmetical hierarchy is the standard yardstick which is used to measure the complexity of the (arithmetical) Turing degrees. The jump hierarchy then further distinguishes sets of the same arithmetical complexity, by measuring the complexity of the n^{th} iterate of the Halting problem relative to the set. In Turing degree theory, one could also measure the complexity of degrees below $\emptyset'$ via cappability and cuppability. Denote the upper-semilattice of the c.e. Turing degrees by $\mathcal{R}$, and the upper-semilattice of the c.e. tt-degrees by $\mathcal{R}_{tt}$.

In a curious way, cappable c.e. degrees can be thought of as bearing certain resemblance to $\mathbf{0}$. The defining characteristic of a cappable c.e. degree is *timing.* Enumerations into the set is possible only during certain windows of opportunity, called gap phases. During the other times (the co-gap phases) the set is not allowed to change below a certain length. In this way one can combine, for instance, such a gap/co-gap strategy with infinitary positive action (such as highness requirements) to make a high cappable c.e. degree. Indeed every nonbounding c.e. degree[1] is cappable; in fact a more complicated version of the gap/co-gap strategy was used by Lachlan in constructing a nonbounding c.e. degree. In the other direction, one could exploit the gap/co-gap timing properties exhibited by cappable sets, to show for instance, that every cappable c.e. degree forms a minimal pair with a high c.e. degree.

On the other hand a cuppable c.e. degree bears certain weak resemblance with $\mathbf{0}'$, in the sense that enumerations into the set must occur from time to time (in any enumeration of a Turing complete set, *every* requested change in the set must occur). Indeed if a set is low cuppable, then enumerations in the set must in fact occur promptly, and cannot be restricted by the gap/co-gap timing issues involved in being half of a minimal pair. This is due to a classical theorem of Ambos-Spies, Jockusch, Shore and Soare:

Theorem 5.1.1 ([ASJSS84]). *A c.e. degree is promptly simple iff it is low cuppable iff it is noncappable.*

In Theorem 4.2.1 we constructed a minimal pair of superhigh c.e. degrees, and left open the question as to whether or not there is a minimal pair of c.e. degrees which were both superhigh via a single arbitrary computable order f, and also the question of a minimal pair of ultrahigh c.e. degrees. We mention that both problems are significantly simpler when looking at minimal pairs in the wtt and the tt-degrees. We make the following conjecture, and support our conjecture with a short argument.

[1]A c.e. degree is nonbounding if it bounds no minimal pair.

Conjecture 5.1.2. *There is a pair of ultrahigh c.e. degrees* $\mathbf{a}, \mathbf{b}$*, which forms a minimal pair in the wtt-degrees.*

Recall that the problem with making both sets in Theorem 4.3.1 ultrahigh was the following. If we were dealing with minimal pairs in the Turing degrees, we have no way of predicting the use of computations which we are waiting for recovery. Hence a minimal pair requirement might be holding, say A restraint and waiting for some Φ^B recovery. We have to place the B-trace marker (i.e. the use of the trace $J^{\emptyset'}(x)$ in some B-c.e. set) before the recovery takes place. The problem was that we might place the B-trace marker badly; in that case when Φ^B recovers above the B-trace marker, we have now to freeze the B-trace marker. This will make the size of the traces grow exponentially, so that we still have a computable bound, but it is not compatible with, say requiring the traces to have sub-exponential bounds. If we were dealing with wtt-functionals, then we could place the B-trace markers cleverly above the projected use of every Φ^B of lower priority, so that the recovery Φ^B is always below our B-trace marker. We remark that making a tt-minimal pair is even easier; we will discuss this in Section 5.2.2.

Lachlan proved however, that making a minimal pair of c.e. sets which also joins to $\emptyset'$ is impossible. This led Jockusch and Mohrherr to investigate the situation in the tt-degrees. They showed [JM85] such a pair was possible in the tt-degrees, and made both degrees low. That is, there are low c.e. tt-degrees $\mathbf{a}$ and $\mathbf{b}$, such that $\mathbf{a} \cup \mathbf{b} = \mathbf{0}'$ and $\mathbf{a} \cap \mathbf{b} = \mathbf{0}$ holds in $\mathcal{R}_{tt}$. Hence this gave an elementary difference between $\mathcal{R}$ and $\mathcal{R}_{tt}$. Recently, Cenzer, Franklin, Liu and Wu [CFLW] showed that the complements $\mathbf{a}, \mathbf{b}$ could be made superhigh. In this chapter we improve their theorem by showing

Theorem 5.1.3. *There are ultrahigh c.e. sets* A *and* B *which forms a minimal pair in the tt-degrees, and such that* $\emptyset' \equiv_{tt} A \oplus B$.

Our notation and terminology are standard and generally follow Soare [Soa87]. Let φ_e, Φ_e^A be the eth partial computable function and the eth A-partial computable function, respectively. In particular, if $\varphi_e(x) \downarrow$, then $[e](x)$ denotes the truth table with numbering $\varphi_e(x)$ (in some effective enumeration of all truth table) and $|[e](x)|$ denotes the length of this truth table. For any set A, $[e]^A(x)$ is 0 or 1 depending on whether or not A satisfies the truth table condition with index $\varphi_e(x)$ (denoted by $A \models [e](x)$ if $[e](x) = 1$, otherwise, $A \not\models [e](x)$). So given two sets A and B, $A \leq_{tt} B$ iff there is an e with φ_e total such that for all x, $[e]^B(x) = A(x)$.

5.2 The proof

In this section we sketch a proof of Theorem 5.1.3, without going into the painful details.

5.2.1 Requirements

We construct two c.e. sets A and B such that A and B are ultrahigh, and form a minimal pair in the tt-degrees. We also ensure that $\emptyset'$ (which denotes the Halting set) is truth table reducible to $A \oplus B$. It suffices to satisfy the following requirements:

$\mathcal{P}$: $\emptyset' \leq_{tt} A \oplus B$,

$\mathcal{N}_{i,j}$: $[i]^A = [j]^B = f$ is total $\Rightarrow f$ is computable,

$\mathcal{R}_e^A$: Φ_e^A is an order $\Rightarrow \emptyset'$ is A-jump traceable via Φ_e^A,

$\mathcal{R}_e^B$: Φ_e^B is an order $\Rightarrow \emptyset'$ is B-jump traceable via Φ_e^B.

Here, we let $[i]$ be the i^{th} truth table reduction, to distinguish it from Φ_e, the e^{th} Turing reduction. We fix an enumeration $\{K_s\}_{s\in\omega}$ of $\emptyset'$, such that at each stage s, at most one number is enumerated. We let $|[i]|$ denote the use of $[i]$, and lowercase Greek letters denote the use of the corresponding Turing reductions.

5.2.2 Constructing tt-minimal pairs

The requirement $\mathcal{P}$ acts globally, and we code $\emptyset'$ into $A \oplus B$ in a tt way. At stage $s+1$, we will enumerate $\langle k_s, 0\rangle$ into A or B or both. These will be the only reason why numbers of the form $\langle n, 0\rangle$ ever enter A or B. It is clear that this achieves the desired truth table reducibility, for $\emptyset' = \{x : \langle x, 0\rangle \in A\} \cup \{y : \langle y, 0\rangle \in B\}$. The decision as to which side(s) (A or B or both) to enumerate depends on the active minimal pair strategy of the highest priority. This issue will be addressed later on.

We now turn to the minimal pair requirement $\mathcal{N}_{i,j}$. We describe how the totality of $[i],[j]$ helps. The reason why we need to guess outcomes in a Turing (or even wtt) minimal pair node, and hence causing the infinite injury is the following: suppose at some stage s we had computed $\Phi_i^A = \Phi_j^B$ up till some length $l+1$, and we need (for the sake of some other requirement) to enumerate a number $x < \varphi_i(l)$ into A. We do not know the result of the recovery of $\Phi_i^{A_s \cup \{x\}}$ (if any). It might very well be the case that $\Phi_i^{A_s \cup \{x\}} \upharpoonright l+1 \neq \Phi_i^{A_s} \upharpoonright l+1$ in which case the disagreement produced by the entry of x must be preserved forever by restraining $B_s \upharpoonright \varphi_j(l)$. On the other hand it might be the case that truly we have $\Phi_i^A = \Phi_j^B$; in this case infinitely often we will have to hold larger B-restraint (just in case upon each recovery we have disagreement). Each time we increase B-restraint, this action was actually unnecessary; however we will not know that fact until the next expansionary stage. The totality of the tt-functional helps us to decide when to increase B-restraint; in particular if $[i]^A = [j]^B$ then each time we enumerate such x into A, we will not need to increase the restraint on B. The only time we ever increase the restraint on B is when a disagreement is produced; all we need to do is to ensure that all but finitely many disagreements are preserved forever.

It is not difficult to see that the tt-minimal pair requirements can be combined with almost any positive demand on the sets produced. Indeed Degtev [Deg79] showed one could make the two sets Turing complete; although his example was indirect. Jockusch and Mohrherr [JM85] also showed that one could combine these requirements with making $A \oplus B$ tt-complete. We now elaborate on how to incorporate the tt-coding of $\emptyset'$ into $A \oplus B$. As mentioned above, this action is global, and the action depends on the active $\mathcal{N}_{i,j}$ of the highest priority, such that the enumeration of $\langle x, 0\rangle$ (for the sake of coding of $\emptyset'(x)$) changes either $[i]^{A_s}(k)$ or $[j]^{B_s}(k)$ for some k.

If $\mathcal{N}_{i,j}$ is not holding any restraint, then we enumerate $\langle x, 0\rangle$ into the side that changes the $\mathcal{N}_{i,j}$-computation. We thus produce a disagreement and $\mathcal{N}_{i,j}$ now increases its restraint. Suppose on the other hand $\mathcal{N}_{i,j}$ is currently already holding some positive restraint. We clearly have $[i]^{A_s}(k) \neq [j]^{B_s}(k)$ for some k; we want to ensure that this disagreement is preserved. There are three cases:

Case 1: If $[i]^{A_s}(k) = [i]^{A_s \cup \{\langle x,0\rangle\}}(k)$, then $\langle x, 0\rangle$ is enumerated into A but not B.

Case 2: If $[i]^{B_s}(k) = [i]^{B_s \cup \{\langle x,0\rangle\}}(k)$, then $\langle x, 0\rangle$ is enumerated into B but not A.

Case 3: If the enumeration of $\langle x, 0\rangle$ changes both $[i]^{A_s}(k)$ and $[i]^{B_s}(k)$, then $\langle x, 0\rangle$ is enumerated into both A and B.

In any case, the disagreement at $\mathcal{N}_{i,j}$ continues to be preserved, so that its restraint does not increase due to $\mathcal{P}$. This action may injure other $\mathcal{N}_{i',j'}$ of lower priority, but only finitely often, because each $\mathcal{N}_{i,j}$ will be considered by $\mathcal{P}$ in this way only finitely often.

5.2.3 Making A ultrahigh

The construction takes place on a tree, which grows downwards. We order the nodes lexicographically. We use $\supset$ for string extension. We let $\alpha <_L \beta$ denote that α is to the left of β. If $\mathcal{R}$ is a requirement, we say that α is an $\mathcal{R}$-node, if α is assigned the requirement $\mathcal{R}$. A negative node is an $\mathcal{N}_{i,j}$-node for some i, j, while an ultrahigh node is a $\mathcal{R}^A_{e,k}$-node or $\mathcal{R}^B_{e,k}$-node for some e, k. We will usually use α to refer to an A-ultrahigh node, and τ to refer to its antecedent. We also use ξ for a B-ultrahigh node with antecedent ζ.

For a detailed explanation of the basic strategy to make A ultrahigh, we refer the reader to Section 4.3. We sketch some basic ideas here. For every order Φ^A_e, we build a uniformly A-computably enumerable sequence $\{V^A_k\}_k$ such that for all $k \in \omega$, $|V^A_k| \le \Phi^A_e(k)$ and $J^A(k) \in V^A_k$ (if convergent). We divide $\mathcal{R}^A_e$ into infinitely many subrequirements $\mathcal{R}^A_{e,0}, \mathcal{R}^A_{e,1}, \cdots$ such that $\mathcal{R}^A_{e,k}$ defines V^A_k. To get $\{V^A_k\}_{k\in\omega}$, we define an A functional $\Psi^A(k, n)$ for $n < \Phi^A_e(k)$, and let $V^A_k = \cup_n \Psi^A(k, n)$.

The ultrahigh strategy interacts with the $\mathcal{N}$-strategies by the following. A higher priority $\mathcal{N}$-strategy may force α to make a mistake. In particular, if the $\mathcal{N}$-strategy is already holding some restraint on A, then this restraint must be obeyed by α. The number of higher priority $\mathcal{N}$-requirements we allow above a particular $\mathcal{R}^A_{e,k}$ will be determined by Φ^A_e. In particular, if $\Phi^A_e(k) < 2^n$, then there cannot be more than n many such $\mathcal{N}$-strategies. To achieve this, we arrange for many consecutive levels of the construction tree to be devoted to different $\mathcal{R}^A_{e,k}$-strategies in between two $\mathcal{N}$-strategies; we call this a *block* $\mathcal{B}$. The n^{th} block is denoted by $\mathcal{B}_n = \{\mathcal{R}^A_{e,k} : 2^n < \Phi^A_e(k) \le 2^{n+1}\}$; there will be exactly n many $\mathcal{N}$-strategies before block $\mathcal{B}_n$. In the construction, an ultrahigh node will be assigned an entire block (instead of a single subrequirement $\mathcal{R}^A_{e,k}$). We say that α is a $\mathcal{B}$-node, if it is assigned the block $\mathcal{B}$.

We now explain why such an arrangement of blocks will work. We claim that if $\mathcal{R}^A_{e,k} \in \mathcal{B}_1$, then either $\Psi^A(k,0)$ or $\Psi^A(k,1)$ will define the correct $J^{\emptyset'}(k)$ (if convergent) value. We let $\mathcal{R}^A_{e,k}$ be operating at level k'. Suppose $\beta_0, \cdots, \beta_M$ are all the negative nodes placed above $\mathcal{B}_1$, and β_0 is the leftmost node accessed infinitely often during the construction. We will need to coordinate the actions amongst all the nodes at level k'; in a classical high or superhigh construction, these highness nodes are on the same level, and working for the same (sub)-requirement. However they will make distinct attempts at tracing $J^{\emptyset'}(k)$ by defining and undefining $\Psi^A(k,r)$ for different r's. This clearly cannot work here; because of the arrangement of the blocks we may have much more than just 2 nodes at level k', but being in $\mathcal{B}_1$ we are only allowed to define $\Psi^A(k,0)$ and $\Psi^A(k,1)$. Therefore, all nodes at level k' will be allowed access to the definition of $\Psi^A(k,0)$ and $\Psi^A(k,1)$ (and only these two). The heuristics is that in a superhigh construction, we only care about having a computable bound. In an ultrahigh construction, we need to *save on the trace locations.*

We need to see why such a restriction on level $\mathcal{R}^A_{e,k}$ will work. This is best described in terms of the following: at level $|\beta_0|$ there is only one active restraint at any time; and before this restraint increases it has to be first taken down. More specifically, once some β_m puts up a restraint to protect a disagreement, all blocks $\mathcal{B}_1, \mathcal{B}_2, \cdots$ will have to respect this restraint. If $J^{\emptyset'}(k)$ later changes such that $\Psi^A(k,0)$ is wrong, then we will have to use $\Psi^A(k,1)$ to record the new $J^A(k)$-value. Note that as long as the restraint on β_m is in force, we will always be able to correct $\Psi^A(k,1)$ if we need to. This restraint on β_m can only be taken down either by the global coding actions of $\mathcal{P}$, or else when we move to the left of β_m. Therefore if $m = 0$ and β_0 ever puts up any restraint, then this restraint will be in force forever, so that β_0 will succeed in preserving the disagreement, and $\mathcal{R}^A_{e,k}$ will succeed needing only at most two tracing locations. The other possibility is that β_0 itself puts up no restraint, but infinitely often β_m for various $m > 0$ puts up some restraint which is later taken down. For instance, β_m may have put up some restraint r for $m > 0$, and later this is taken down when $\beta_{m'}$ for some $0 \leq m' < m$ is visited. In this situation, $\beta_{m'}$ will not believe any A-computation $[i]^A$ with a use larger than the use of $\Psi^A(k,0)$

if $\Psi^A(k,0)$ is currently recording the wrong $J^A(k)$-value. This is so that we do not put a new larger β-restraint until some α on level k' would have had the chance to correct $\Psi^A(k,0)$.

The above extends to the n^{th} block as well, and generally we need $\Psi^A(k,0)$, $\cdots, \Psi^A(k, 2^n-1)$ for every k in $\mathcal{B}_n$, since we need to record all possible restraint-states that the higher priority negative requirements are in. Note that due to the arrangement of the blocks, generally the n^{th} negative requirement will be placed at a level much larger than $2n$, and so there can be much more than 2^{2n} many different versions of the negative requirement. Hence the number of mistakes in which a requirement $\mathcal{R}^A_{e,k} \in \mathcal{B}_n$ is allowed to make depends not on the level at which it is operating at (as is the case in a high or superhigh construction), but rather on the *number of negative requirements* which are placed before it.

5.2.4 Technical considerations for implementing ultrahighness on the tree

Let $\mathcal{B}^\tau_n$ denote the n^{th} block of τ; note that $\mathcal{B}^\tau_n$ will change with time. As before the entire block $\mathcal{B}_n$ will be assigned to a single node α extending τ. We will say that τ is the *antecedent node* of α, and also that α is *a descendant node* of τ. Suppose that $\mathcal{R}^A_{e,k_0}, \mathcal{R}^A_{e,k_1}, \cdots, \mathcal{R}^A_{e,k_m}$ are in the same block $\mathcal{B}^\tau_n$. Then α will be responsible for tracing all of $\{k_i\}^{\emptyset'}(k_i)$ for $i \leq m$. The outcomes of α are exactly the same as in Section 4.3.

We have discussed the strategies involved; we will now discuss how to tidy up the layout of the construction tree. As we have seen, each negative level increases the number of trace locations needed by $\mathcal{R}$-nodes below it by a multiplicative factor of 2. We will equip each negative node with two outcomes: d stands for preserving a diagonalization, to the left of w which stands for waiting. There is no infinitary action associated with these nodes and outcomes; introducing outcomes for the $\mathcal{N}$-requirements will solely be for the purpose of categorizing the $\mathcal{R}$-nodes below. We do this by the following. Suppose that $\mathcal{B}^\tau_n$ is assigned to level k for some antecedent node τ. For each $\sigma \in \{w,d\}^n$, we define the class $C^\tau_{n,\sigma}$ to be the collection of all $\alpha \supset \tau$ and $|\alpha| = k$ such that the outcomes of the n negative nodes between τ and

α reads σ (when moving downwards). This clearly partitions the $\mathcal{B}_n^\tau$-nodes into 2^n many classes, labelled by σ. We arrange all the $\sigma \in \{w, d\}^n$ lexicographically from left to right $\sigma_0, \sigma_1, \cdots, \sigma_{2^n-1}$. For convenience we will also denote C_{n,σ_j}^τ by $C_{n,j}^\tau$ for all $j < 2^n$. $\mathcal{B}_n^\tau$-nodes of the same label σ will make use of the same trace location to trace $\{k\}^{\emptyset'}(k)$ for the associated k. Once the states of the negative nodes between τ and the n^{th} block increases (i.e. the state moves lexicographically left), we will move on to another trace location for the n^{th} block.

If $\alpha \in C_{b,\sigma}^\tau$, we let $\Psi_\alpha^\tau(k)$ denote $\Psi^A(k, n)$ for the appropriate trace location n. We drop mention of the oracle A. If $\alpha_1, \alpha_2 \in C_{b,\sigma}^\tau$, then $\Psi_{\alpha_1}^\tau(k) = \Psi_{\alpha_2}^\tau(k)$. We let $V_k^\tau = \{\Psi_\alpha^\tau(k) : \tau^\frown\infty \subseteq \alpha \text{ and } \alpha \in \mathcal{B}_b^\tau\}$, so that $|V_k| \leq 2^b$. Here, we let $\mathcal{B}_b^\tau$ be the b-th block working for τ. Since it is slightly harder to always arrange for exactly b many negative requirements between τ and $\mathcal{B}_b^\tau$, we will define

$$n_b^\tau = |\{\langle i, j\rangle : \tau < \mathcal{N}_{i,j} < \mathcal{B}_b^\tau\}|,$$

and adjust the contents of $\mathcal{B}_b^\tau$ accordingly. Note that the number n_b^τ is fixed. Since the size of the blocks needs to be adjusted dynamically, we define for α working for $\mathcal{B}_b^\tau$:

$$L(\alpha, s) = \{k < l(\tau, s) : 2^{n_b^\tau} < \Phi_e^A(k)[s] \leq 2^{n_{b+1}^\tau}\}.$$

That is, $L(\alpha, s)$ gathers all the subrequirements of τ which should be put into block $\mathcal{B}_b^\tau$ at stage s. Note that if Φ_e^A is an order, then every τ-block will eventually settle on a finite set, and $L(\alpha, s)$ also reaches a limit for every descendent α of τ. This corresponds to a Π_3^0-situation if we analyze the outcomes of requirements as in a $\emptyset'''$-argument. The other possibility is the Σ_3^0 situation, in which Φ_e^A is a constant function, but at all stages appears to be an order. In this case, at some block b, $\lim_s L(\alpha, s)$ will be a cofinite set. If this is the case then some descendent node α of τ will have true outcome 0.

At the $\mathcal{B}_b^\tau$-node α, we believe that the Π_3^0-situation holds if both the following holds:

(L1) $\Phi_e^A(l(\tau, s)) > 2^{n_b^\tau}$. *Hence Φ_e^A currently looks like it is an order.*

(L2) $L(\alpha, s) = L(\alpha, s^-)$, where s^- is the previous α-stage, and also that every $\Phi_e^A(k)$ for $k \in L(\alpha, s)$ has persisted for a while. *Hence $L(\alpha, s)$ currently looks stable.*

In this case, α can continue tracing $\{k\}^{\emptyset'}(k)$-values for all $k \in L(\alpha, s)$ as described previously. The uses of these traces are chosen fresh (in particular, larger than the Φ_e^A-uses), so that when α next hit the outcome 0, all of the previous traces would also be cleared at the same time. This is important because each time the $\mathcal{B}_b^\tau$-node α plays outcome 0, there would be some rearrangement of τ-subrequirements amongst $\mathcal{B}_b^\tau, \mathcal{B}_{b+1}^\tau, \cdots$. For instance, some $\mathcal{R}_{e,k}^A$ may have moved from $\mathcal{B}_{b+1}^\tau$ to $\mathcal{B}_b^\tau$. It is important that when the $\mathcal{B}_b^\tau$-nodes begin the next tracing phase, they start with a clean slate.

The rest of the proof involves a formal construction with the above-mentioned ideas. The verification process is routine, and is similar to the verification of Theorem 4.3.1.

Question 5.2.1. *Can we make the sets A and B of Theorem 5.1.3 Turing complete?*

Part II

Algorithmic Randomness

Chapter 6

Weak reducibilities and Π^0_1 classes

6.1 Weak reducibilities and jump traceability

We give a characterization of the following notion due to Simpson [Sim07]. A set A is said to be *weakly jump traceable* if there is a computable g such that for every x, $|W_{g(x)}| < \infty$ and $J^A(x)\downarrow \Rightarrow J^A(x) \in W_{g(x)}$. Hence this differs from jump traceability by losing the effectivity in the bound. We show that amongst the Δ^0_2 sets, weak jump traceability is the same as being low.

We define the class of reals which have *$\emptyset'$-bounded use* to be $\{X \in 2^\omega \mid \exists f \leq_T \emptyset'$ such that for every y, $J^X(y)\downarrow \Rightarrow f(y) >$ use on $J^X(y)\}$. That is, there is a Δ^0_2-bound on the amount of X needed to compute convergent $J^X(z)$ computations. It is easy to see that if $A \leq_T \emptyset'$, then A is low iff A has $\emptyset'$-bounded use. We show that

Theorem 6.1.1. *The following are equivalent.*

(i) A is weakly jump traceable,

(ii) A has $\emptyset'$-bounded use,

(iii) $\Theta^A(x) =$ least stage s where $J^A(x)[s]\downarrow$ is bounded by a Δ^0_2 function.

If $A \leq_T \emptyset'$ then all of these are equivalent to being low.

Proof. (i)$\Rightarrow$(ii): suppose A is weakly jump traceable. Let $\alpha(z)$ be a reduction function for the functional $\Psi^X(z) = \sigma$ where $\sigma \subset X$ and $J^\sigma(z)\downarrow$. Hence $\Psi^A(z)$ is traced by some $T_{\alpha(z)}$ with a finite bound on the cardinality. Use $\emptyset'$ to compute all the elements of $T_{\alpha(z)}$, and we can easily obtain an upper bound for the use of (convergent) $J^A(z)$.

(ii)⇒(iii): using $\emptyset'$ we compute a bound u on the use, as well as a stage s for which no string of length $< u$ will take longer than s stages to finish its computation.

(iii)⇒(i): use the fact that a Δ^0_2 function has an effective approximation which reaches a limit to get a trace for $J^A(z)$ with finitely many elements. □

Hence for Δ^0_2 sets, lowness can be expressed as a form of traceability; compare this to the relationship between superlow sets and jump traceable sets. Weak jump traceability generalizes lowness to the non-Δ^0_2 case, and can in fact be seen as a strong version of being GL$_1$: one of the standard ways of showing that a given set A is GL$_1$, is to obtain a Δ^0_2 bound on the function $\Theta^A(x) =$ in (iii) above. For instance,

1. Every Demuth random[1] is weakly jump traceable (but can never jump traceable).

2. If some $Z \leq_T \emptyset'$ is ML-random in A, then A is weakly jump traceable. In particular, every set low for Ω is weakly jump traceable.

We think that this notion is worth studying as a generalization of lowness, for this notion is at least closed downwards under Turing reducibility. It is clearly sufficient for bounding only GL$_1$ sets.

Question 6.1.2. *Is being weakly jump traceable equivalent to bounding only GL$_1$ sets?*

Nies [Nie09] suggested that a *weak reducibility* should be a preordering on $2^{\mathbb{N}}$ that can be used to compare sets by their computational complexity, much like Turing reducibility. These should be reducibilities which are implied by $\leq_T$, and as a relation on sets, should be arithmetically definable (with a simple formula). The intuition for a weak reducibility $A \leq_W B$ is that B should understand only part of what A can do, instead of being able to determine A completely as in $\leq_T$. For instance one can

[1]We will mention more on this in Chapter 9

compare two sets by their initial segments, K-complexity. One can look at the K-reducibility $A \leq_K B \Leftrightarrow \exists c \forall n (K(A\restriction n) \leq K(B\restriction n) + b)$, or the relative K-reducibility $A \leq_{rK} B \Leftrightarrow \exists c \forall n (K(A\restriction n \mid B\restriction n) \leq c)$.

One can also relativize a lowness notion to give a weak reducibility. This relativization has to possibly be a partial one to ensure that the reducibility obtained is transitive. For instance $\leq_{SJT}$ is transitive while $A \in SJT(B)$ and "A is sjt relative to B" are not. We mention a few other reducibilities arising in randomness, which have generated interest.

Definition 6.1.3. (i) *low for K reducibility*: $A \leq_{LK} B \Leftrightarrow \exists c \forall \sigma (K^B(\sigma) \leq K^A(\sigma) + c)$,

(ii) *low for random reducibility*: $A \leq_{LR} B \Leftrightarrow \forall Z(Z$ is B-random $\Rightarrow Z$ is A-random),

(iii) *jump traceable reducibility*: $A \leq_{JT} B \Leftrightarrow$ there is a computable order h, and a computable function g such that $\{W^B_{g(x)}\}$ obeys h, and for every x, $J^A(x) \downarrow \Rightarrow J^A(x) \in W^B_{g(x)}$.

Given a weak reducibility $\leq_W$, the equivalence classes induced by this preordering are called W-degrees. The sets in the bottom W-degree (i.e. the sets X such that $X \leq_W \emptyset$) for the reducibilities coming out of lowness notions are precisely the sets with the lowness notion. On the other hand one can look at the dual highness notion generated by these reducibilities (i.e. the sets X such that $\emptyset' \leq_W X$). These sets are called *W-hard.* Simpson [Sim07] showed that if $A \leq_{LR} B$ then $A \leq_{JT} B$. The LR-hard sets are the u.a.e.d. sets [KHMS06b].

The LK-reducibility $\leq_{LK}$ introduced in [Nie05b] is one of the standard measures of the relative compressing power of oracles. This definition formalizes the intuitive idea of B compressing more efficiently than A (modulo a constant). It is clear that if $A \leq_{LK} B$ then $A \leq_{LR} B$ (also introduced in [Nie05b]). The converse was shown by Miller in [Milb], so that the relations $\leq_{LK}, \leq_{LR}$ are identical. In particular, the relativized version of the result that the low for random sets are exactly the low for K sets also holds.

Lower cones of weak reducibilities can be uncountable. In Turing reducibility, each functional Φ maps A to at most one choice for $B = \Phi^A$. In $\leq_{LR}$ or $\leq_{JT}$

reducibilities, even a single "reduction procedure"[2] can have uncountably many B being reduced to a single A. For instance, there are uncountably many sets $\leq_{LR} \emptyset'$, while if $A \leq_{LR} B$ and B is low for Ω, then $A \leq_T B'$. Hence there are only countably many sets which are $\leq_{LR}$ a low for Ω set. It is still open as to whether the low for Ω sets characterize the sets with a countable $\leq_{LR}$-lower cone. There are uncountably many jump traceable sets; in fact there is a perfect Π^0_1-class of jump traceable sets. In the next theorem we show that $\emptyset'$ has uncountably many $\leq_{SJT}$-predecessors.

Question 6.1.4. *Does $A \leq_{SJT} B$ imply $A \leq_{LR} B$? How about when $A = \emptyset'$?*

A direct relativization of the proof when $B = \emptyset$ does not work, because the relativization in $\leq_{SJT}$ is a partial one. The box promotion method from [CDG08] does not seem to adapt directly when considering B-c.e. traces. The trouble is that we have to consider all possible initial segments of B, and we might get promotion at a certain W^ρ_x but along another incomparable $\rho' \mid \rho$, we have $W^{\rho'}_x$ not being promoted. Our basic box promotion strategy does not know how to proceed (unless of course we were able to test $J^{A\oplus B}$ instead of just J^A). This is a problem even if A is c.e. relative to B (or even just c.e.). In particular it is not known if SJT-hard implies LR-hard.

6.2 Uncountably many $\leq_{SJT}$ predecessors

We prove that there are uncountably many reals which are both $\leq_{CT} \emptyset'$ *and* $\leq_{SJT} \emptyset'$. $\leq_{CT}$ is the weak reducibility obtained from the lowness notion of being computably traceable. That is, $A \leq_{CT} B$ if there is a computable order h such that for every total $f \leq_T A$, there is a $g \leq_T B$ such that $\{D_{g(x)}\}$ obeys h and traces f. This shows that both weak reducibilities do not always have countable lower cones.

Theorem 6.2.1. *There is a perfect $\Pi^0_1(\emptyset')$ class of reals X such that $X \leq_{CT} \emptyset'$ and $X \leq_{SJT} \emptyset'$.*

Proof. Fix a nondecreasing unbounded function $h \leq_T \emptyset'$ such that h is dominated by every computable function. We define a perfect $\emptyset'$-computable tree T, and also

[2]In the case of $\leq_{LR}$, this is a relative Σ^0_1 set of strings; in $\leq_{JT}$ this is a relativized c.e. trace.

simultaneously define a $\emptyset'$-computable trace $\{V_x\}$ such that the following holds: for every $X \in [T]$, and every $x \in \mathbb{N}$, we have $|V_x| \leq h(x)$ and whenever $J^X(x)\downarrow$ then $J^X(x) \in V_x$. It is easy to verify that this shows both $X \leq_{CT} \emptyset'$ and $X \leq_{SJT} \emptyset'$.

Let $I_k = \{x \in \mathbb{N} \mid 2^k \leq h(x) < 2^{k+1}\}$. At stage s we assume we have defined $T(\sigma)$ for every $\sigma \in 2^{<s}$. We will define V_x and $T(\sigma)$ for every $x \in I_s$ and $|\sigma| = s$. Repeat the following procedure for each node $\sigma^\frown i$ where $|\sigma| = s-1$ and $i \in \{0,1\}$.

Procedure for $\sigma^\frown i$: if $I_s = \emptyset$ we set $T(\sigma^\frown i) = T(\sigma)^\frown i$. Otherwise we go through each $x \in I_s$, starting with the smallest and beginning with $\eta(\min I_s - 1) = T(\sigma)^\frown i$. Inductively we ask if there is some string ν extending $\eta(x-1)$ such that $J^\eta(x)\downarrow$. If ν exists we let $\eta(x) = \nu$ (for the first such ν found) and also put $J^\nu(x)$ into V_x; otherwise we simply let $\eta(x) = \eta(x-1)$. Once we have done this for every $x \in I_s$ we let $T(\sigma^\frown i) = \eta(\max I_s)$.

Verification: it is clear that T is perfect and $T \leq_T \emptyset'$. It is also clear that $V_x = D_{g(x)}$ for some $g \leq_T \emptyset'$ and every x, and that $|V_x| \leq h(x)$. Take an $X \in [T]$ and x such that $J^X(x)\downarrow$. Then $x \in I_k \neq \emptyset$ for some k, and in the procedure for $\sigma^\frown i$ (where $|\sigma| = k$ and $T(\sigma^\frown i) \subset X$), we would put $J^X(x)$ into V_x. $\square$

Question 6.2.2. *Which sets have uncountable* $\leq_{SJT}$ *lower cones?*

In particular, can a jump traceable set have uncountably many $\leq_{SJT}$ predecessors? On the other hand a strongly jump traceable set has only countably many $\leq_{SJT}$-predecessors, because every predecessor has to be itself strongly jump traceable and hence Δ^0_2. We think that there is a low c.e. set with uncountably many $\leq_{SJT}$-predecessors, but the situation for superlow sets is unclear. We conclude this discussion with a final question.

Question 6.2.3. *Are there minimal JT- or SJT-degrees?*

6.3 There is no nontrivial JT-base for randomness

In [Kuč93] Kučera defined a set A to be a *base for randomness* if $A \leq_T Z$ for some A-random set Z. He showed in the same paper that there were noncomputable sets A with this property and that all of them had to be GL_1. In [HNS07, Nie05b] it was shown that they coincide with the low for random sets. We study the following analogue of a base for randomness with respect to weak reducibilities.

Definition 6.3.1. If $\leq_W$ is a weak reducibility then we say that A is a *W-base for randomness* if $A \leq_W Z$ for some A-random set Z.

The notion of an LR-base for randomness was first studied in [BLS08b]. Trivially every low for random set is also a W-base for randomness (for any weak reducibility). In [BLS08b], a perfect Π^0_1-class P of reals which were all $\leq_{LR} \emptyset'$ was constructed, where P contained no K-trivial paths. As a corollary, every low for Ω path in P is an example of an LR-base for randomness which was not K-trivial. Since their example is necessarily not Δ^0_2, we ask

Question 6.3.2. *Do the bases for randomness and the LR-bases for randomness coincide within the Δ^0_2 sets?*

Since their example happened to be a low for Ω real, it is also natural to ask the following.

Question 6.3.3. *Does an LR-base for randomness have to be low for Ω? How about hyperimmunefree?*

It is not hard to show that every LR-base for randomness is GL_1.

Lemma 6.3.4. *For every set A which is not low for random and every oracle Σ^0_1 class V of bounded measure there exists a function $g \leq_T A \oplus \emptyset'$ such that $\mu\{M \mid U^{A \restriction g(n)} \subseteq V^M\} < 2^{-n}$.*

Proof. This lemma follows from the fact that LR-upper cones have measure 0 (see [BLS08a]) and that $S = \cap_j S_j$ where

$$S = \{X \mid U^A \subseteq V^X\} \qquad\qquad S_j = \{Y \mid U^{A \restriction j} \subseteq V^Y\}.$$

Now $\lim_j \mu(S_j) = \mu(S) = 0$, so given $n \in \mathbb{N}$ the oracles A and $\emptyset'$ can find some j such that $\mu(S_j) < 2^{-n}$. $\square$

Theorem 6.3.5. *If A is an LR-base for randomness then $A' \equiv_T A \oplus \emptyset'$, i.e. A is generalized low.*

Proof. Let $f \leq_T A$ be a partial function which, given $e \in \mathbb{N}$, waits until a stage s where e is enumerated into A' and defines $f(e) = s$ for the least such s. Suppose that A is not low for random (otherwise A is anyway low) and $A \leq_{LR} X$ for some A-random set X. Then there is an oracle Σ^0_1 class V of bounded measure such that $U^A \subseteq V^X$. Consider the following computable sequence of $\Sigma^0_1(A)$ classes

$$C_e = \begin{cases} \{M \mid U^{A\upharpoonright f(e)} \subseteq V^M\}, & \text{if } n \in A' \\ \emptyset, & \text{if } n \notin A' \end{cases}$$

and define $B_e \subseteq C_e$ as follows: start enumerating C_e into B_e until $\mu(B_e) = 2^{-e}$, in which case stop the enumeration. Now it is clear that (B_e) is a Martin-Löf test relative to A and therefore, any A-random set has to avoid B_i for all but finitely $i \in \mathbb{N}$. In order to show that $A' \leq_T A \oplus \emptyset'$ it suffices to consider the function g of Lemma 6.3.4 and show that

$$f(e)\downarrow \Rightarrow f(e) \leq g(e) \tag{6.1}$$

for all but finitely many $e \in \mathbb{N}$. This is true because when 6.1 fails for some $e \in \mathbb{N}$ by definition of B_e we have $X \in B_e$. □

We give a complete characterization of the notion of being a JT-base for randomness. We show that a real A is jump traceable if and only if $A \leq_{JT} Z$ for some Z which is A-random. That is, there is no nontrivial JT-degree which contains a JT-base for randomness. As a corollary we have that every LR-base for randomness is in fact jump traceable.

Theorem 6.3.6. *A real is jump traceable iff it is a JT-base for randomness.*

Proof. Suppose A is jump traceable. Then it is jump traceable by every real, including Ω^A. Suppose now that $J^A(x)$ is traced by some c.e. trace $\{T^Z_x\}_x$ with bound $t(x)$, where Z is A-random. We want to build a plain c.e. trace $\{V_x\}$ tracing $J^A(x)$.

Strategy: we use the fact that if A can be reduced to Z under some weak reducibility, where Z is A-random, then there cannot be many choices for A. A demonstration on the use of this fact can be found in the "hungry sets" theorem [HNS07]. Whenever $J^\sigma(x)$ converges on some string σ, we want to request for certification that σ is relevant to us (i.e. a possible initial segment of A). We do this by issuing small descriptions for every string ρ for which $J^\sigma(x)$ appears in T_x^ρ. Our descriptions for ρ are issued with oracle σ. We keep doing this until the measure of all strings ρ which traces $J^\sigma(x)$ have exceeded 2^{-x}, and when this happens we will trace $J^\sigma(x)$ into our plain trace V_x. Doing this ensures that:

1. the weight of all ρ described with oracle σ is at most 2^{-x}, and hence the total description with oracle A is at most 1,

2. we only trace some $J^\sigma(x)$ into V_x if a large number of ρ (with measure at least 2^{-x}) have traced the value of $J^\sigma(x)$. Hence the number of *different* $J^\sigma(x)$ we can trace in V_x is at most $2^x t(x)$. In other words, once V_x has filled up with $2^x t(x)$ distinct numbers, we can ignore any new convergent $J^\sigma(x)$, because these values can only be traced by ρ where $|T_x^\rho| > t(x)$.

Since Z is A-random, no initial segment of it can have a small description with oracle A. Hence we will discover that T_x^ρ traces $J^A(x)$ (for some $\rho \subset Z$) *only after we have seen a large number of incorrect* ρ' which also trace $J^A(x)$. This ensures that we will capture $J^A(x)$ in our plain trace V_x.

Notations: we build a $\Sigma_1^0(A)$-class U^A by enumerating axioms; the intention is that U^A is a Solovay test relative to A. The Solovay test will contain strings ρ for which we would like to issue small descriptions relative to A; this plays the role of the hungry set. For the purpose of this proof, when we refer to a convergent jump computation $J^\sigma(x)\downarrow$, we mean the minimal σ that produces a convergent computation. That is, we assume the set of oracle strings which produces convergence at each argument x is a prefix-free set of finite strings. We also assume that T_x^ρ is a finite set whose canonical index can be computed uniformly from ρ and x. The important point of assuming this convention is to ensure that no new elements get enumerated later into T_x^ρ on a use which is extended by ρ; if such elements get enumerated later (with use ρ at t), we assume they converge on all oracle strings extending ρ of length t. For a finite string η and a set of finite strings T, we say that η extends T if η extends some string in T.

Construction of U and $\{V_x\}$: for each $x < s$ we do the following at stage s. Look at all strings σ such that $J^\sigma(x)[s]\downarrow$ and the set $G(\sigma)[s]$ defined to be the set of all strings $\rho \in 2^{<\omega}$ with the following properties:

1. $|\rho| < s$,
2. $J^\sigma(x)[s] \in T_x^\rho$,
3. $|T_x^\rho| < t(x)$, and
4. ρ is minimal in the sense that its predecessor fails one of (i)-(iii).

It is clear that each $G(\sigma)[s]$ is a finite prefix-free set of strings, and furthermore $G(\sigma)[s] \subseteq G(\sigma)[s+1]$ (provided of course $J^\sigma(x)[s] \downarrow$). If $\mu G(\sigma)[s] < 2^{-x}$, we add $G(\sigma)[s]$ to U^σ. That is, for every $\rho \in G(\sigma)[s]$ such that $\langle \rho, \sigma' \rangle \notin U$ for any $\sigma' \subseteq \sigma$, we add $\langle \rho, \sigma \rangle$ to U. On the other hand if $\mu G(\sigma)[s] \geq 2^{-x}$ we trace $J^\sigma(x)[s]$ into V_x.

Verification: we first claim that for every x, $|V_x| \leq t(x)2^x$. Suppose the contrary, and let $J^{\sigma_1}(x)[s_1], \cdots, J^{\sigma_N}(x)[s_N]$ be distinct values in V_x enumerated into V_x at the respective stages $s_1, \cdots, s_N$, and $N = 1 + t(x)2^x$. At each s_i, we have $\mu G(\sigma_i)[s_i] \geq 2^{-x}$. Every ρ appearing in any of the $G(\sigma_i)[s_i]$ is of length $< s_N$. Fix an arbitrary η of length s_N. Now η can extend $G(\sigma_i)[s_i]$ for at most $t(x)$ many different i, because each i will correspond to a different value $J^{\sigma_i}(x)$ appearing in T_x^η.

Since $\mu G(\sigma_i)[s_i] \geq 2^{-x}$ and is prefix-free for each i, it follows that there are at least $2^{s_N - x}$ many different η of length s_N extending $G(\sigma_i)[s_i]$ for each i. A simple combinatorial calculation produces a contradiction.

Now we claim that $\mu U^A < 1$ along every infinite path A. Fix a path A. For each x we enumerate nothing into U^A unless $J^A(x) \downarrow$, and by convention there is a unique $\sigma \subset A$ which is taken to be the use. We will enumerate members of $G(\sigma)[s]$ into U^σ, and we only keep doing so if the cumulated contribution to U^σ is of measure $< 2^{-x}$ (since $G(\sigma)[s]$ is nondecreasing).

Now take reals A, Z such that $J^A(x)$ is traced by $\{T_x^Z\}_x$ with bound $t(x)$, and Z is A-random. Since $\mu U^A < 1$, Z extends only finitely many strings in U^A, say Z extends no string of length $> x_0$ in U^A. For every $x > x_0$ such that $J^A(x) \downarrow$, there is a unique $\sigma \subset A$ such that $J^\sigma(x) \downarrow$. At some point we must discover that $\mu G(\sigma)[s] \geq 2^{-x}$, because if $\mu G(\sigma)[t] < 2^{-x}$ for every stage t (after $J^\sigma(x)$ has converged), then some initial segment of $\rho \subset Z$ will be in $G(\sigma)[t]$, and furthermore $|\rho| > x_0$ (otherwise the measure of $G(\sigma)[t]$ would be too large). Hence ρ would be enumerated into $U^\sigma \subseteq U^A$, and is a contradiction. Hence $\mu G(\sigma)[s] \geq 2^{-x}$ at some s, where we will add $J^\sigma(x)$ to V_x. So, A is jump traceable via $\{V_x\}$. $\square$

Question 6.3.7. *Do the bases for randomness and the SJT-bases for randomness coincide within the Δ^0_2 sets?*

6.4 The importance of Π^0_1 classes in effective randomness.

The work in this section is joint with George Barmpalias and Andy Lewis. It is to appear in the Journal of Symbolic Logic.

6.4.1 Introduction

Π^0_1 classes in computability and effective randomness

Many arguments in computability theory and algorithmic randomness involve Π^0_1 sets of reals and techniques specific to such sets in an essential way. Two major references to such arguments in computability theory and in particular the degrees of unsolvability, are the well known Jockusch-Soare papers [JS72a, JS72b] on degrees of theories and members of Π^0_1 classes. In this work, Jockusch and Soare introduced the method of forcing with Π^0_1 classes, proved the now classic low basis theorem and showed a number of degree theoretic results using compactness arguments with Π^0_1 sets. For a survey of results concerning Π^0_1 classes in computability theory we refer the reader to Cenzer [Cen99]. In algorithmic randomness, Kučera's early papers [Kuč85, Kuč86] (partly inspired by some questions in the above papers of Jockusch and Soare) are a demonstration of how central Π^0_1 classes are in the study of the degrees of complete extensions of Peano Arithmetic (PA degrees) and effective randomness. In this work he also introduced fundamental methods for coding into PA degrees (using universal Π^0_1 classes) and coding into effectively random sets (using Π^0_1 classes of positive measure). The importance of Π^0_1 sets in arguments can be seen in a lot of recent work. As an example we mention the construction of a low bound for the ideal of K-trivial degrees by Kučera and Slaman [KS07] which uses Π^0_1 sets for coding in a very essential way.

In this section we show a number of results about PA degrees and relative randomness demonstrating the applicability of methods with Π^0_1 classes to the solution of some problems in this area, which do not currently have a known solution via different methods. Firstly we show that every PA degree is the least upper bound of two Martin-Löf random degrees, thus revealing more connections between these two classes of degrees, in the spirit of Kučera's work (see below). Secondly, we study relative randomness and in particular the LR-degrees using (for the first time in the literature) methods based on Π^0_1 sets. The main result of this approach is the construction of a minimal pair of LR-degrees below the LR-degree of the halting problem. This was a problem in the area of relative randomness which had resisted other techniques.

Background: PA and Martin-Löf random degrees

The collection of sets separating two disjoint c.e. sets is a natural Π^0_1 class. Hence the set of complete extensions of a consistent theory forms a Π^0_1 class, which is universal in some sense. A Turing degree is called PA if it is the degree of a complete extension of Peano Arithmetic. One of the standard characterizations of the PA degrees (due to Scott and Solovay, see e.g. [Odi89]) says that a Turing degree $\mathbf{a}$ is PA iff the degrees below $\mathbf{a}$ form a basis for Π^0_1 classes. In [JS72a, JS72b] Jockusch and Soare, partly motivated by the study of PA degrees, demonstrated how Π^0_1 classes and compactness arguments can be used in order to prove results about the Turing degrees. One of the most popular definitions of algorithmic randomness is the one given by Martin-Löf in [ML66], according to which a set is random if it does not belong in any 'effectively null' set in the Cantor space.

Kučera was the first to see the strong connection between the PA degrees and the Martin-Löf random degrees. In particular, in his well known early paper [Kuč85] he demonstrated how coding techniques based on Π^0_1 classes of positive measure can be applied in order to show results about the degrees of Martin-Löf random sequences. A distinctive feature of Kučera's work has always been that the theory of Martin-Löf random degrees is developed in parallel to the theory of PA degrees, with the techniques in the two topics being intrinsically connected. A definitive result about the relation between the PA degrees and the Martin-Löf random degrees (extending previous work of Kučera) was shown by Stephan in [Ste06] and says that a PA degree is Martin-Löf random iff it computes the halting problem. As discussed in [Ste06] this result strongly suggests a dichotomy of the Martin-Löf random degrees to the ones which contain a lot of information (they compute the halting problem) and the ones which are computationally weak, in the sense that they are not PA. In Section 6.4.2 we reveal another connection between these classes of degrees: every PA degree is the least upper bound of two Martin-Löf random degrees. The techniques involved in the proof are based on properties of Π^0_1 classes and Π^0_1 classes of positive measure.

Background: relative randomness, prefix-free complexity and LR-degrees

In Sections 6.4.3 and 6.4.4 we demonstrate how methods that are based on Π^0_1 classes can be used to prove results about relative randomness. Martin-Löf randomness is equivalent to the so-called Chaitin-Levin randomness, which is based on Kolmogorov's idea of incompressibility of binary strings. The coincidence of the LR-degrees with the LK-degrees makes this structure a very natural one to study. This was explored in [BLS08a, BLS08b, Sim07] and one of the questions that was not answered with the techniques developed in these papers was about the existence of minimal pairs of LR-degrees. Miller [Milb] later showed that there exist such minimal pairs by using a cardinality argument in conjunction with some properties of the low for Ω sets. He actually showed that every pair of relatively 2-random sets forms a minimal pair in the LR-degrees.

In Section 6.4.3 we use arguments based on Π^0_1 classes to show that there is a minimal pair of LR-degrees, LR below $\emptyset'$. This result cannot be obtained with previously known methods, and was the main motivation for introducing Jockusch-Soare (as in [JS72a, JS72b]) type of arguments for the study of the LR-degrees.[3] We also get a number of useful facts about upper cone avoidance in the LR-degrees in relation with Π^0_1 classes, which can be used in order to derive other results about $\leq_{LR}$. In Section 6.4.4 we discuss a number of other applications of Jockusch-Soare arguments to the study of the LR-degrees. In Section 6.4.5 we discuss how the same methodology can give results about the connection between hyperarithmetical complexity and $\leq_{LR}$.

[3]These include the compactness arguments they used, for example, in their proof that every Π^0_1 class with no computable paths contains two paths which form a minimal pair in the Turing degrees.

Finally in Section 6.4.6 we give a simple proof of a result in [Bar06] on cupping with random sets. The first result in this topic was given in Nies [Nie07] and was later simplified by Hirschfeldt and Miller using a more general argument with Π^0_2 null classes. We use the actual construction of Hirschfeldt and Miller as a "black box" to give a very short proof of the result in [Bar06], which was originally given as a generalization of the noncupping result of Nies [Bar06].

6.4.2 PA and random Turing degrees

In a number of widely cited papers [Kuč85, Kuč86] Kučera developed some theory about the PA, the d.n.c. and the Martin-Löf random Turing degrees. In particular, he treated these classes of degrees using similar approaches, while often commenting on the differences between the coding methods available for the PA degrees and the Martin-Löf random degrees. Coding into PA degrees was seen to be much more flexible than coding into a Martin-Löf random degree, and this is also reflected in [KS07] where all the K-trivial degrees are coded into a low PA degree, while it is not known whether there is a low random degree with this property. The relation between the PA degrees and the Martin-Löf random degrees was clarified by the following result of Stephan: a PA degree is Martin-Löf random iff it computes the halting problem. This is a clear demonstration of a well-known dichotomy in the Martin-Löf random degrees. There is a sharp qualitative distinction between the complete Martin-Löf random degrees and the incomplete ones. The former are random because they have a lot of information (indeed all complete degrees are random [Kuč85]) while the latter are often branded *the true Martin-Löf random degrees* (see [DHNT06]) and they have, as a matter of fact, low information content. For example, by the above mentioned theorem of Stephan, they cannot compute any complete extension of Peano arithmetic.

In this section we provide a further relation between these two classes.

Theorem 6.4.1. *Every PA degree is the join of two random degrees.*

This result, combined with the above mentioned theorem of Stephan, gives a plethora of pairs of Martin-Löf random degrees which join to a degree which is not Martin-Löf random; we just need to apply the result to any incomplete PA degree and get a pair with this property. In fact, this was the original motivation for this result.

Corollary 6.4.2. *Incomplete PA degrees are nonrandom degrees which are the join of two random degrees.*

The category version of Theorem 6.4.1 is not true, however. Recall that a sequence is weakly 1-generic if every dense Σ_1^0 set of strings contains a prefix of it, and is 1-generic if for every Σ_1^0 set of strings which does not contain a prefix of it there is a prefix of the sequence which is not a prefix of any string in that set.

Proposition 6.4.3. *There exists a PA degree which is not the join of two (weakly) 1-generic degrees.*

Proof. By the hyperimmunefree basis theorem applied to the complete Π_1^0 class containing only complete extensions of PA, there is a hyperimmunefree PA degree. This cannot be the join of two (weakly) 1-generics as such sequences are hyperimmune by a result of Kurtz [Kur81] (also presented in [DH09]) and hyperimmune degrees are upward closed. $\square$

Introduction to the proof of Theorem 6.4.1

Let C be a set of PA degree. We wish to find randoms A, B such that $C \equiv_T A \oplus B$. We would like to start with a Π_1^0 class P of randoms and find A, B inside P. The plan is to use a perfect tree T of paths in P in order to code C into the join of two of its paths A, B, thus achieving $C \leq_T A \oplus B$. The coding will be done in such a way that if C can compute the tree T, then $A \oplus B \leq_T C$. In order to achieve this, we will define *a class of trees* T such that $[T] \subseteq P$, which can be defined by a Π_1^0 formula. In other words, there is a Π_1^0 set of reals which effectively represent the trees in this class. Then we can use the fact that C is of PA degree to get a tree in that class which is computable from C, and then use it in order to do the coding for $C \leq_T A \oplus B$. Before we go into the details of the argument, let us give a formal definition of a tree.

Definition 6.4.4. If T is a partial function from $2^{<\omega}$ to $2^{<\omega}$ we say that T is a tree if for every $\sigma \in 2^{<\omega}$ and $i \in \{0, 1\}$ such that $T(\sigma * i) \downarrow$:

- $T(\sigma) \downarrow \subset T(\sigma * i)$;

- $T(\sigma * (1-i)) \downarrow |T(\sigma * i)$.

A tree T is perfect if $T(\sigma) \downarrow$ for all σ. A finite tree T of level n is the restriction of a tree (as a map) to strings of length n.

A first attempt would be (and indeed was) to use an (infinite) indifferent set on a random sequence (with respect to the Martin-Löf random sequences) in order to get the required tree of randoms for coding (see figure 6.1). In [FMN09] a set of positions was called *indifferent* for a sequence with respect to a class $\mathcal{A}$ if any alteration of the digits of the real in these positions produces a sequence in $\mathcal{A}$. It was proved (also see [Nie09]) that every random sequence has an indifferent set (with respect to the Martin-Löf random sequences). To find randoms A, B such that $C \leq_T A \oplus B$ we can take any random X and define

- A to be the random we get if we let the digit in the nth indifferent position of X be $C(n)$
- B to be the random we get if we let the digit in the nth indifferent position of X be $1 - C(n)$.

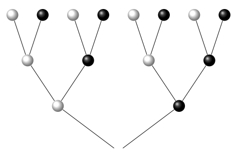


Figure 6.1: The associated tree of random sequences that we get if we put 0 (represented by a white ball) or 1 (represented by a black ball) into the indifferent positions on the real.

Now the indifferent set is computable in $A \oplus B$ (as it consists of the positions where A, B differ) and so is C. However the class of trees of this type cannot be expressed as a Π^0_1 class in the Cantor space. The reason for this is that 2^ω is compact while the space of perfect trees of Definition 6.4.4 (even the restricted class of Figure 6.1) with the natural topology generated by the finite trees[4], is not (being homeomorphic to

[4]In this topology the basic open sets are indexed by the finite trees F of Definition 6.4.4 and the basic open set corresponding to a particular F is the collection of perfect trees T which have F as an initial segment.

ω^ω). So although we can achieve $C \leq_T A \oplus B$ we cannot achieve Turing equivalence through this approach.

The trouble with considering arbitrary perfect trees T, is that there is no upper bound on $|T(\sigma)|$. If we only considered the subspace of perfect trees T such that $|T(\sigma)| \leq h(|\sigma|)$ for a fixed computable h, then the natural correspondence (between classes of trees and subspaces of the Baire space) gives a computable homeomorphism with the Cantor space 2^ω, which is compact. With this in mind, we make the following definitions.

Definition 6.4.5. Let $f : \mathbb{N} \to \mathbb{N}$ be an increasing function. The function f defines a partition on any given infinite string A. Let $(\sigma_A(i))$ be the unique sequence of strings such that $|\sigma_A(0)| = f(0)$, $|\sigma_A(i+1)| = f(i+1) - f(i)$ and

$$A = \sigma_A(0) * \sigma_A(1) * \dots \tag{6.2}$$

Say that A, B are *piecewise f-different from level n* if $\sigma_A(i) \neq \sigma_B(i)$ for all $i \geq n$.

Now given A, B which are piecewise f-different from level n, define the tree $T^{f,n}_{AB}$ as follows (for convenience we let $f(-1)$=0):

$$\begin{aligned} T^{f,n}_{AB}(\emptyset) &= A \restriction f(n-1) \\ T^{f,n}_{AB}(\tau * 0) &= T^{f,n}_{AB}(\tau) * \min\{\sigma_A(n+|\tau|), \sigma_B(n+|\tau|)\} \\ T^{f,n}_{AB}(\tau * 1) &= T^{f,n}_{AB}(\tau) * \max\{\sigma_A(n+|\tau|), \sigma_B(n+|\tau|)\} \end{aligned}$$

and consider the space

$$\mathcal{T}^{f,n} = \{T^{f,n}_{AB} \mid A, B \text{ are piecewise } f\text{-different from level } n\}.$$

Now it is not hard to see that $\mathcal{T}^{f,n}$ is compact and f-effectively homeomorphic to the Cantor space. The space $\mathcal{T}^{f,n}$ is still too big, since only the trees $T^{f,n}_{AB}$ such that $[T^{f,n}_{AB}]$ consists entirely of randoms are useful to us. Hence we define

$$\mathcal{C}^n_f = \{A \oplus B \mid A, B \text{ are piecewise } f\text{-different from level } n \text{ and } [T^{f,n}_{AB}] \subseteq P\}$$

(where P is a Π^0_1 class consisting entirely of random sequences) is Π^0_1 whenever f is computable. This is now exactly what we want because C can compute a tree $T^{f,n}_{AB}$ in $\mathcal{C}^n_f$. In particular $C \equiv_T A_0 \oplus B_0$ for the appropriate $A_0, B_0 \in [T^{f,n}_{AB}]$ obtained by

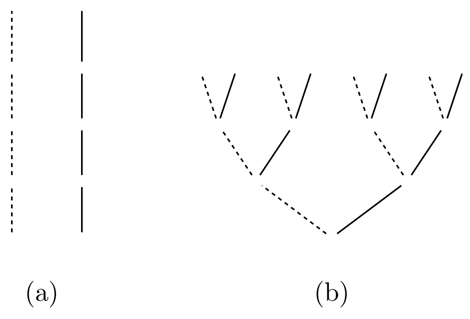


Figure 6.2: (a) The segments of piecewise f-different from level n reals A, B. Solid and dashed lines represent lexicographically larger and smaller corresponding segments respectively. (b) The tree $T^{f,n}_{AB}$.

switching segment of $A \upharpoonright f(n-1), B \upharpoonright f(n-1)$ according to $C(n)$. The only thing we need to show now is the following key lemma, which asserts that we can always choose f such that $\mathcal{C}^n_f \neq \emptyset$.

Lemma 6.4.6. *There exists a computable function f such that, if P is a Π^0_1 class of positive measure, then $\mathcal{C}^n_f \neq \emptyset$ for some $n \in \mathbb{N}$.*

We need to work with finite approximations to the notion of piecewise different pairs of reals, therefore we introduce the following terminology.

Definition 6.4.7 (Switching reals inside Π^0_1 classes)**.** Let $f : \mathbb{N} \to \mathbb{N}$ be an increasing function and P a Π^0_1 class. We say that A can be f-switched (or just switched) at $[n, m]$ inside P if it belongs to P and there exists B such that $\sigma_A(i) \neq \sigma_B(i)$ for all $i \in [n, m]$ and for every sequence $(x_i) \in \{A, B\}^\omega$ with $x_i = A$ for $i \notin [n, m]$, the real $\sigma_{x_0}(0) * \sigma_{x_1}(1) * \dots$ is in P. In this case B is a *switching partner of A at $[n, m]$ inside P*. We say that A can be f-switched (or just switched) from level n inside P if it belongs to P and there is B such that $\sigma_A(i) \neq \sigma_B(i)$ for all $i \geq n$ and for every sequence $(x_i) \in \{A, B\}^\omega$ with $x_i = A$ for $i < n$, the real $\sigma_{x_0}(0) * \sigma_{x_1}(1) * \dots$ belongs to P.

If $C_{n,m}(A)$ denotes the class of switching partners of A at $[n, m]$ inside P then $C_n(A) = \cap_m C_{n,m}(A)$ is the class of switching partners of A from level n inside P. Note that $C_{n,m}(A)$ is clopen and $C_n(A)$ is closed. Let $D_{n,m}$ (for $0 \leq n \leq m$) denote the set of reals which cannot be switched at $[n, m]$ inside P. So $D_{n,m} \subseteq D_{n,m+1}$ and it is clear that the classes $D_{n,m}$ are Σ^0_1 uniformly in f. Let D_n be the set of reals which cannot be switched from level n inside P.

Lemma 6.4.8. *$D_n = \cup_m D_{n,m}$ for all n.*

Proof. It is enough to show that $\overline{D}_n = \cap_m \overline{D}_{n,m}$ for all n. Indeed, if $A \in \cap_m \overline{D}_{n,m}$ by compactness we have that $C_n(A) = \cap_m C_{n,m}(A) \neq \emptyset$ and so $A \in \overline{D}_n$ (the other direction is trivial). $\square$

We are going to show the following, which is stronger than Lemma 6.4.6 since, by [Kuč85], a Π^0_1 class of positive measure contains a final segment of every Martin-Löf random real. Intuitively $\mathcal{C}^n_f \neq \emptyset$ follows from the fact that every random real in P can be switched (fixing some good choice for f). This in turn follows from the fact that the set of reals in P which *cannot be switched* in P can be effectively approximated by Σ^0_1 open sets.

Lemma 6.4.9. *There exists a computable function f such that if P is a Π^0_1 class and $X \in P$ is random then for some $n \in \mathbb{N}$ and some Y piecewise f-different to X from level n we have $[T^{f,n}_{XY}] \subseteq P$.*

Proof of Lemma 6.4.9

It suffices to inductively define a computable function f such that

$$\mu(\hat{D}_n) \leq \mathcal{O}(2^{-n}), \text{ where } \hat{D}_n = D_n \cap P. \tag{6.3}$$

Indeed, in that case the class $\cap_i \hat{D}_i$ is Π^0_2 and is null. So for every weakly 2-random $X \in P$ there some n such that $X \notin D_n$, which means that $[T^{f,n}_{XY}] \subseteq P$ for some Y. For (6.3) it suffices to make

$$\mu(\hat{D}_{n,n}) \leq 2^{-n-1} \tag{6.4}$$

$$\mu(\hat{D}_{n,m+1} - \hat{D}_{n,m}) \leq 2^{-n-m-2} \tag{6.5}$$

for all n and all $m \geq n$, where $\hat{D}_{n,m} = P \cap D_{n,m}$. We show that

$$\mu(\hat{D}_{n,n}) \leq 2^{f(n-1)} \cdot 2^{-f(n)} \tag{6.6}$$

$$\mu(\hat{D}_{n,m+1} - \hat{D}_{n,m}) \leq 2^{f(m)} \cdot 2^{f(m)-f(n-1)} \cdot 2^{-f(m+1)} \tag{6.7}$$

for an arbitrary increasing f, and then choose a computable f appropriately. For (6.6), fix σ of length $f(n-1)$ and for each τ of length $f(n)-f(n-1)$ let $M_{\sigma\tau}(n,n)$ be the set of reals B such that $\sigma * \tau * B \in \hat{D}_{n,n}$. By the definition of $\hat{D}_{n,n}$ we have that $M_{\sigma\tau}(n,n) \cap M_{\sigma\rho}(n,n) = \emptyset$ for any strings $\tau \neq \rho$ of length $f(n)-f(n-1)$. Hence $\sum_{\tau \in 2^{f(n)-f(n-1)}} \mu(M_{\sigma\tau}(n,n)) \leq 1$ and so $\mu(\hat{D}_{n,n} \cap [\sigma]) \leq 2^{-f(n)}$. Given that there are $2^{f(n-1)}$ such strings σ, we get (6.6).

For (6.7), say that a string τ of length $f(m)-f(n-1)$ is a switching string for $A \in P$ at $[n,m]$ inside P if for some (equivalently, for all) η of length $f(n-1)$ and $B \in 2^\omega$ the real $\eta * \tau * B$ is a switching partner for A at $[n,m]$ inside P. For any string τ of length $f(m)-f(n-1)$ let L_τ be the set of reals in P for which τ is a switching string at $[n,m]$. Since every $A \in P - \hat{D}_{n,m}$ belongs to some L_τ we have

$$\hat{D}_{n,m+1} - \hat{D}_{n,m} = \cup\{\hat{D}_{n,m+1} \cap L_\tau \cap [\sigma] \mid \sigma \in 2^{f(m)} \ \wedge \ \tau \in 2^{f(m)-f(n-1)}\}. \qquad (6.8)$$

Fix strings σ, τ of lengths as in (6.8) and for each string ρ of length $f(m+1)-f(m)$ let $M_{\sigma\rho,\tau}(n,m+1)$ be the set of all B such that $\sigma * \rho * B \in \hat{D}_{n,m+1} \cap L_\tau$. As before we have that $M_{\sigma\rho,\tau}(n,m+1) \cap M_{\sigma\rho',\tau}(n,m+1) = \emptyset$ for any $\rho \neq \rho'$ of length $f(m+1)-f(m)$. Hence

$$\sum_{\rho \in 2^{f(m+1)-f(m)}} \mu(M_{\sigma\rho,\tau}(n,m+1)) \leq 1$$

and so $\mu(\hat{D}_{n,m+1} \cap L_\tau \cap [\sigma]) \leq 2^{-f(m+1)}$. Then from (6.8) we get (6.7). Now if we let

$$f(0) = 1 \qquad\qquad f(n+1) = 2f(n) + n + 2 \qquad (6.9)$$

then (6.6) and (6.7) give (6.4) and (6.5) respectively. This concludes the proof of the lemma.

Proof of Theorem 6.4.1

Let P be a Π^0_1 class which contains only Martin-Löf random reals, fix f as defined in (6.9) and consider n such that $\mathcal{C}^n_f \neq \emptyset$. If C is of PA degree it computes some member $T^{f,n}_{AB}$ of $\mathcal{C}^n_f$. Without loss of generality we can assume that $A \upharpoonright f(n-1) = B \upharpoonright f(n-1)$ and that for each $i \geq n$ the string $\sigma_A(i)$ is lexicographically smaller than $\sigma_B(i)$ iff $C(i) = 0$. Then $A, B \in P$ and so they are random reals. We claim that $C \equiv_T A \oplus B$. It is clear that $A \leq_T C$ and $B \leq_T C$. On the other hand for every i, $C(i) = 0$ iff $\sigma_A(i)$ is lexicographically smaller than $\sigma_B(i)$.

6.4.3 LR minimal pairs and Jockusch-Soare arguments with Π^0_1 classes

LR minimal pairs and Π^0_1 classes

A basic question that one can ask about a reducibility on 2^ω or a degree structure is whether there exist minimal pairs. These are pairs of nontrivial (with respect to the particular reducibility) reals with the property that every real below both of them is trivial. Minimal pairs usually (for example with respect to the Turing reducibility) formalize the basic idea that two reals have no nontrivial common information. The same intuition applies to other reducibilities like LK, only that the notion of triviality is now weaker: a real is trivial with respect to $\leq_{LK}$ if it does not help a prefix free machine to compress more efficiently than when it works without an oracle. A minimal pair with respect to $\leq_{LK}$ is a pair of reals such that for every nontrivial real X at least one of them does not compress more efficiently than X.

Minimal pairs of LR- (and so, LK-) degrees were first constructed by Miller [Milb] via a measure theoretic argument (this proof also appears in [Nie09, Exercise 8.1.12]). He showed that for every low for Ω real A the set $\{X \mid X \leq_{LR} A\}$ is countable hence, given that nontrivial LR upper cones are null, he deduced the claim by the fact that a countable union of null sets is null.[5] With a similar argument (see [Milb, Nie09]) he also showed that every pair of relatively 2-random reals form a minimal pair in the LR-degrees (this was also observed by Yu). Since there is a pair of $\mathbf{0}''$-computable relatively 2-random reals[6], this implies that there two reals Turing below $\mathbf{0}''$ which form a minimal pair in the LR-degrees.

In this section we present another method of obtaining minimal pairs of LR-degrees, which allows to construct them inside null sets, for example LR below $\emptyset'$. This methodology involves Π^0_1 sets of reals and compactness arguments and goes back to [JS72b, JS72a] where it was used to derive results about the Turing degrees. Before we start the formal argument we recall the characterization of $\leq_{LR}$ given in [KH07]: $A \leq_{LR} B$ iff there exists V^B which is a $\Sigma^0_1(B)$ class of measure < 1 such

[5]The same argument gives minimal pairs in the Turing degrees, but in that case it is much easier since all Turing lower cones are countable.

[6]This follows by the low basis theorem relativized to $\emptyset'$ and the existence of a $\Pi^0_1(\emptyset')$ class which contains only 2-randoms.

that $U^A \subseteq V^B$ where U^A is a member of the universal Martin-Löf test relative to A; also, $A \leq_{LR} B$ iff every $\Sigma^0_1(A)$ class of measure < 1 is contained in a $\Sigma^0_1(B)$ class of measure < 1. From now on U will always denote a fixed member of the universal oracle Martin-Löf test and the term *bounded* $\Sigma^0_1(X)$ will refer to a $\Sigma^0_1(X)$ class of measure < 1. We often use the term *oracle* Σ^0_1 *class* to refer to a c.e. operator which takes a set X to an X-c.e. set W^X of strings, which is seen as the $\Sigma^0_1(X)$ class of reals consisting of the infinite binary extensions of the strings in W^X.

The following is an atomic version of LR cone avoidance inside a Π^0_1 class.

Lemma 6.4.10. *Let P be a nonempty Π^0_1 class, V an oracle Σ^0_1 class such that $\forall Z \in 2^\omega$, $\mu(V^Z) < 1$, and $A \not\leq_{LR} \emptyset$. Then there exists some $B \in P$ such that $U^A \not\subseteq V^B$.*

Proof. Suppose that for all $B \in P$ we have $U^A \subseteq V^B$. We define a Σ^0_1 class E such that $\mu(E) < 1$ and $U^A \subseteq E$, which shows that $A \leq_{LR} \emptyset$. Let

$$E = \{\sigma \mid [\sigma] \subseteq V^Z \text{ for all } Z \in P\}.$$

By hypothesis we have $U^A \subseteq E$ and by compactness E is a Σ^0_1 class. Now take $Z \in P$ which exists since $P \neq \emptyset$. Then $E \subseteq V^Z$ and hence $\mu(E) \leq \mu(V^Z) < 1$. □

Now we are ready to show the full LR cone avoidance theorem inside a Π^0_1 class.

Theorem 6.4.11. *Given a nonempty Π^0_1 class P and a countable sequence (C_i) of sets such that $C_i \not\leq_{LR} \emptyset$ for all $i \in \mathbb{N}$, there exists $B \in P$ such that $C_i \not\leq_{LR} B$ for all i.*

Proof. We show this for one C as it is clear how to generalize to (C_i). Let (V_e) be an effective enumeration of all oracle Σ^0_1 classes of bounded measure. We force with Π^0_1 classes and use Lemma 6.4.10. First note that if $U^A \not\subseteq V^Z_e$ then by the compactness of computations[7] there is $n \in \mathbb{N}$ such that $U^{A\restriction n} \not\subseteq V^Z_e$. So if Q_e is a Π^0_1 class containing Z such that $U^A \not\subseteq V^Z_e$ then there is some $n \in \mathbb{N}$ such that the Π^0_1 subclass

$$M_{e,n} = \{Z \in Q_e \mid U^{A\restriction n} \not\subseteq V^Z_e\}$$

[7]That is, the fact that oracle computations only use a finite segment of the oracle. This is also known as the *use principle*.

is nonempty. Now let $P_0 = P$ and inductively suppose that $P_i \downarrow$ and $P_i \neq \emptyset$. By Lemma 6.4.10 and the previous discussion there is $n_i \in \mathbb{N}$ such that the Π^0_1 class

$$\{Z \in P_i \mid U^{C\restriction n_i} \not\subseteq V_i^Z\}$$

is nonempty. Let P_{i+1} be this class. Now if we take $B \in \cap_i P_i$, then by construction we have $B \in P$ and $U^C \not\subseteq V_i^B$ for all $i \in \mathbb{N}$. □

An application of Theorem 6.4.11 gives the following, in analogy with the line of argument in [JS72b] for the Turing degrees.

Theorem 6.4.12. *Every nonempty Π^0_1 class contains two paths with greatest lower bound 0 in the LR-degrees.*

Proof. Let P be a nonempty Π^0_1 class. If it contains a low for random path, the claim is immediate. Otherwise, by the low for Ω basis theorem of [DHMN05] pick $C \in P$ which is low for Ω. By [Milb] the lower cone

$$\{Z \mid Z \leq_{LR} C\}$$

is countable, so let (C_i) be an enumeration of it. Now apply Theorem 6.4.11 and get $B \in P$ such that $C_i \not\leq_{LR} B$ for all $i \in \mathbb{N}$. Then B, C form a minimal pair in the LR-degrees. □

Now a combination of Theorem 6.4.12 with a result from [BLS08b] gives the minimal pair LR below $\emptyset'$ that we mentioned above.

Corollary 6.4.13. *There are $A, B \leq_{LR} \emptyset'$ which form a minimal pair in the LR degrees.*

Proof. In [BLS08b] it is shown that there is Π^0_1 class P such that $Z \leq_{LR} \emptyset'$ and $Z \not\leq_{LR} \emptyset$ for all $Z \in P$. If we apply Theorem 6.4.12 to P we get the claim. □

All known minimal pairs in the LR-degrees have the property that one member of the pair has countable lower cone and, indeed, is low for Ω.[8] It is therefore natural to ask the following.

Question 6.4.14. *Are there sets A, B which have uncountable LR lower cones and form a minimal pair in the LR-degrees?*

[8]By definition the property *low for* Ω is downward closed with respect to $\leq_{LR}$.

6.4.4 More Jockusch-Soare arguments with $\leq_{LR}$

The methodology introduced in [JS72a, JS72b] for the study of the Turing degrees through Π^0_1 classes and compactness arguments can also be applied to the study of randomness (weak) reducibilities—for example, the study of $\leq_{LR}$. Recall that the degree spectrum (with respect to some notion of degrees) of a Π^0_1 class is the set of the degrees of its members. We can show that *the LR-degree spectrum of a Π^0_1 class with no K-trivial members contains an antichain of size $2^{\aleph_0}$. Moreover this antichain can be chosen disjoint from any given countable sequence of nontrivial LR upper cones.* This can be seen as an analogue for $\leq_{LR}$ of Theorem 2.5 in [JS72b] which referred to the Turing degrees. Notice that a collection of reals that form an antichain of LR-degrees also forms an antichain in the Turing degrees.

Theorem 6.4.15. *The LR-degree spectrum of a Π^0_1 class with no K-trivial members contains an antichain of size $2^{\aleph_0}$. Moreover this antichain can be chosen disjoint from any given countable sequence of nontrivial LR upper cones.*

Proof. Let P be a Π^0_1 class which does not contain K-trivial paths. We show the first part of the theorem, and then indicate a modification which gives the more general statement. We inductively define a perfect tree T inside P by repeatedly applying Lemma 6.4.10, in such a way that for any two paths A, B on T we have $A|_{LR}B$. Along with the nodes T_σ we define Π^0_1 classes $P_\sigma \subseteq P \cap [T_\sigma]$ for $\sigma \in 2^{<\omega}$ which force the condition we wish to meet. Given an effective sequence (V_e) of all bounded oracle Σ^0_1 classes, our requirements are:

$$\forall X, Y \in [T]\ (X \neq Y \Rightarrow U^X \not\subseteq V_e^Y). \tag{6.10}$$

Let $P_\emptyset = P$, $T_\emptyset = \emptyset$ and inductively suppose that $P_\sigma \downarrow, T_\sigma \downarrow$, $P_\sigma \subseteq [T_\sigma]$ for all $\sigma \in 2^e$ such that for all distinct $\sigma, \tau \in 2^e$ we have

$$A \in P_\sigma \Rightarrow U^{T_\tau} \not\subseteq V_e^A \tag{6.11}$$

and, of course, $P_\sigma \neq \emptyset$. Now we define T_ρ, P_ρ for each $\rho \in 2^{e+1}$ in k steps, where k is the number of ordered tuples (σ, τ) for two distinct $\sigma, \tau \in 2^{e+1}$. Let (σ_i, τ_i) denote the ith such tuple and let the parameters at step 0 be as follows:

$$T_{\rho^- * j}[0] = T_{\rho^-} * \tau^j \qquad\qquad P_\rho[0] = P_{\rho^-} \cap [\tau^j]$$

where τ^0, τ^1 are incompatible extensions of T_{ρ^-} which have infinite extensions in P_{ρ^-}. Now given $s < k$ and assuming that $T_\rho[s], P_\rho[s]$ are defined (and $P_\rho[s] \neq \emptyset$) for all $\rho \in 2^{e+1}$, we check if ρ equals σ_s or τ_s. If not, we let the parameters as in the previous stage, i.e.

$$T_\rho[s+1] = T_\rho[s] \qquad\qquad P_\rho[s+1] = P_\rho[s].$$

To define the parameters for σ_s, τ_s we use Lemma 6.4.10 to find some $A \in P_{\sigma_s}$ such that

$$\{X \in P_{\tau_s} \mid U^A \not\subseteq V_e^X\} \neq \emptyset.$$

Then there must exist some $\eta \subset A$, $\eta \supseteq T_{\sigma_s}[s]$ such that

$$Q = \{X \in P_{\tau_s} \mid U^\eta \not\subseteq V_e^X\}$$

is not empty. Now define

$$\begin{aligned} T_{\sigma_s}[s+1] &= \eta & \qquad P_{\sigma_s}[s+1] &= P_{\sigma_s}[s] \cap [\eta] \\ T_{\tau_s}[s+1] &= T_{\tau_s}[s] & \qquad P_{\tau_s}[s+1] &= Q. \end{aligned}$$

When we finish all k steps for the eth level it is easy to verify that (6.11) holds. Therefore all requirements for the tree T are satisfied. For the second part of the theorem, suppose we are given a countable sequence of sets (C_i) which are not K-trivial and we wish to construct a tree T inside P such that (6.10) and $C_i \not\leq_{LR} X$ for all $i \in \mathbb{N}$ and $X \in [T]$. All we have to do is to augment the above construction with steps which deal with the requirements

$$\exists n \; [U^{C_i \restriction n} \not\subseteq V_e^X \text{ for all } X \in P_\rho,\ \rho \in 2^{\langle i,e \rangle}]$$

for all $i, e \in \mathbb{N}$. This step is a direct application of Lemma 6.4.10. □

In connection to this result, it is well known (see Sacks [Sac63a]) that the Turing degree spectrum of every perfect set of reals contains an antichain of size $2^{\aleph_0}$. We do not know if the same holds for the LR-degrees.

In relation to category, we can show that *the LR upper closure of a Π^0_1 class which does not contain K-trivials is meager.* This can be seen as the LR analogue of Theorem 5.1 in [JS72b]. The latter result states that the Turing upper closure of a Π^0_1 class with no computable members is meager.

Theorem 6.4.16. *The LR upper closure of a Π^0_1 class which does not contain K-trivials is meager.*

Proof. Let P be a Π^0_1 class which does not contain K-trivials and for a contradiction assume that the LR upper closure of P is not meager. Then for some e the set

$$S = \{X \mid \{A \in P \mid U^A \subseteq V_e^X\} \neq \emptyset\}$$

is not meager. Hence there exists a string σ such that every string $\tau \supset \sigma$ extends to member of S. We will define a computable real B in stages, so let $B[s]$ be the segment of B that has been defined by stage s ($B[s]$ can be thought of as a movable marker which moves monotonically through the full binary tree). Define the following length of agreement:

$$\ell[s+1] = \max\{t \mid \exists Y \exists \rho (Y \in P[s+1] \ \wedge \ B[s] \subseteq \rho \ \wedge \ U^{Y\restriction t}[s+1] \subseteq V_e^{\rho}[s+1])\}.$$

A stage s is called *expansionary* if $\ell[s] > \ell[t]$ for all $t < s$. Let $B[0] = \sigma$ (with σ as above) and if s is an expansionary stage, move $B[s]$ to a string $\rho \supset B[s-1]$ such that

$$\{Y \in P[s] \mid U^{Y\restriction \ell[s]}[s] \subseteq V_e^{\rho}\} \neq \emptyset.$$

Since S is not meager, B is well defined. Also, if

$$M_s = \{Y \in P \mid U^{Y\restriction s} \subseteq V_e^B\}$$

we have $M_s \neq \emptyset$ for all $s \in \mathbb{N}$. Since these are clopen sets and $M_{s+1} \subseteq M_s$, by compactness $\cap_s M_s \neq \emptyset$. If we consider some $X \in \cap_s M_s$ then $X \in P$ and $U^X \subseteq V_e^B$ which is a contradiction since V_e^B is a bounded Σ^0_1 class but by hypothesis $X \not\leq_{LR} \emptyset$. □

Naturally, results about the spectra of Π^0_1 classes have consequences in the study of the Medvedev and Muchnik lattices of Π^0_1 classes. Motivated by these connections, Cole and Simpson showed in [CS07] that given any special Π^0_1 class P (i.e. one containing no computable paths) we can find another special nonempty Π^0_1 class Q such that $X \not\leq_T Y$ for all $X \in P$, $Y \in Q$. The analogue of this result for the LR-degrees is also true:

Theorem 6.4.17. *For any Π^0_1 class P which does not have K-trivial members there is a Π^0_1 class Q which contains only members of effective packing dimension 1, such that $X \not\leq_{LR} Y$ for all $X \in P$, $Y \in Q$.*

Proof. Given P as above it suffices to define the approximation to a Π^0_1 class Q such that

$$R_{2e}: \ X \in Q \Rightarrow \exists n > e \ [K(X \upharpoonright n) > n \cdot (1 - 2^{-e})] \tag{6.12}$$

$$R_{2e+1}: \ X \in Q \Rightarrow \forall Y \in P \ [U^Y \not\subseteq V^X_e] \tag{6.13}$$

where U is a member of the universal oracle Martin-Löf test and (V_e) is an effective list of all bounded Σ^0_1 classes. To define Q we start from the full binary tree $T[0]$ and we start chopping particular branches, by enumerating the corresponding strings (as clopen sets) into the complement of Q. At every stage s the current approximation to Q is the set of all infinite paths through $T[s]$. Level ℓ of a tree is the set of all strings of length ℓ which are extendible in the tree (i.e. which are extended by an infinite path through the tree). Each strategy R_{2e}, R_{2e+1} will operate on a particular level, which may change as the construction progresses. Strategy R_{2e} will operate on level ℓ_{2e} and strategy R_{2e+1} on level ℓ_{2e+1}. We will have $\ell_i[s] < \ell_{i+1}[s]$ for all $i, s \in \mathbb{N}$.

R_{2e+1} *strategy.* We will use a version of the Sacks preservation strategy at level ℓ_{2e+1} of the tree. In order to describe the strategy let us assume inductively that ℓ_{2e} has reached a limit and that level ℓ_{2e} of T has settled, i.e. the strings of this level that are currently extendible in T will remain so. Let us denote the set of strings of level n of T by $L(n)$. For each string σ and $e \in \mathbb{N}$ we define the following *length of agreement* with respect to the eth LR-reduction:

$$d_e(\sigma)[s] = \max\{t \mid \exists Y \exists \rho (Y \in P[s] \ \wedge \ \rho \in T[s-1] \ \wedge \ \sigma \subseteq \rho \ \wedge \ U^{Y \upharpoonright t}[s] \subseteq V^\rho_e[s])\}.$$

A stage s is called *σ-expansionary* if σ has length $\ell_{2e}[s-1]$ for some $e \in \mathbb{N}$ and $d_e(\sigma)[s] > d_e(\sigma)[t]$ for all $t < s$. A stage s is called *e-expansionary* if it is σ-expansionary for some $\sigma \in L(\ell_{2e})[s-1]$. The strategy for R_{2e+1} is to wait for an e-expansionary stage $s+1$, choose some $\sigma \in L(\ell_{2e})[s]$ such that $s+1$ is σ-expansionary, consider a string $\rho \supset \sigma$ in $T[s]$ of length greater than $\ell_{2e+1}[s]$ such that

$$\{Y \in P[s+1] \mid U^{Y \upharpoonright d_e(\sigma)}[s+1] \subseteq V^\rho_e[s+1]\} \neq \emptyset \tag{6.14}$$

and chop from T all paths which extend σ and are incompatible with ρ. We let $\ell_{2e+1}[s+1] := |\rho|$ and for uniformity we also make sure that each string in $L(\ell_{2e})[s+1]$ has exactly one extension in $L(\ell_{2e+1})[s+1]$ (by arbitrarily chopping superfluous branches).

We claim that if we follow this strategy there can only be finitely many e-expansionary stages, so that ℓ_{2e+1} reaches a limit and the same happens for $d_e(\sigma)$ for all $\sigma \in L(\ell_{2e})$. In this case it is clear that no real extending a node in $L(\ell_{2e+1})$ (hence no path through T) can be LR above a member of P via the eth LR-reduction, i.e. R_{2e+1} is satisfied. If there were infinitely many e-expansionary stages then for some (say, leftmost) $\sigma \in L(\ell_{2e})$ there would be infinitely many *σ-expansionary* stages. But then $\ell_{2e+1} \to \infty$ and the construction would define an infinite computable sequence G, namely the unique path through T which extends σ, and if

$$M_s = \{Y \in P \mid U^{Y\restriction s} \subseteq V_e^G\}$$

we would have $M_s \neq \emptyset$ for all $s \in \mathbb{N}$. Since these are clopen sets and $M_{s+1} \subseteq M_s$, by compactness $\cap_s M_s \neq \emptyset$. If we consider some $X \in \cap_s M_s$ then $X \in P$ and $U^X \subseteq V_e^G$ which is a contradiction since V_e^G is a bounded Σ_1^0 class but by hypothesis $X \not\leq_{LR} \emptyset$.

R_{2e} strategy. For (6.12) we will use a computable function f such that for every $m, n \in \mathbb{N}$ and every string σ of length m, there exists $\tau \supset \sigma$ of length $f(m,n)$ such that $K(\tau) > |\tau| \cdot (1 - 2^{-n})$. The strategy for R_{2e} does the following for each σ of level ℓ_{2e-1}. First it chooses an extension τ of σ such that $[\tau] \subseteq Q[s-1]$ (i.e. no paths extending τ have been thrown out of Q so far) and removes all paths extending σ which are incompatible with τ from Q. Then all extensions of τ of length $f(|\tau|, e)$ are currently extendible in Q. It defines $\ell_{2e} = f(|\tau|, e)$ and in the following stages s, whenever

$$K_s(\rho) < |\rho| \cdot (1 - 2^e) \tag{6.15}$$

for some ρ of length ℓ_{2e}, we enumerate $[\rho]$ into the complement of Q, thus removing ρ from $L(\ell_{2e})$. The choice of f ensures that $L(\ell_{2e})$ will remain nonempty.

Construction. We say that R_{2e} *requires attention at stage* $s+1$ if either $\ell_{2e}[s] \uparrow$ or (6.15) holds for some ρ in $L(\ell_{2e})[s]$. We say that R_{2e+1} *requires attention at stage* $s+1$ if either $\ell_{2e+1}[s] \uparrow$ or $s+1$ is an e-expansionary stage. At stage $s+1$ proceed as follows for the least $i < s$ such that R_i requires attention. Suppose that $i = 2e$ for some $e \in \mathbb{N}$. If $\ell_{2e}[s] \uparrow$ then pick a *large* number t, for each $\sigma \in L(\ell_{2e-1})[s]$ remove all but one extension of σ of length t and set $\ell_{2e}[s+1] = f(t,e)$. If $\ell_{2e}[s] \downarrow$, for each $\sigma \in L(\ell_{2e-1})[s]$ (putting $\ell_{-1} = 0$) remove any extensions ρ of σ of length $\ell_{2e}[s]$ which satisfy (6.15). Now suppose that $i = 2e+1$ for some $e \in \mathbb{N}$. If $\ell_{2e+1}[s] \uparrow$ define $\ell_{2e+1}[s+1]$ to be a *large number* and for each $\sigma \in L(\ell_{2e})[s]$ remove all but one extensions of σ of length $\ell_{2e+1}[s+1]$. If $\ell_{2e+1}[s] \downarrow$ choose some $\sigma \in L(\ell_{2e})[s]$ such that $s+1$ is σ-expansionary, consider a string $\rho \supset \sigma$ in $T[s]$ of length greater than $\ell_{2e+1}[s]$ such that (6.14) holds and chop from T all paths which extend σ and are incompatible with ρ. Let $\ell_{2e+1}[s+1] := |\rho|$ and ensure that each string in $L(\ell_{2e})[s+1]$ has exactly one extension in $L(\ell_{2e+1})[s+1]$ by chopping superfluous branches. When any strategy acts, all lower priority strategies are initialized.

Verification. The construction computably enumerates clopen sets into the complement of Q, so the set Q of reals that are not prefixed by any strings enumerated by the construction is a Π^0_1 set. By induction we show that the parameters ℓ_n, $L(\ell_n)$ reach a limit and that the corresponding requirements are satisfied. Suppose that this holds for all $i < n$. If n is even, when ℓ_{n-1}, $L(\ell_{n-1})$ reach a limit strategy R_n will define ℓ_n and after a finite number of stages $L(\ell_n)$ will stabilize, containing at least one extension for each $\rho \in L(\ell_{n-1})$ with the property that $\neg$(6.15) (see the analysis of strategy R_{2e}). If n is odd, say $2e+1$, by the analysis of R_{2e+1} this strategy will only act finitely many times with respect to each string in $L(\ell_{n-1})$, thus finitely many times over all. It will define final values for ℓ_n, $L(\ell_n)$ at some stage s and R_n will be satisfied as there will be no more e-expansionary stages after stage s. □

Since every Π^0_1 class contains a path of c.e. Turing degree, Theorem 6.4.17 implies the following.

Corollary 6.4.18. *If P is a Π^0_1 class with no K-trivial members then there exists a set A of packing dimension 1 and c.e. degree, such that $X \not\leq_{LR} A$ for all $X \in P$.*

Finally, we can show the following, which can be seen as a strengthening of Theorem 4.7 of [JS72b].

Theorem 6.4.19. *There is a perfect Π^0_1 class such that any two distinct members of it are LR-incomparable.*

Proof. We only give a brief sketch of the proof, as it does not involve new ideas. According to the above mentioned characterization of $\leq_{LR}$, it suffices to build a perfect Π^0_1 class P and an oracle Σ^0_1 class $U_\star$ such that $\mu(U_\star^Z) < 1$ for each $Z \in 2^\omega$ and

$$\forall\ X, Y \in P\ (X \neq Y \Rightarrow U_\star^X \not\subseteq V_e^Y)$$

for all $e \in \mathbb{N}$, where (V_i) is an effective list of all bounded oracle Σ^0_1 classes. Given two extendible in P strings σ, τ we describe a strategy which ensures that $U_\star^X \not\subseteq V_e^Y$ for all $X \in P \cap [\sigma]$, $Y \in P \cap [\tau]$ and succeeds by adding arbitrarily small (say, at most 2^{-t}) measure in $U_\star^Z$, for each $Z \in 2^\omega$. Once this is clear, the full construction can be induced as a finite injury argument using these atomic strategies, where every time a strategy is injured the measure quota 2^{-t} that it holds becomes half of its present value (this ensures that $U_\star$ is a *bounded* Σ^0_1 class). The strategy is to pick 2^t incomparable strings ρ_i extending σ, which are currently extendible in P (such strings will exist since σ is currently extendible in P), split 2^ω into 2^t equal intervals of size 2^{-t} and enumerate each of them in exactly one $U_\star^{\rho_i}$, $i < 2^t$. It also removes from P all reals extending σ which do not extend one of ρ_i. From this point on, every time that some string $\eta \supseteq \tau$ is found with $U_\star^{\rho_j} \subseteq V_e^\eta$ for some $j < 2^t$, the strategy removes from P all reals extending τ which are incomparable with η and all reals extending ρ_j. And so on. Since V_e is a bounded oracle Σ^0_1 class, it follows that this procedure will terminate, leaving at least one ρ_i extendible in P and satisfying the requirement $U_\star^X \not\subseteq V_e^Y$ for all $X \in P \cap [\sigma]$, $Y \in P \cap [\tau]$. Moreover notice that the strategy has added at most 2^{-t} measure in $U_\star^Z$ for any $Z \in 2^\omega$. $\square$

6.4.5 Hyperarithmetical complexity LR below $\emptyset'$

The Jockusch-Soare methodology discussed above is also a very useful tool for the study of features of $\leq_{LR}$ that do not have an analogue in the Turing degrees. For example, we can show that there is a proper hyperarithmetical hierarchy of LR-degrees below the LR-degree of the halting problem. An LR-degree is Δ^0_2 if it contains a Δ^0_2 set. Similarly, it is Δ^0_α (where α is a computable ordinal) if it contains a set in Δ^0_α. For the definition of the hyperarithmetical hierarchy we refer the reader to [AK00]. Recall from [AK00] that given Kleene's $\mathcal{O}$ as a system of notations for the computable ordinals we can define the sets $H(a)$ for $a \in \mathcal{O}$ by recursion, in such way that $H(x) \equiv_T H(y)$ for notations $x, y \in \mathcal{O}$ representing the same ordinal. Given a computable ordinal α, let $\emptyset^{(\alpha)}$ be some $H(a)$ for a notation $a \in \mathcal{O}$ such that a represents α. For an infinite ordinal α we let Δ^0_α be the class of oracles which are computable from $\emptyset^{(\alpha)}$; for finite ordinals n let Δ^0_n be the usual arithmetical class $\Sigma^0_n \cap \Pi^0_n$ (notice the nonuniformity in the transition from the finite to the infinite case). We can show that *for each computable ordinal $\alpha \geq 2$ there is an LR-degree below the LR-degree of $\emptyset'$, which is Δ^0_α and is not Δ^0_γ for any $\gamma < \alpha$.* The proof is a forcing argument with Π^0_1 classes. It uses a Π^0_1 class with no K-trivial paths, such that all of its paths are $\leq_{LR} \emptyset'$. This was constructed in [BLS08b]. Also, it uses certain features of $\leq_{LR}$ like those used in Section 6.4.3.

Theorem 6.4.20. *For each computable ordinal α there is a Δ^0_α LR-degree below the LR-degree of $\emptyset'$, which is not Δ^0_γ for any $\gamma < \alpha$.*

Proof. We start with the Π^0_1 class P constructed in [BLS08b], which does not contain any K-trivials and $X \leq_{LR} \emptyset'$ for all $X \in P$. First assume that $\alpha = \beta+1$ is a successor computable ordinal and $\beta \geq 1$. It suffices to construct a set $X \leq_T \emptyset^{\beta+1}$ such that the following requirements are satisfied.[9]

$$R_e : \text{If } \Phi_e^{\emptyset^{(\beta)}} = Z_e \text{ is total and } U^{Z_e} \subseteq V_e^X \text{ then } Z_e \leq_{LR} \emptyset$$

The construction will be computable in $\emptyset^{(\beta)}$ and X will be approximated effectively in $\emptyset^{(\beta)}$. Notice that since $\beta \geq 1$ we can ask Σ^0_1 questions during the construction. We have parameters O_e which will be defined (and revised) during the construction and will be either $\emptyset$ (if we believe that Z_e is either K-trivial or not total) or a nonempty Π^0_1 class (otherwise). Let

$$J_s = \{j \mid O_j[s] \neq \emptyset\}$$

and let X_s be the leftmost path of $\cap_{j \in J_s} O_j$ of length s (by convention the empty intersection equals the entire Cantor space). We set $O_e[0] = \emptyset$ for all $e \in \mathbb{N}$. At

[9]Notice that these requirements guarantee a stronger result: the LR-degree that we construct is Δ^0_α and it *does not bound* any nontrivial LR-degrees which are Δ^0_γ for some $\gamma < \alpha$. Thus Theorem 6.4.20 could be stated in this more general form. With a little bit more effort it is possible to construct (given any computable ordinal α) an LR-degree which is Δ^0_α and it is incomparable with any nontrivial LR-degree which is Δ^0_γ for some $\gamma < \alpha$.

stage $s+1$ look for the least $i \leq s$ such that $i \notin J_s$ and for some $n < s$ such that the segment $Z_s \upharpoonright n$ is defined and if

$$Q_{i,n} = \{Y \mid U^{Z_i \upharpoonright n} \nsubseteq V_i^Y\}$$

the class $\cap\{O_j \mid j \in J_s \ \wedge \ i < e\}) \cap P \cap Q_{i,n}$ is nonempty. If there is such i, set $O_i = Q_{i,n}$ and $Q_j = \emptyset$ for all $j > i$. The sequence (X_s) is computable in β so $X \leq_T \emptyset^{\beta+1}$. By induction we show that for all $e \in \mathbb{N}$ parameter O_e reaches a limit and R_e is satisfied. Indeed, suppose that this holds for all $i < e$ and s_0 is a stage where no R_i, $i < e$ receives attention. If R_e receives attention after s_0, it will never receive attention again. Also, if $\Phi_e^{\emptyset^{(\beta)}} = Z_e$ is total and Z_e is not K-trivial then by Lemma 6.4.10 some k, s will be found such that $\cap\{O_i \mid i \in J_s \ \wedge \ i < e\}) \cap P \cap Q_{e,k} \neq \emptyset$, and O_e will permanently take the value $Q_{e,k}$. Since $X \in Q_{e,k}$, requirement R_e is satisfied.

Finally assume that α is a limit ordinal. But then $\emptyset^{(\alpha)}$ has computable access to the triple jumps of all oracles $\emptyset^{\beta}$ for $\beta < \alpha$. This means that it can instantly decide if they are K-trivial (given that K-triviality is a Σ_3^0 property) and hence, whether to act for R_e or not. If it decides to act, it defines $O_i = Q_{e,n}$ for a large enough n and proceeds with R_{e+1}. The construction now is just a forcing argument with Π_1^0 classes, in particular there is no injury in contrast with the successor ordinal case. $\square$

6.4.6 Random noncupping revisited.

The property of joining a random degree to $\emptyset'$ was first addressed by Kučera in a meeting in Cordoba in 2004. The first result was produced by Nies in [Nie07] where he constructed a promptly simple set which cannot be joined with a random to $\emptyset'$. Such sets with the latter property were called Martin-Löf noncuppable. He also observed that such a set has to be K-trivial and showed that the above result holds even if we replace $\emptyset'$ with an arbitrary Δ_2^0 random set. Shortly after this proof was circulated, Barmpalias [Bar06] produced a different proof, which shows the stronger result that $\emptyset'$ in the above statement can be replaced by an arbitrary Δ_2^0 set. In fact, he showed the following.

Theorem 6.4.21 (Barmpalias [Bar06]). *For every Δ_2^0 noncomputable set Y there is a promptly simple set A such that $Y \leq_T A \oplus R \Rightarrow A \leq_T R$ for every random set R.*

Later Hirschfeldt showed that if $\emptyset' \leq_T A \oplus R$ for a K-trivial A and a Martin-Löf random set R then $\emptyset' \leq_{LR} R$, and Miller and Hirschfeldt (see [Nie05a] or [Nie09]) showed that for every null Σ^0_3 class there exists a promptly simple set which is Turing below all Martin-Löf random members of the class. Given that the class of LR-complete sets is Σ^0_3 and null, and it contains Turing incomplete Martin-Löf random sets, it follows that there is a promptly simple Martin-Löf noncuppable set.

Let (W_e) be the standard effective sequence of all c.e. sets and let (Φ_e) be the standard effective sequence of all Turing functionals. In the following we give a simple proof of Theorem 6.4.21 *given* the proof of the above mentioned result of Hirschfeldt and Miller. Their proof is effective and produces a c.e. set. It can also easily be combined with lowness requirements, so that it uniformly produces a lowness index for the constructed set. In other words, this modified argument can be seen as a pair of computable functions f_1, f_2 which take an index of a Σ^0_3 class[10] S and return a c.e. index $f_1(e)$ of a c.e. set W (i.e. a number k such that $W = W_k$) and a lowness index $f_2(e)$ of W (i.e. a number t such that $W' = \Phi^{\emptyset'}_t$) such that

- W is Turing below all Martin-Löf random members of S
- if S is null then W is promptly simple.

Notice that by the properties of f_1, f_2 we also have $W'_{f_1(e)} = \Phi^{\emptyset'}_{f_2(e)}$ for all e. Given a noncomputable Δ^0_2 set Y with a computable approximation (Y_s) the class

$$S_e = \{X \mid Y \leq_T X \oplus W_e\} = \cup_m \cap_{s_0,n} \cup_{s>s_0} \{X \mid \Phi^{X \oplus W_e}_{m,s} \supset Y_s \restriction n\}$$

(where $\Phi_{m,s}$ denotes the finite approximation to the Turing functional Φ_m at stage s) is $\Sigma^0_3(W_e)$. Also, S_e is null in the case that $Y \not\leq_T W_e$, and it is Σ^0_3 in the special case where W_e is low. In fact, there is a computable function g such that if n is a lowness index of a low set W_e then $g(e,n)$ is a Σ^0_3 index of S_e. By the double recursion theorem there exist $e, k \in \mathbb{N}$ such that

$$W_{f_1(g(e,k))} = W_e \ \text{ and } \ \Phi^{\emptyset'}_{f_2(g(e,k))} = \Phi^{\emptyset'}_k = W'_e. \tag{6.16}$$

[10]As usual, we can assume that every number is the index of some Σ^0_3 class. Recall that a (Σ^0_3) index of a Σ^0_3 class S is the index of a Turing machine which, given a triple (i,j,k), outputs a clopen set of reals $V_{i,j,k}$ such that $S = \cup_i \cap_j \cup_k V_{i,j,k}$.

The set W_e is not computable, because otherwise $Y \not\leq_T W_e$, so by [Sti72] S_e is null and therefore $W_{f_1(g(e,k))}$ is promptly simple (by the properties of f_1), a contradiction. If $Y \leq_T W_e \oplus R$ for any Martin-Löf random set R, then $R \in S_e$ and by the properties of f_1 we have $W_e \leq_T R$.

Chapter 7

Effective packing dimension and traceability

The work in this chapter is joint with Rod Downey.

7.1 Introduction

The concern of this chapter is with effective packing dimension. This concept can be traced back to the work of Borel and Lebesgue who studied measure as a way of specifying the size of sets. Carathéodory later generalized Lebesgue measure to the n-dimensional Euclidean space, and this was taken further by Hausdorff [Hau19] who generalized the notion of s-dimensional measure to include non-integer values for s in any metric space. In the Cantor space with the clopen topology, this can be viewed as a scaling of the usual Lebesgue measure by a factor of s, in the sense of

$$\mu_s([\sigma]) = 2^{-s|\sigma|},$$

where $[\sigma]$ is the clopen set generated by σ, and $0 \leq s \leq 1$. This gave rise to the concept of classical Hausdorff dimension, which provided a way of classifying different sets of measure zero, based on the intuition that not all null sets are created equal.

There appeared many other related classical notions of fractional dimensions, such as box-counting dimension and packing dimension. The study of effective notions of randomness and their relationship with the Turing degrees was initiated by the early work of de Leeuw, Moore, Shannon and Shapiro [dLMSS56]. The effective versions of these various notions of fractional dimensions have been studied in connection with randomness. The best known examples of such effective notions are the effective Hausdorff, and effective packing dimensions.

Hausdorff measure talks about covering the set by open balls from the exterior, while packing measure considers filling up a set from the interior. One can effectivize these two notions by looking at covering with Σ^0_1 open sets in the Cantor space with s-measure. This work took a new direction when various authors Lutz [Lut90, Lut03], Staiger [Sta93], Mayordomo [May02], Artheya et al [AHLM04], and Reimann [Rei04] showed that there were simple characterizations of effective Hausdorff and packing dimensions using Kolmogorov complexity. Indeed, the effective Hausdorff dimension of a real A can be written as

$$dim_H(A) = \liminf_{n\to\infty} \frac{K(A\restriction n)}{n},$$

while its dual notion, the effective packing dimension is

$$dim_p(A) = \limsup_{n\to\infty} \frac{K(A\restriction n)}{n}.$$

We also refer the reader to Lutz [Lut00] for a characterization in terms of martingales. We mention here that there is a natural way to define the effective dimension of any countable collection of reals, by looking at the lim sup of the effective dimensions of its members. In particular one can talk about the effective dimension of a Turing degree (or a lower cone with respect to Turing reducibility).

Effective packing dimension is a very natural notion of effective dimension to study; indeed the reals of effective packing dimension 1 can be described as one where "measure meets category". In particular this property is shared by both the Martin-Löf random reals, as well as reals which are sufficiently generic (for instance 2-generic). Consequently the class of reals having effective packing dimension one is both co-meager and of measure 1.

Unlike effective Hausdorff dimension, the notion of effective packing dimension is much more tractable. Fortnow, Hitchcock, Aduri, Vinochandran and Wang [FHA$^+$06], proved that the dimension extraction property was true for effective packing dimension with respect to weak truth table reducibility:

Theorem 7.1.1 (Fortnow et al [FHA$^+$06]). *For every $\varepsilon > 0$ and every A, if $dim_P(A) > 0$, then there is $B \equiv_{wtt} A$ such that $dim_P(B) > 1 - \varepsilon$.*

Hence their result gives a 0-1 law on the effective packing dimension of wtt degrees - this can be only 0 or 1. In contrast, Miller [Mila] recently solved a longstanding question on "broken Hausdorff dimension", where he constructed a Δ^0_2 degree with effective Haudorff dimension $\frac{1}{2}$, which does not compute any real of higher Hausdorff dimension.

It is still open if every degree of effective packing dimension one contains a real of effective packing dimension one, and this seems to be a difficult problem. Our task at hand is less ambitious; we are interested in answering a more general question: which Turing degrees are of effective packing dimension 1? Downey and Greenberg gave a classification in the case of c.e. degrees:

Theorem 7.1.2 (Downey and Greenberg [DG08]). *A c.e. degree contains a real with positive effective packing dimension iff it is array noncomputable.*

Recall that the array computable degrees were the degrees $\mathbf{a}$ such that there is some $f \leq_{wtt} \emptyset'$ which dominates every $\mathbf{a}$-computable function. A degree is array noncomputable if it is not array computable. Their result was related to a theorem of Kummer [Kum96], where he proved a gap phenomenon in the growth of C-complexity. In particular he showed that every c.e. array noncomputable degree contains a set which has infinitely many segments of maximal C-complexity. On the other hand every c.e. array computable set has initial segments with C-complexity as close to $\log n$ as we want. Downey and Greenberg's classification reinforces the fact that array (non-)computability was intimately related to Kolmogorov complexity.

One would naturally conjecture that the above characterization of Downey and Greenberg holds in general. Unfortunately this tempting guess does not work out because there are array computable random degrees (any random hyperimmunefree degree is an example), so the array noncomputable degrees fail to give a characterization. Recall that a set Z is of hyperimmunefree degree, if every function computable from Z is dominated by a computable function. In fact the array noncomputable degrees also fail to give a characterization within the Δ^0_2 degrees because any superlow random real is also array computable.

Greenberg and Downey observed that it was easy to generalize Kummer's Gap Theorem to a notion called c.e. traceability, which is akin to array computability.

Recall that a degree $\mathbf{a}$ is c.e. traceable if there is some computable, nondecreasing and unbounded function h such that for all $f \leq_T \mathbf{a}$ there is a uniformly c.e. sequence $\{T_x\}$ such that for all x, $|T_x| \leq h(x)$ and $f(x) \in T_x$. This has been studied by Zambella [Zam90], Terwijn and Zambella [TZ01] and also Ishmukhametov [Ish97] who showed that in the c.e. degrees, array computability coincided with c.e. traceability. Greenberg and Downey observed that every c.e. traceable set has effective packing dimension 0.

One might now hope that the weaker notion of being not c.e. traceable would give a characterization. The degrees which were not c.e. traceable contain all random degrees, and so the obvious counter-examples for array noncomputability are not relevant. In this chapter we show that this feeble conjecture fails. In Theorem 7.2.1 we first construct a hyperimmunefree and Δ^0_3 example:

Theorem 7.2.1. *There is a Δ^0_3 real A which is of hyperimmune-free degree and not c.e. traceable, such that every real $\alpha \leq_T A$ has effective packing dimension 0.*

Since the degrees containing no real of packing dimension 1 may be thought of as having low algorithmic information content, one might hope to be able to relate this concept with some known lowness class arising in algorithmic randomness. Theorem 7.2.1 says that the low for Schnorr random reals fail to give a characterization even amongst the hyperimmunefree degrees, because the low for Schnorr random reals are all computably traceable and constitute too strong a notion. It is not clear if there is any relationship between the degrees of positive effective packing dimension, and the degrees containing a low for Kurtz random (i.e. the hyperimmunefree and non-d.n.c. degrees).

In Theorem 7.3.1 we show that the property of being not c.e. traceable fails to give a characterization amongst the sets computable from $\emptyset'$:

Theorem 7.3.1. *There is a real $A \leq_T \emptyset'$ which is not c.e. traceable, such that every real $\alpha \leq_T A$ has effective packing dimension 0.*

Finally we ask if there is a combinatorial characterization of the degrees in PD_1. By combinatorial we mean a definition which does not mention K-complexity, nor any randomness notion.

We associate finite strings with code numbers for them. If σ is a finite string of positive length then σ^- denotes the predecessor of σ. We use $|\sigma|$ to denote the length of a finite string, and # to denote the cardinality of a finite set. The rest of our notations are generally standard, and follow Soare [Soa87].

7.2 A Δ^0_3 and hyperimmunefree example

Theorem 7.2.1. *There is a Δ^0_3 real A which is of hyperimmunefree degree and not c.e. traceable, such that every real $\alpha \leq_T A$ has effective packing dimension 0.*

Proof. By Theorem 7.1.1, we only need to ensure that $dim_P(\alpha) \leq \frac{1}{2}$ for every $\alpha \leq_T A$. We build the set A of HIF degree by an oracle construction and we define a total function $g = \Gamma^A$ satisfying the requirements

$$\begin{aligned} \mathcal{P}_e &: \quad g(x) \notin V^e_x \text{ for some } x. \\ \mathcal{N}_e &: \quad \text{if } \Phi^A_e \text{ is total, then } K(\Phi^A_e \restriction x) \leq x/2 \text{ for almost all } x. \end{aligned}$$

We let $\{V^e_x\}_x$ be the e^{th} c.e. trace such that $\#V^e_x < x$ for every e, x. We observe that there are plenty of reals which are of hyperimmunefree degree, but not computably traceable. For instance, any HIF random real will do, but random reals all have effective packing dimension 1. On the other hand, the standard construction of a Δ^0_3 real of HIF degree also makes it computably traceable, so one has to go out of the way to construct such a real directly (see Terwijn's thesis [Ter98]). The basic idea there is to work in a tree T where every node at level x has x branches or successors (one could work in the Cantor space since T is a homeomorphic copy, but it will be notationally more cumbersome). We have to define the total functional $g = \Gamma^X$ externally, by letting $\Gamma^\sigma(|\sigma|) = \sigma$, which will be clearly total along every path in $[T]$. We want to defeat all possible traces for g, and since T has enough splits at each level we could kill off enough branches at a certain level in order to diagonalize against the e^{th} trace. This is reminiscent of a "bushy tree" type construction used to construct minimal d.n.c. degrees. At the same time we will be able obtain an upper bound for the possible values of $\Phi^X_e(n)$ (to force HIF), just by reading it off the tree.

Suppose we now want to combine the above construction with the requirement $\mathcal{N}_e$. Note that $\mathcal{N}_e$ requires us not only to have to keep the initial segment complexity of

A small (which is easy), but rather we need to keep the complexity of Φ_e^A small. This creates an additional difficulty, because in general there is no effective relationship between the length of a segment σ, and the length of the use which produces that segment (i.e. τ such that $\Phi_e^\tau = \sigma$). In particular, we could have very long segments τ such that $\Phi_e^{\tau^\frown i}$ are all different for different values of i, such that $|\Phi_e^{\tau^\frown i}|$ is relatively short. Remember that we have to keep enough successors of τ left on the tree for diagonalization, so we might have to end up issuing many descriptions witnessing $K(\Phi_e^{\tau^\frown i}) \leq \frac{1}{2}|\Phi_e^{\tau^\frown i}|$ for many different i. The number of different i could be too large relative to $|\Phi_e^{\tau^\frown i}|$. The obvious thing to try might be for instance, to choose a longer τ so that $|\Phi_e^{\tau^\frown i}|$ is longer, and hence costs less to describe, but remember that generally at level $|\tau|$ we have to keep at least $|\tau|$ many successors of τ left on the tree (since g is, and in fact has to be, defined externally). So generally looking for a longer τ doesn't help, since this also corresponds to having even more possibilities of $|\Phi_e^{\tau^\frown i}|$ for which we have to issue descriptions.

Our solution to this is to gather a *majority vote*, or consensus. We start with a tree T which has, at level x, a large number of successors say $x2^{L(x)}$ many (for some $L(x)$ to be determined). We will define a computable subtree T_e of T by the following. We may assume that for every x and above every string σ on T we can always force convergence, i.e. find some $\eta \supset \sigma$ on T such that $\Phi_e^\eta(x) \downarrow$. (otherwise we can just take T_e to be the full subtree above some node). First pick a level x_0 which will be the first level in T_e for which we will put up splits. We then pick a length $L(x_0)$ which is very large relative to x_0, and search for strings $\sigma_1 \supset 1^{x_0}1, \sigma_2 \supset 1^{x_0}2, \cdots$ such that $\Phi_e^{\sigma_i}\upharpoonright L(x_0) \downarrow$ for all i. Since there are $x_0 2^{L(x_0)}$ many different σ_i, and only at most $2^{L(x_0)}$ many possibilities for $\Phi_e^{\sigma_i}\upharpoonright L(x_0)$, it follows there is some τ such that $\Phi_e^{\sigma_i}\upharpoonright L(x_0) = \tau$ for at least x_0 many different σ_i. Leave 1^{x_0} on T_e, as well as the extensions σ_i which voted for the majority, and kill all other incomparable nodes. We then move on to the next level x_1. This ensures that the tree T_e still has enough splits at infinitely many levels (so that we can proceed with diagonalization for other $\mathcal{P}$ requirements), but yet we are able to restrict the possibilities for Φ_e.

Suppose $\sigma_{i_1}, \cdots, \sigma_{i_{x_0}}$ were the extensions of 1^{x_0} which survived on T_e. After we pick x_1 we will repeat the "majority vote" strategy separately above each σ_{i_j} to ensure we have enough splits left on T_e at level x_1, and kill off all other incomparable nodes. We now have $\eta_1, \eta_2, \cdots, \eta_{x_0} \supset \Phi_e^{\sigma_{i_0}} \upharpoonright L(x_0)$ where η_j was voted by the majority of nodes extending σ_{i_j}. To satisfy $\mathcal{N}_e$ we need to issue descriptions for all possible segments for Φ_e. There is one segment of length $L(x_0)$, x_0 many segments of length $L(x_1)$, $x_0 x_1$ many segments of length $L(x_2)$ and so on, for us to describe. As long L is chosen such that $x_0 \cdots x_k 2^{-L(x_k)/2}$ is small, then we will be fine. The exact details are supplied in the formal construction.

Formal construction. For each $x \in \mathbb{N}$ and rational $0 < \delta < 1$, we define $\ell(x, \delta)$ to be the least number larger than $4 - 2 \log \frac{\delta}{x^x}$, so that $x^x 2^{-\frac{1}{2}\ell(x,\delta)+2} < \delta$ holds. This seemingly bizarre choice for ℓ will become clear later; it is simply a huge number that bounds everything we need. Define the computable sequence of functions $L_1, L_2, \cdots$ inductively by

$$L_1(x) = \ell(x, 2^{-x}), \quad L_{n+1}(x) = \ell(x 2^{L_1(x)+\cdots+L_n(x)}, 2^{-x}),$$

for all positive $x \in \mathbb{N}$. It is a simple exercise to show that $L_n(x)$ is increasing in both variables. In this proof, a tree is defined to be a partial computable function $T : \omega^{<\omega} \mapsto \omega^{<\omega}$, such that $\sigma \subset \tau \ \wedge \ T(\tau) \downarrow \Rightarrow T(\sigma) \downarrow \subset T(\tau)$, and incomparable strings map to incomparable strings. A tree T is said to be *crowded* if it satisfies the following

1. $T(\emptyset) \downarrow$.
2. if $T(\sigma) \downarrow$, then $T(\sigma^\frown i) \downarrow$ for all $i = 1, \cdots, x 2^{L_1(x)+\cdots+L_{|\sigma|+1}(x)}$, where $x = |T(\sigma)|$.
3. if $i \neq j$ then $T(\sigma^\frown i) \upharpoonright 1 + |T(\sigma)| \neq T(\sigma^\frown j) \upharpoonright 1 + |T(\sigma)|$.
4. if $T(\sigma) \downarrow$ and $T(\tau) \downarrow$ and $|\sigma| = |\tau|$, then $|T(\sigma)| = |T(\tau)|$.
5. T is defined nowhere else, and is being built up from $\emptyset$ by applying rules (1) to (4).

Condition (5) compacts the tree in the sense that we eliminate the situation where we have $T(0) \downarrow$ and $T(2) \downarrow$ but $T(1) \uparrow$. A crowded tree generalizes simultaneously, to the non-binary case, both the ideas of having a "perfect" tree, as well as having enough branches at infinitely many levels.

Generate the tree T by letting $T(\emptyset) = 1$, and inductively if $T(\sigma) \downarrow$ then let $T(\sigma^\frown i) = T(\sigma)^\frown i$ for $i = 1, \cdots, x 2^{L_1(x)+\cdots+L_x(x)}$ where $x = |\sigma| + 1$. T is in some sense the identity tree, and is clearly crowded. We say that T is the *full crowded tree*. Since $Ran(T)$ is computably homeomorphic to the Cantor space, we will construct A as an infinite path through $Ran(T)$. If T is a tree, we let $[T]$ be the set of all infinite strings X such that there are infinitely many $\tau \in Ran(T)$ such that $\tau \subset X$. Equivalently, $[T] = \{X : \exists Y \forall n T(Y \upharpoonright n) \subset X\}$. If P and Q are trees, then we say that $P \subseteq Q$ if for every σ such that $P(\sigma) \downarrow$, we have some η such that $P(\sigma) = Q(\eta)$.

The functional Γ is defined as follows: $\Gamma^\sigma(x) \downarrow = \sigma$ if σ is on T and $|\sigma| = x$. Clearly Γ^X is total for any path $X \in [T]$. During the construction, at each stage $s+1$ we will define a crowded subtree $T_{s+1} \subset T_s$, and let $A = \cup_s T_s(\emptyset)$. For each s and k note that $T_s(1^k)$ is always convergent, and if T_s is crowded, then $|T_s(\sigma)| = |T_s(1^k)|$ for every convergent $|\sigma| = k$. If P is a tree we say that σ *is on* P, or equivalently $\sigma \in P$ to mean that $\sigma \subseteq P(\eta)$ for some η. If P is crowded, then $Ran(P)$ is computable as a set of nodes, so that the relation $\sigma \in P$ is a computable relation (given an index for P).

If P is a crowded tree and σ is on P, then we define $\tilde{P}$ as the *crowded subtree of P above σ* by the following. Look for the minimal η such that $P(\eta) \supseteq \sigma$. Let $\tilde{P}(\tau) = P(\eta^\frown\tau)$ for all τ, and then we chop off the superfluous branches, i.e. restrict the domain of $\tilde{P}$ sufficiently so as to satisfy condition (2). It is clear that $\tilde{P}$ is crowded as well. This is where we use the idea that crowded trees are "perfect" in some sense; at any point in the construction we can just extract P above any σ and still end up with a crowded tree, and this makes no sense if, for instance, P contains dead ends.

The construction. At stage $s = 0$ we let $T_0 = T$. At stage $s = 3e > 0$, we satisfy $\mathcal{N}_e$. Ask if it is the case that $(\forall x \forall \sigma \in T_{s-1})(\exists \tau \supset \sigma)(\tau \in T_{s-1} \ \wedge \ \Phi_e^\tau(x) \downarrow)$. If the answer is no, find a counterexample σ on T_{s-1}, and let T_s be the crowded subtree of T_{s-1} above σ. If the answer is yes, we will define both T_s and a computable tree P_s by the following. The idea is that $[P_s]$ is a Π^0_1 class (with very few splits) and containing all possibilities for Φ_e^A.

First let $\eta \neq \emptyset$ be the first string found such that $\Phi_e^{T_{s-1}(\eta)}(0) \downarrow$ and set $T_s(\emptyset) = T_{s-1}(\eta)$. Next, assume that $T_s(\sigma)$ has been defined up till level k, i.e. for all relevant $|\sigma| \leq k$. Let $N := \{\sigma : |\sigma| = k \ \wedge \ T_s(\sigma) \downarrow\}$. Assume that inductively we have the properties

- T_s is crowded so far
- for every $\sigma \in N$, $T_s(\sigma)$ has at least $x2^{L_1(x)+\cdots+L_{k+2}(x)}$ many successors on T_{s-1}, where $x = |T_s(1^k)|$
- for every $\sigma \in N$, $P_s(\sigma^-)\downarrow$ and $\Phi_e^{T_s(\sigma)} \supseteq P_s(\sigma^-)$.

For each $\sigma \in N$, do the following. Since σ has at least $x2^{L_1(x)+\cdots+L_{k+2}(x)}$ many successors on T_{s-1}, we label these successors by $T_s(\sigma)^\frown n_1, T_s(\sigma)^\frown n_2, \cdots$. For each $i \leq x2^{L_1(x)+\cdots+L_{k+2}(x)}$ find the first string $\sigma_i \supset T_s(\sigma)^\frown n_i$ on T_{s-1}, such that $\Phi_e^{\sigma_i}\restriction L_{k+2}(x)\downarrow\supset P_s(\sigma^-)$. There must be some τ of length $L_{k+2}(x)$ such that $\tau \supset P_s(\sigma^-)$ and we have at least $x2^{L_1(x)+\cdots+L_{k+1}(x)}$ many values of i such that $\Phi_e^{\sigma_i}\restriction L_{k+2}(x) = \tau$. Take the first $x2^{L_1(x)+\cdots+L_{k+1}(x)}$ many such i, and arrange them in increasing order ($i_1 < i_2 < \cdots$) and for each $j \leq x2^{L_1(x)+\cdots+L_{k+1}(x)}$ we set $T_s(\sigma^\frown j) = T_{s-1}(\nu)$ where ν is some string such that $T_{s-1}(\nu) \supset \sigma_{i_j}$. Also set $P_s(\sigma) = \tau$. Repeat for each $\sigma \in N$, and we may assume (by choosing a longer ν) that $|T(\sigma^\frown j)|$ is constant for all $\sigma \in N$ and all j, and that $T_s(\sigma^\frown j)$ has enough successors on T_{s-1}.

Now assume we are at stage $s = 3e+1$, and T_{s-1} is crowded. We want to satisfy $\mathcal{P}_e$. Pick the least $i \leq 1 + |T_{s-1}(\emptyset)|$ such that $T_{s-1}(i)\restriction 1 + |T_{s-1}(\emptyset)| \notin V^e_{1+|T_{s-1}(\emptyset)|}$, and let T_s be the crowded subtree of T_{s-1} above $T_{s-1}(i)\restriction 1 + |T_{s-1}(\emptyset)|$.

Finally assume we are at stage $s = 3e+2$. We run the usual hyperimmunefree strategy. Ask if $(\forall x \forall \sigma \in T_{s-1})(\exists \tau \supset \sigma)(\tau \in T_{s-1} \wedge \Phi_e^\tau(x)\downarrow)$. If the answer is no, find a counter example σ on T_{s-1}, and let T_s be the crowded subtree of T_{s-1} above σ. If the answer is yes, we define T_s by the following. Set $T_s(\emptyset) = T_{s-1}(\emptyset)$. Suppose $T_s(\sigma)$ has been defined for all σ of length $\leq k$, and that $T_s(\sigma)$ has at least $x2^{L_1(x)+\cdots+L_{|\sigma|+1}(x)}$ many successors where $x = |T_s(1^{|\sigma|})|$ on T_{s-1}. Label them $T_s(\sigma)^\frown n_0, T_s(\sigma)^\frown n_1, \cdots$. For each $|\sigma| = k$ and $i \leq x2^{L_1(x)+\cdots+L_{k+1}(x)}$, we set $T_s(\sigma^\frown i)$ to be $T_{s-1}(\eta)$ for some η such that $T_{s-1}(\eta) \supset T_s(\sigma)^\frown n_i$ and $\Phi_e^{T_{s-1}(\eta)}(k)\downarrow$. Once again we may assume that $|T_s(\sigma^\frown i)|$ are all equal for all combinations of σ and i, by choosing a longer η if necessary.

Verification. The construction produces a sequence of trees $T_0 \supseteq T_1 \supseteq \cdots$, where the sequence of indices for the trees is computable in $\emptyset''$. Hence A is Δ^0_3 and $A \in [T_s]$ for every s. We verify that for every s, T_s is crowded and every infinite path through T_s (in particular A) has the desired properties. Assume that T_{s-1} is crowded, and let $s = 3e > 0$. If the answer to the Π_2-question is no, then T_s is clearly crowded and there is some x such that for every $X \in [T_s]$, we have $\Phi_e^X(x)\uparrow$. Suppose on the other hand that the answer given was yes. Observe that the inductive definition of T_s done level by level is well-defined, and maintains the property of being crowded at each finite level. Observe also that P_s is a tree with the following properties:

1. $dom(P_s) = dom(T_s)$.
2. for every σ, $|P_s(\sigma)| = L_{|\sigma|+2}(|T_s(\sigma)|)$.
3. for every infinite path $X \in [T_s]$, we have Φ_e^X total and is an infinite path through $[P_s]$.

We claim that every infinite path through $[P_s]$ has effective packing dimension at most $\frac{1}{2}$, because P_s is extremely sparse. This is similar to the proof that every Π_1^0 class of measure 0 contains no random path. To see this, we enumerate a Kraft-Chaitin set, where we enumerate axioms $\langle \eta, \frac{1}{2}|\eta|\rangle$ for every string $\eta \supset P_s(\emptyset)$ on P_s. We let $h_k := |T_s(1^k)|$, and $S_k := \{\sigma \in dom(T_s) \mid |\sigma| = k\}$. We can show easily that $\#S_0 = 1$ and $\#S_k \leq (h_{k-1}2^{L_1(h_{k-1})+\cdots+L_k(h_{k-1})})^k$ for any $k > 0$. It follows then that the total cost of all these requests is bounded above (very generously) by

$$\sum_{\eta\in dom(P_s)} \sum_{x \geq |P_s(\eta)|} 2^{-\frac{1}{2}x} \cdot \#\text{ of successors of } P_s(\eta)$$
$$\leq \sum_{\eta \in dom(P_s)} 2^{-\frac{1}{2}|P_s(\eta)|+2} \cdot \#\text{ of successors of } P_s(\eta).$$

Since T_s is crowded, every level looks the same, hence we can express the above sum as

$$\begin{aligned}
&\sum_k \#S_k \cdot 2^{-\frac{1}{2}L_{k+2}(h_k)+2} \cdot h_k 2^{L_1(h_k)+\cdots+L_{k+1}(h_k)} \\
\leq\ &\sum_k 2^{-\frac{1}{2}L_{k+2}(h_k)+2} \cdot (h_k 2^{L_1(h_k)+\cdots+L_{k+1}(h_k)})^{k+1} \\
\leq\ &\sum_k 2^{-h_k} < 1 \text{ (by the choice of } \ell).
\end{aligned}$$

This shows that every infinite path through $[P_s]$ has effective packing dimension at most $\frac{1}{2}$.

Next, we let $s = 3e+1$. In the construction we will be able to find the required i, since $\#V^e_{1+|T_{s-1}(\emptyset)|} \leq |T_{s-1}(\emptyset)|$. We have $\Gamma^X(1 + |T_{s-1}(\emptyset)|) = T_{s-1}(i)\upharpoonright 1 + |T_{s-1}(\emptyset)| \notin V^e_{|T_{s-1}(\emptyset)|+1}$ for every infinite path X through $[T_s]$.

Finally, consider $s = 3e+2$. If the answer to the Π_2-question is no, then again T_s is clearly crowded and for every $X \in [T_s]$, Φ^X_e is not total. If the answer given was yes, then it is clear that T_s is also crowded, and that furthermore for every infinite path X through $[T_s]$, we must have that for every x, $\Phi^X_e(x)$ must converge with use at most $|T_s(1^{x+1})|$. One can then proceed to generate a computable bound for all possible Φ^X_e.

Since $A \in [T_s]$ for every s, it follows that A has all the properties we require. $\square$

7.3 A Δ^0_2 example

In this section we provide an effective version of the proof of Theorem 7.2.1. By doing so we prove that:

Theorem 7.3.1. *There is a Δ^0_2 set A which is not c.e. traceable, such that every real $\alpha \leq_T A$ has effective packing dimension 0.*

Proof. We work in a variation of the Cantor space which is finitely branching. We build a path g in this space by finite extension where $g = \cup_e \sigma_e$. Again we only need to ensure that $dim_P(\alpha) \leq \frac{1}{2}$ for every $\alpha \leq_T g$. We ensure that the following requirements are met:

$$\mathcal{P}_e \quad : \quad g(x) \notin V^e_x \text{ for some } x.$$
$$\mathcal{N}_e \quad : \quad \text{if } \alpha_e = \Phi^g_e \text{ is total, then } K(\alpha_e\upharpoonright x) \leq x/2 \text{ for almost all } x.$$

Then the set A can be taken to be say, the graph of g. Again we let $\{V^e_x\}_x$ to be the e^{th} c.e. trace, with identity bound. We maintain a sequence of computable trees $T_0 \supset T_1 \supset \cdots$ and build g as a path through $\cap_e T_e$. At every stage s we use $\emptyset'$ as an oracle to search through the tree T_e, and when we discover that the tree is not total we change our mind on T_e. This will resemble a finite injury with oracle $\emptyset'$, and is similar to the way in which Sacks' construction of a minimal degree below $\emptyset'$ is a $\emptyset'$-effective version of Spector's construction.

We retain most of the notations and parameters of the previous Δ^0_3 construction. Like in Sacks' proof, we have to allow our crowded trees to be "partial", in the sense that they may now contain dead ends. To wit, we now declare that a tree is *crowded*, if

1. If $T(\sigma)\downarrow$, then either $T(\sigma^\frown i)\downarrow$ for all $i = 1, \cdots, x2^{L_1(x)+\cdots+L_{|\sigma|+1}(x)}$, where $x = |T(\sigma)|$ or else $T(\sigma^\frown i)\uparrow$ for every i.
2. If $i \neq j$ then $T(\sigma^\frown i)\restriction 1 + |T(\sigma)| \neq T(\sigma^\frown j)\restriction 1 + |T(\sigma)|$ whenever they converge.
3. T is defined nowhere else, and is built up using the above rules.

The difference now is that we allow $T(\sigma)\downarrow$ but has no successors. We also have to allow $T(\sigma)$ and $T(\eta)$ to be of different lengths when $|\sigma| = |\eta|$, because we might not be able to find convergent strings densely; as we will see this will have no serious impact on the calculations. If $T(\sigma)\downarrow$ but $T(\sigma^\frown i)\uparrow$ for every i then we say that $T(\sigma)$ *has no successors.*

If T is a crowded tree and σ is on T, we let $Full(T, \sigma)$ be the crowded subtree of T above σ as before. An index for $Full(T, \sigma)$ can be found effectively in σ and an index for T. The second operation is the *majority e-subtree above* σ, denoted as $Maj(e, T, \sigma)$. This is the tree Q defined by the following. We also define a partial computable tree P together with Q:

First let $\gamma \neq \emptyset$ be the first string found such that $T(\gamma)\downarrow\supset \sigma$ and $\Phi_e^{T(\gamma)}(0)\downarrow$. If no such string is found then $Q(\emptyset)\uparrow$, otherwise set $Q(\emptyset) = T(\gamma)$. Next, assume that $Q(\eta)$ has been defined, and that inductively we have the properties

- Q is crowded so far,
- $Q(\eta) = T(\gamma)$ for some $|\gamma| > |\eta|$,
- $P(\eta^-)\downarrow$ and $P(\eta^-) = \Phi_e^{Q(\eta)}\restriction L_{|\eta^-|+2}(|Q(\eta^-)|)$.

We now compute $Q(\eta^\frown i)$ for an appropriate number of i's. Let $x = |Q(\eta)|$. First wait for $T(\gamma^\frown i)\downarrow$ for every $i \leq x2^{L_1(x)+\cdots+L_{|\gamma|+1}(x)}$. For each i find the first string $\gamma_i \supseteq \gamma^\frown i$ such that $T(\gamma_i)\downarrow$ and $\Phi_e^{T(\gamma_i)}\restriction L_{|\eta|+2}(x)\downarrow$. Necessarily we must have $\Phi_e^{T(\gamma_i)}\restriction L_{|\eta|+2}(x) \supset P(\eta^-)$. There must be some τ of length $L_{|\eta|+2}(x)$ such that $\tau \supset P(\eta^-)$ and we have at least $x2^{L_1(x)+\cdots+L_{|\eta|+1}(x)}$ many values of i such that $\Phi_e^{T(\gamma_i)}\restriction L_{|\eta|+2}(x) = \tau$. Take the first $x2^{L_1(x)+\cdots+L_{|\eta|+1}(x)}$ many such i's and define $Q(\eta^\frown i) = T(\gamma_i)$. Observe that the three properties still hold for $Q(\eta^\frown i)$. This ends the definition of Q. Note that we may assume that $|Q(\eta)| \neq |Q(\eta')|$ whenever $|\eta| = |\eta'|$ (by searching further along T).

An index for Q is obtained effectively from e, T and σ. In fact the following holds:

Lemma 7.3.2. *If T is crowded and σ is on T, then*

(i) Q is crowded,

(ii) every $\alpha = \Phi_e^X$ for $X \in [Q]$ has $dim_P(\alpha) \leq \frac{1}{2}$,

(iii) if $Q(\emptyset) \uparrow$ then there is no $X \supset \sigma$ such that $X \in [T]$ and Φ_e^X is total,

(iv) if $Q(\eta) \downarrow = T(\gamma)$ but has no successors on Q, then either

- *it has no successors on T, or else*
- *there is some k such that $T(\gamma^\frown k) \downarrow$, and for every $X \supset T(\gamma^\frown k)$, $X \in [T]$, we have Φ_e^X is not defined somewhere below $L_{|\eta|+2}(|Q(\eta)|)$.*

Proof. The others are straightforward, so we only prove (ii). Observe that P is closed under initial segments, and in this case satisfies similar properties as before:

1. for every η, $P(\eta) \downarrow \Leftrightarrow Q(\eta) \downarrow$ and has successors,
2. for every η, $|P(\eta)| = L_{|\eta|+2}(|Q(\eta)|)$,
3. for every infinite path $X \in [Q]$, we have Φ_e^X total and is an infinite path through $[P]$.

Since P is a partial computable tree, hence the set of strings τ on P is a c.e. set. We then enumerate a KC-set $\{\langle \tau, \frac{1}{2}|\tau| \rangle : \tau \supset P(\emptyset) \text{ is on } P\}$. We need to show that the total size of these requests is bounded. Just as before, the size of these requests is bounded above by

$$\sum_{\eta \in dom(P)} 2^{-\frac{1}{2}|P(\eta)|+2} \cdot \#\text{ of successors of } P(\eta) \leq \sum_{\eta \in dom(P)} 2^{-|Q(\eta)|},$$

where $o(\eta) = x2^{L_1(x)+\cdots+L_{|\eta|+1}(x)}$ and $x = |Q(\eta)|$. That is $o(\eta)$ is the number of successors of $Q(\eta)$. Since we assumed that $|Q(\eta)| \neq |Q(\eta')|$ whenever η, η' are of the same length, we can reduce the sum to

$$\sum_k \sum \{2^{-|Q(\eta)|} : |\eta| = k, \eta \in dom(Q)\} \leq \sum_k 2^{-k+1} < \infty.$$

This shows that every infinite path through $[P]$ has effective packing dimension at most $\frac{1}{2}$, and shows that (ii) holds. $\square$

Construction of g. We build g by finite extension. At each stage s, $T_e[s]$ denotes the tree which we use to satisfy requirement $\mathcal{N}_e$. Let T be the full crowded tree in Theorem 7.2.1. By convention $T_{-1} = T$. At stage $s = 0$ we initialize T_e for every e, and let $\sigma_0 = \langle\rangle$. At $s > 0$ we assume that inductively we have the following:

(1) $T \supset T_0[s] \supset \cdots$, and are all crowded,

(2) $\eta_0 \geq \eta_1 \geq \cdots$ such that $\sigma_{s-1} = T_0(\eta_0)[s] = T_1(\eta_1)[s] = \cdots$.

We find the least $e \geq 0$ such that T_e is defined and σ_{s-1} has no successors on T_e. If e exists, then T_e must have been obtained from T_{e-1} by taking the e-majority subtree operation. We claim that $\emptyset'$ can compute some $\rho \supset \eta_{e-1}$ such that $T_{e-1}(\rho)\downarrow$, and for every $X \supseteq T_{e-1}(\rho)$ and $X \in [T_{e-1}]$, Φ_e^X is not total.

First go through each k and ask if there is some string $\rho_k \supseteq \eta_{e-1}{}^\frown k$ such that $T_{e-1}(\rho_k)\downarrow$, and $\Phi_e^{T_{e-1}(\rho_k)} \upharpoonright L_{|\eta_e|+2}(|\sigma_{s-1}|)\downarrow$. If the answer no for some k, then take $\rho = \eta_{e-1}{}^\frown k$. If ρ_k is found for every k, then by Lemma 7.3.2(iv), there will be some k such that $[T_{e-1}(\rho_k)] \cap [T_{e-1}] = \emptyset$. By compactness we can search for it using $\emptyset'$. Let $\rho = \rho_k$. In any case once ρ is found we let $\sigma_s = T_{e-1}(\rho)$. We keep $T_0, \cdots, T_{e-1}$ and set T_e to be the $Full(T_{e-1}, \sigma_s)$. Initialize all $T_{e+1}, T_{e+2}, \cdots$. Adjust $\eta_0, \cdots, \eta_e$ accordingly.

Suppose on the other hand e does not exist. Let e_0 be the largest such that $T_{e_0}\downarrow$. Hence $T_{e_0}(\eta_{e_0})$ has at least $1 + |\sigma_{s-1}|$ many successors on T_{e_0}; since $T_{e_0}(\eta_{e_0}{}^\frown i)$ are all different at the $|\sigma_{s-1}|^{th}$ bit, we pick some i so that $T_{e_0}(\eta_{e_0}{}^\frown i)(|\sigma_{s-1}|) \notin V^{e_0}_{|\sigma_{s-1}|}$. Let $\rho = T_{e_0}(\eta_{e_0}{}^\frown i)$. Ask if $Maj(e_0 + 1, T_{e_0}, \rho)(\emptyset)\downarrow$. If the answer is yes, let T_{e_0+1} be $Maj(e_0 + 1, T_{e_0}, \rho)$ and $\sigma_s = T_{e_0+1}(\emptyset)$. Otherwise the answer is no; we let $T_{e_0+1} = Full(T_{e_0}, \rho)$ and let $\sigma_s = \rho$. Define $\eta_0, \cdots, \eta_{e_0+1}$ appropriately.

Verification. Clearly for every s, $\sigma_{s+1} \supsetneq \sigma_s$. Let $g = \cup\sigma_s$ and hence $g \leq_T \emptyset'$. It is also easy to see that for each e, T_e is initialized finitely often and receives a final definition; let $\tilde{T}_e$ denote this. It is clear that $g \in [\tilde{T}_e]$ for every e. Now fix an e. We claim that $\mathcal{P}_e$ is satisfied. There is a least stage s where T_{e+1} receives a definition; at that stage we ensured that $g(|\sigma_{s-1}|) \not\in V^e_{|\sigma_{s-1}|}$. Now we verify that $\mathcal{N}_e$ is satisfied. Suppose that Φ_e^g is total. Let s be the stage where T_e is defined as $\tilde{T}_e$. Suppose at s we found that e is the least such that T_e is defined and σ_{s-1} has no successors on T_e. However σ_s is defined such that any infinite extension $X \supset \sigma_s$, where $X \in [\tilde{T}_{e-1}]$ has the property that Φ_e^X is not total. Since Φ_e^g is total, hence at s the second scenario in the construction applies, where $e_0 + 1 = e$ and ρ is on $\tilde{T}_{e-1}$. By Lemma 7.3.2(iii) we must have $Maj(e, \tilde{T}_{e-1}, \rho)(\emptyset) \downarrow$. Hence $\tilde{T}_e = Maj(e, \tilde{T}_{e-1}, \rho)$. By Lemma 7.3.2(ii), we have $dim_P(\Phi_e^g) \leq \frac{1}{2}$. $\square$

Chapter 8

Finite variations on Martin-Löf randomness

The work in this chapter is joint with Paul Brodhead and Rod Downey.

8.1 Basic results

We introduce the following variations on ML-randomness, by restricting the cardinality of the tests. $\#W$ denotes the cardinality of W. $|\sigma|$ denotes the length of a finite string σ. We work in the Cantor space 2^ω with the usual clopen topology. The basic open sets are of the form $[\sigma]$ where σ is a finite string, and $[\sigma] = \{X \in 2^\omega \mid X \supset \sigma\}$. We fix some effective coding of the set of finite strings, and we freely identify finite strings with their code numbers. We denote $[W] = \cup\{[\sigma] : \sigma \in W\}$ as the Σ_1 open set associated with the c.e. set W. $\mu([W])$ denotes Lebesgue measure, and we write $\mu(W)$ instead of $\mu([W])$.

As it is customary, we sometimes denote $P[s]$ as the value of the expression P evaluated at stage s. This is not to be confused with $[\sigma]$ which denotes the neighbourhood of σ. We use P_s as much as possible, and limit the appearance of $P[s]$ to situations which are notationally cumbersome.

Definition 8.1.1. (a) A Martin-Löf test is a uniformly c.e. sequence $\{U_n\}_{n\in\omega}$ of Σ^0_1 sets $U_n \subset 2^{<\omega}$ such that $\mu(U_n) < 2^{-n}$.

(b) A Martin-Löf test $\{U_n\}_{n\in\omega}$ is finitely bounded (FB) if each U_n is finite.

(c) A Martin-Löf test $\{U_n\}_{n\in\omega}$ is computably bounded (CB) if there is some total computable function f such that $\#U_n \leq f(n)$ for every n.

(d) A real $X \in 2^\omega$ passes a (CB-, FB-) Martin-Löf test $\{U_n\}_{n\in\omega}$ if $X \notin \cap_n[U_n]$.

Definition 8.1.2. A real $X \in 2^\omega$ is *computably bounded random (CBR)* if X passes every CB-Martin-Löf test. X is *finitely bounded random (FBR)* if it passes every FB-Martin-Löf test.

We will sometimes refer to a CB-Martin-Löf test simply as a CB-test (similarly for a FB-Martin-Löf test). There are no universal CB- and FB-tests because every CB- and FB-test has a computable set passing it. These two notions of randomness are weaker than Martin-Löf randomness, although they imply Kurtz randomness. Over the hyperimmunefree degrees, all of these notions collapse, but on the other hand there are hyperimmune degrees which do not contain a CBR. The obvious implications are:

$$\begin{tikzcd}
ML\text{-random} \arrow[r] \arrow[dr] & FBR \arrow[r] & CBR \arrow[r] & \text{Kurtz random} \\
& \text{Schnorr random} \arrow[urr] & &
\end{tikzcd}$$

Our first task is to separate these notions and to show non-inclusions apart from the obvious ones above. We first show that over the Δ^0_2 sets, the notions of FBR and Martin-Löf -randomness coincide and that they differ on the Δ^0_3 sets. The reason for this is because $\emptyset''$ is able to separate a finite c.e. set from an infinite one; in fact the set $\{e : \#W_e = \infty\}$ is Π^0_2-complete.

Theorem 8.1.3. *(i) Suppose $Z \leq_T \emptyset'$. Then Z is random iff Z is FBR.*

(ii) There is some $Z \leq_T \emptyset''$ such that Z is FBR but not random.

Proof. (i): Given an approximation Z_s of Z. Let us suppose that $\{U_x\}$ is the universal ML-test where $Z \in \cap_x[U_x]$. Enumerate an FB-test $\{V_x\}$ by the following: at stage s, enumerate into V_x, the string $Z_s\restriction n$ for the least n such that $Z_s\restriction n \in U_x[s]$. Then, $\{V_x\}$ is uniformly c.e., where $\mu(V_x) \leq \mu(U_x) < 2^{-x}$ for all x. Clearly $Z \in [V_x]$ for all x. We know $Z\restriction n \in U_x$ for some least n, and let s be a stage such that $Z_s\restriction n$ is correct and $Z\restriction n$ has appeared in $U_x[s]$. Then, $Z\restriction n$ will be in V_x by stage s, and we will never enumerate again into V_x after stage s.

(ii): We build $Z = \cup_s \sigma_s$ by finite extension. Let $\{U_x\}$ be the universal ML-test, and $\{U_x^e\}_x$ be the e^{th} ML-test. Assume we have defined σ_s, where for all $e < s$, we have

- all infinite extensions of σ_s are in U_e,
- if $\#U_x^e < \infty$ for all x, then there is some k such that no infinite extension of σ_s can be in U_k^e.

Now we define $\sigma_{s+1} \supset \sigma_s$. Firstly, find some $\tau \supseteq \sigma_s$ such that all infinite extensions of τ are in U_s; such τ exists because $\{U_e\}$ is universal. Let $k = |\tau|$. Next, ask if $\#U_k^s < \infty$. If not, let $\sigma_{s+1} = \tau^\frown 0$ and we are done. If yes, then figure out exactly the strings ρ_i such that $[U_k^s] = \cup\{[\rho_1], [\rho_2], \cdots, [\rho_n]\}$. We cannot have $[U_k^s] \supseteq [\tau]$ since $\mu(U_k^s) < 2^{-k}$, so there has to be some $\sigma_{s+1} \supset \tau$ such that $[\sigma_{s+1}] \cap [U_k^s] = \emptyset$, by the finiteness of U_k^s. We can figure σ_{s+1} out effectively from $\rho_1, \rho_2, \cdots, \rho_n$. Clearly the properties above continue to hold for σ_{s+1}. All questions asked can be answered by the oracle $\emptyset''$. □

Note that there is no way of making $\{V_x\}$ computably bounded in (i), even if $Z \leq_{tt} \emptyset'$. It is easy to construct a low c.e. real which is CBR, while from Theorem 8.1.4 below, no superlow c.e. real can be CBR. Hence CB-randomness and FB-randomness differ even on the c.e. reals. This is somewhat surprising because on the c.e. degrees, being random typically implies being computationally complex; for instance, Schnorr random implies highness, while being Martin-Löf random already implies wtt-completeness.

The above demonstrates that CBR reals can be computationally weak in terms of the jump operator. Our next result shows that CB-randomness is still sufficiently strong a notion to exclude being traceable. Recall that jump traceability is a weak computational property which is akin to superlowness; the difference is that for jump traceability we only have to specify *possibilities* rather than an *approximation* for the values of $J^A(x)$. In fact we show that:

Theorem 8.1.4. *No CB-random is c.e. traceable.*

Proof. Suppose that A is c.e. traceable, and that A is coinfinite (otherwise it is trivial). We define the functional Φ by evaluating $\Phi^X(n)$ as σ where $\sigma \subset X$ is the shortest string such that $\#\{k : \sigma(k) = 1\} = 2n$, for any X and n. Since Φ^A is total, there is a c.e. trace $\{T_x\}_{x\in\mathbb{N}}$, such that $\#T_x \leq x$ and $\Phi^A(x) \in T_x$ for every x. We define the CB-test $\{U_x\}$ by the following: we enumerate σ into U_x if $|\sigma| \geq 2x$ and $\sigma \in T_x$. Then $\#U_x \leq x$ and $\mu(U_x) \leq x2^{-2x} < 2^{-x}$ for every x and $A \in \cap_x[U_x]$. □

Thus, it is not true that every hyperimmune degree contains a CBR, which separates it from the Kurtz randoms. We next investigate the connection between CB-randomness and effective dimensions.

Theorem 8.1.5. *Every incomplete c.e. real which is CB-random is not of d.n.c. degree (hence has effective Hausdorff dimension 0). However every CB-random real is of effective packing dimension* 1.

Proof. The first statement follows from Arslanov's completeness criterion. For the second statement, suppose $K(\alpha\restriction n) \leq cn$ for all $n \geq N$ for some $N \in \mathbb{N}$ and $c < 1$ is rational. Fix a computable increasing sequence of natural numbers $\{n_i\}$ all larger than N, such that $n_i > \frac{i}{1-c}$ for all i. Now define a CB-test $\{V_i\}$ by the following: $V_i := \{\sigma \in 2^{n_i} \mid K(\sigma) \leq cn_i\}$. Here we have $\#V_i \leq 2^{cn_i}$. □

Next we show there is a Δ^0_3 FB-random real which is not of d.n.c degree. Kjos-Hanssen, Merkle and Stephan [KHMS06a] characterized the sets of d.n.c degree with the *autocomplex* sets. A set A is called autocomplex if $\forall n(K(A\restriction n) \geq h(n))$ for some total, nondecreasing and unbounded $h \leq_T A$. Hence FBR reals can infinitely often have low initial prefix-free complexity.

Theorem 8.1.6. *There is a Δ^0_3 FB-random which is not of d.n.c degree.*

Proof. We modify the proof of Theorem 8.1.3. We need to satisfy for each e, the statement: if Φ^Z_e is total, then $\exists x(\Phi^Z_e(x) = \varphi_x(x))$. It is easy to use the Recursion Theorem to obtain a total computable function p, such that for every pair $\langle\rho, e\rangle$, $\varphi_{p(\rho,e)}(p(\rho, e))$ outputs $\Phi^\tau_e(p(\rho, e))$ where τ is the first string found to extend ρ such that $\Phi^\tau_e(p(\rho, e)) \downarrow$. We modify the construction of Theorem 8.1.3 in the following way. At stage s after we have defined $\sigma_s \subset Z$, we search for τ in the following way. Ask if $\varphi_{p(\sigma_s,s)}(p(\sigma_s, s)) \downarrow$. If the answer is no, we let $\tau = \sigma_s$; clearly $Z \supset \tau$

cannot converge on $\Phi_s^Z(p(\sigma_s, s))$. If the answer is yes, we find the $\tau \supset \sigma_s$ such that $\Phi_s^\tau(p(\sigma_s, s)) = \varphi_{p(\sigma_s,s)}(p(\sigma_s, s))$. The second part of the proof remains the same. $\square$

Again note that the finiteness of the FB-tests is important. This makes it possible to easily construct a FB-random by finite extension.

Question 8.1.7. *If $A \leq_T B$ and A is CB-random, must $deg_T(B)$ contain a CB-random?*

Question 8.1.8. *Are there characterizations of CB-randomness and FB-randomness in terms of prefix-free complexity and martingales?*

Question 8.1.9. *What is the relationship of the bounded variations of Martin-Löf randomness with Schnorr randomness?*

8.2 A characterization of the c.e. reals bounding a CB-random real

The class of array computability c.e. sets was introduced by Downey, Jockusch and Stob [DJS90, DJS96] to explain a number of multiple permitting arguments in computability theory. Recall that a degree $\mathbf{a}$ is array noncomputable if for every function $f \leq_{wtt} \emptyset'$ there is a function $g \leq_T \mathbf{a}$ such that $f(x) < g(x)$ infinitely often. Downey, Greenberg and Weber [DGW07] later introduced the totally ω-c.e. sets to explain the construction needed for a weak critical triple, for which array noncomputability seems too weak.

Definition 8.2.1 ([DGW07])**.** A c.e. degree $\mathbf{a}$ is totally ω-c.e. if every $f \leq_T \mathbf{a}$ is ω-c.e..

Note that array computability can be viewed as a uniform version of this notion where the computable bound (for the mind changes) can be chosen independently of f; hence every c.e. array computable set is totally ω-c.e.. The class of totally ω-c.e. degrees capture a number of natural constructions. Downey, Greenberg and Weber [DGW07] proved that a c.e. degree is not totally ω-c.e. iff it bounds a weak critical triple in the c.e. degrees. Recall that a c.e. set A is a presentation of a c.e. real α, if $\alpha = \mu([A])$. It is easy to see (by padding) that every c.e. real has a computable presentation. On the other hand Downey and LaForte [DL02] showed that there are noncomputable c.e. reals with only computable presentations. Downey and Greenberg [DGb] then showed that a general construction of a c.e. real which controls the Turing degree of all of its presentations, will require the permitting strength of the non totally ω-c.e. degrees. They showed that $\mathbf{a}$ is not totally ω-c.e. iff $\mathbf{a}$ bounds a left c.e. real β and some c.e. set $C \leq_T \beta$ such that every presentation of β is below C.

In Theorem 8.2.2 we show that the non totally ω-c.e. degrees are exactly the class of c.e. degrees which permit the construction of a CB-random real:

Theorem 8.2.2. *Suppose A is a c.e. real. The following are equivalent.*

(i) $deg_T(A)$ is not totally ω-c.e.,

(ii) there is some c.e. real $B \leq_T A$ which is CB-random,

(iii) there is some $B \leq_T A$ which is CB-random.

As an immediate corollary to this theorem, we have that not every superlow set is bounded above by a superlow c.e. set.

We fix a computable enumeration$\{\phi_n\}_{n\in\omega}$ of all partial computable functions. For $\sigma, \tau \in 2^{<\omega}$, we write $\sigma \subset \tau$ if $|\sigma| < |\tau|$ and $\sigma(i) = \tau(i)$ for $i < |\sigma|$; similarly, we write $\sigma \subset x$ if now $x \in 2^\omega$. We let $\{W_n^m\}_{n\in\omega}$ be the m^{th} Martin-Löf test. We use $<_L$ to denote the left-to-right lexicographical ordering on finite strings σ, τ, with 0 being to the left of 1 and $\sigma <_L \tau$ meaning that σ is to the left of τ. This ordering is extended naturally to $x <_L y$ for infinite strings x, y. We assume for any c.e. set U, that if $\sigma \in U_s$ then $|\sigma| < s$.

8.2.1 (i) $\Rightarrow$ (ii)

Assume that $f = \Delta^A$ and that f is not ω-c.e. We will build $B \leq_T A$ and ensure that B is CB-random. We must ensure that $R_{m,i}$ holds for every m, i:

$$R_{m,i}: \quad B \notin \cap_n [W_n^m] \quad \text{if } \phi_i \text{ is total and for all } n,\ \#W_n^m \leq \phi_i(n).$$

To ensure that each requirement R is satisfied, suppose that R is the k^{th} requirement, where $k = \langle m, i\rangle$. Our construction will implement a sequence of modules $(M_j^k)_{j\in\omega}$ for R and each module is given infinitely many opportunities to act. At any particular stage, the construction attempts to satisfy at most one requirement through the implementation of at most one module. Associated with each module M_j^k is an integer $n = n_j^k$, and the module aims to ensure that if $(W_e^m)_{e\in\omega}$ is a CB-test then $B \notin [W_n^m]$ as follows. (Note that as long as some module succeeds, the requirement succeeds.)

Suppose at the current stage s of the construction that it is module M_j^k's turn to act and B is in $[W_n^m]$— that is, $B_{s-1} \in [W_{n,s}^m]$. The module's strategy is to redefine B to the right (outside of $[W_n^m]$), but on precondition that it receives an A-permission, due to certain conditions related to Δ^A.

To be more precise: throughout the construction, the modules $(M_j^k)_{j\in\omega}$ will collectively be defining an approximating function f_k for Δ^A towards ensuring that, for some j, module M_j^k's strategy succeeds (so that R_k is satisfied). We further discuss f_k and the A-permission below.

Module M_j^k is responsible for defining $f_k(j, s)$ for all s; it does so as follows. Whenever $B_{s-1} \in [W_n^m]$ as above, then— supposing this is the ${t_s}^{th}$ time it acts— M_j^k defines $f_k(j, t_s) := \Delta^A(j)[t_s]$. Module M_j^k waits to act at a later stage $q > s$ when either

- B remained in $[W_n^m]$ throughout all intermediate stages $\leq q$ and A changes below the use $\delta(j)$ for $\Delta^A(j)$, or
- B does not remain in $[W_n^m]$ until stage q due to an A-permission being granted to some other module, or perhaps some other requirement.

In either of these two cases, an A-permission is granted and M_j^k moves B to the right.

Now suppose (W_n^m) is a CB-test so that $\#W_n^m \leq \phi_i(n)$. Since B is only ever redefined to the right, it follows that there can be at most $\phi_i(n) = \phi_i(n_j^k)$ A-permissions associated with module M_j^k so that

$$\#\{s : f_k(j, s) \neq f_k(j, s+1)\} \leq \phi_i(n) = \phi_i(n_j^k).$$

It follows that if $B \in [W^m_{n^k_j}]$ for all j, then eventually no A-permission occurs for module M^k_j to act, for all j. Consequently, $f_k(j,t) = \Delta^A(j)[t] = f(j)$ for sufficiently large t and f_k must be an approximating function for $\Delta^A = f$. This means that f is ω-c.e., a contradiction, and thus requirement $R = R_k$ must be satisfied.

We are ready to describe the stage-by-stage construction.

Construction. The construction will proceed in stages of the form $\langle a+1, \langle j,k\rangle\rangle$. The intention is that stage $\langle a+1, \langle j,k\rangle\rangle$ is the a^{th} time in which module M^k_j is allowed to act. Consequently, in what follows, we will use ℓ to denote $\ell = \langle j,k\rangle$. We also define the integer $n^k_j = \langle k,j\rangle + 1$ associated with module M^k_j of the k^{th} requirement. Since Δ^A is total, we assume that $\Delta^A(j)[s]\downarrow$ at every stage $s > j$.

At stage $s = 0$, define $B_0 = 0^\omega$ and go to stage $s+1$.

At stage $s = \langle 0, \ell\rangle > 0$ define $f_k(j,0) = \Delta^A(j)[0]$ and go to stage $s+1$.

At stage $s = \langle a+1, \ell\rangle$, implement the j^{th} module M^k_j of requirement R_k defined as follows.

Module M^k_j.

1. If $\phi_{i,s}(n^k_j)\uparrow$, or $\#W^m_{n^k_j,s} \not\leq \phi_{i,s}(n^k_j)$, or $B_{s-1} \notin [W^m_{n^k_j,s}]$, then no nontrivial action is needed for M^k_j. We simply define $f_k(j,a+1) := f_k(j,a)$, define $B_s := B_{s-1}$ and go to stage $s+1$.

2. Otherwise, define $f_k(j,a+1) = \Delta^A(j)[a+1]$, let $r = \langle a, \ell\rangle$, and implement the following. If $A_{a+1} \upharpoonright \delta(j) \neq A_a \upharpoonright \delta(j)$, then do the following. Let $\sigma \subset B_{s-1}$ be maximal such that $N_\sigma := ([\sigma] \cap \{x : B_{s-1} <_L x\}) \setminus [W^m_{n^k_j,s}]$ is nonempty. Define B_s to be the left-most path of N_σ, and go to stage $s+1$. Otherwise define $B_s := B_{s-1}$ and go to stage $s+1$.

This completes the construction.

Verification. First observe that for any module M^k_j, whenever it changes B, it only adds an amount $q \in \mathbb{Q}$ to B_s where q can be accounted against a distinct part of $W^m_{n^k_j}$. Therefore M^k_j contributes at most $2^{-n^k_j}$ to B. Consequently the total effect of all the modules can contribute at most $\sum_{k,j\in\omega} 2^{-n^k_j} \leq \frac{1}{2}$ to B, which means that σ in the construction, at every stage, can always be found so that N_σ is nonempty.

Lemma 8.2.3. *Every requirement is satisfied.*

Proof. Suppose to the contrary that for some pair m, i, $B \in \cap_n[W_n^m]$, ϕ_i is total, and $\#W_n^m \leq \phi_i(n)$ for all n. We first observe that $\lim_a f_k(j,a) = \Delta^A(j)$ for each j. Let $W = W^m_{n_j^k}$. Since $B \in [W]$, hence at almost every stage of the construction when M_j^k acts, we have case 2 holds; hence we will set $f_k(j,a) = \Delta^A(j)[a]$ at almost every a. Next, we want to show that the f_k-changes is bounded by $O(\phi_i(n_j^k))$. We fix a j, and argue that if $\langle a_0+1, \ell\rangle < \langle a_1+1,\ell\rangle$ are two stages in the construction such that M_j^k acts under case 2, and $f_k(j, a_0+1) \neq f_k(j, a_1+1)$, then $B_s \notin [W_{\langle a_0+1,\ell\rangle}]$ for some $\langle a_0+1,\ell\rangle < s \leq \langle a_1+1,\ell\rangle$. This is because there must be some $a_0 < a \leq a_1$ such that $A_{a+1} \restriction \delta(j) \neq A_a \restriction \delta(j)$. At stage $\langle a+1,\ell\rangle$ of the construction we may assume case 2 holds (otherwise we are done). Hence we will define $B_{\langle a+1,\ell\rangle}$ to avoid $W_{\langle a+1,\ell\rangle} \supseteq W_{\langle a_0+1,\ell\rangle}$. This proves the claim. Now to see that the number of changes in $f_k(-,a)$ is bounded by $O(\phi_i(n_-^k))$, observe that if $f_k(j,a) \neq f_k(j,a+1)$, we must have case 2 applies at stage $\langle a+1,\ell\rangle$ of the construction. □

Lemma 8.2.4. *$B \leq_T A$.*

Proof. Next we describe how to compute $B \leq_T A$. To compute $B(x)$, we would like to say that only modules M_j^k for $n_j^k \leq x$ can change $B(x)$. This is unfortunately not true, because of the "carry-over" in the addition. Instead we have to compute B from A in a slightly more elaborate fashion. Define the total function $g \leq_A$ by the following. Let $g(0) = x$, and given $g(z)$ we define $g(z+1)$ by first searching computably in A for some number a such that $A_a \restriction \delta(g(z))$ is stable and correct. Let $g(z+1) = \max\{\langle a+1, \langle j,k\rangle\rangle \mid n_j^k \leq g(z)\}$. Hence the function g is defined so that after stage $g(z+1)$ of the construction, no module M_j^k for $n_j^k \leq g(z)$ can change B.

Assume we have computed $\sigma = B \restriction x$. Now search for the least z such that either $B_{g(z+2)}(x) = 1$, or else $B_{g(z+2)}(y) = 0$ for some $x < y < g(z+1)$. This search will terminate because otherwise $B = \sigma 011111\cdots$ which means B is computable. Let z be the first found. If $B_{g(z+2)}(x) = 1$ then $B(x) = 1$. Otherwise we claim that $B(x) = 0$. After stage $g(z+2)$, only modules M_j^k for $n_j^k > g(z+1)$ can contribute to B, and the sum of their total contribution to B is $< 2^{-g(z+1)}$. On the other hand if $B_t(x) = 1$ at some $t > g(z+2)$, then the amount added to B after $g(z+2)$ is at least $2^{-x-1} - (2^{-x-2} + \cdots + 2^{-y-1}) = 2^{-y-1} \geq 2^{-g(z+1)}$. □

8.2.2 (iii) $\Rightarrow$ (i)

Suppose $B = \Delta^A$ and B is CB-random. Let ϕ_e be the e^{th} partial computable function. Fix a left c.e. approximation $\{A_s\}$ to A. Define $f(\langle e,k\rangle)$ by the following. Search for the first stage s such that $A_s \upharpoonright \delta(\langle e,k\rangle) = A \upharpoonright \delta(\langle e,k\rangle)$. If $\phi_e(\langle e,k\rangle)[s] \uparrow$ then output $A \upharpoonright \delta(\langle e,k\rangle)$; otherwise output $A \upharpoonright \delta(\phi_e(\langle e,k\rangle) + \langle e,k\rangle)$. Clearly f is total and $f \leq_T A$. Note that the use of the computation is not (and cannot be) computable. We claim f is not ω-c.e.; suppose the contrary we have $f(x) = \lim_s g(x,s)$ where $g(x,-)$ has at most $\phi_e(x)$ mind changes for some total computable functions g and ϕ_e. We build a CB-test $\{V_k\}$ capturing B, contrary to assumption. For each k we find a stage s_0 such that $\phi_e(\langle e,k\rangle)[s_0] \downarrow$, and $\Delta^A \upharpoonright \langle e,k\rangle[s_0] \downarrow$. We then enumerate $\Delta^A \upharpoonright \langle e,k\rangle[s_0]$ into V_k, and for every $s > s_0$ such that $\Delta^A \upharpoonright \langle e,k\rangle + \phi_e(\langle e,k\rangle)[s] \downarrow$ with $g(\langle e,k\rangle, s) \supseteq A \upharpoonright \delta(\langle e,k\rangle + \phi_e(\langle e,k\rangle))[s]$, we enumerate $\Delta^A \upharpoonright \langle e,k\rangle + \phi_e(\langle e,k\rangle)[s]$ into V_k.

Clearly for each k we have $\#V_k \leq 1 + \phi_e(\langle e,k\rangle)$, and that $\mu(V_k)$ is at most $2^{-\langle e,k\rangle} + \phi_e(\langle e,k\rangle)2^{-\langle e,k\rangle - \phi_e(\langle e,k\rangle)} < 2^{-\langle e,k\rangle+1} \leq 2^{-k}$. We claim that $B \in [V_k]$. At stage s_0 we threw in $\Delta^A \upharpoonright \langle e,k\rangle[s_0]$, and if $A \upharpoonright \delta(\langle e,k\rangle)$ is stable at s_0 then clearly $B \in [V_k]$. Since $\{A_s\}$ is a monotonic approximation to A, we therefore may assume that A was not stable at s_0, hence $f(\langle e,k\rangle) = A \upharpoonright \delta(\phi_e(\langle e,k\rangle) + \langle e,k\rangle)$. Since g approximates f correctly, at some large enough stage we will enumerate $B \upharpoonright \langle e,k\rangle + \phi_e(\langle e,k\rangle)$ into V_k.

Question 8.2.5. *Is there a general characterization of the degrees containing a CB-random (in terms of a traceability property)?*

Another direction one can take is to investigate the associated lowness notions. If R is a notion of randomness, we say that A is *low for R-randomness*, if

$$\forall Z(Z \text{ is } R\text{-random} \Rightarrow Z \text{ is } R\text{-random relative to } A).$$

Our concern here is when R is CB-randomness or FB-randomness.

Question 8.2.6. *Is there a noncomputable set which is low for CB-randomness?*

It would appear that every set low for CB-randomness would have to be hyperimmunefree. This is similar to the situation in other lowness classes arising from a randomness notion which involves a computable bound. For instance, every set low for Demuth randomness is hyperimmunefree, the sets low for Kurtz randomness are exactly the sets which are hyperimmunefree and not d.n.c., while low for Schnorr randomness corresponds to the computably traceable sets. On the other hand, lowness classes arising from randomness notions which do not explicitly mention a computable bound are likely to be related to the K-trivial sets; for instance, the low for random sets and the low for weakly 2-random sets. Therefore, we conjecture that:

Conjecture 8.2.7. *The low for FB-randomness sets are exactly the K-trivials.*

It is obvious that every low for FB-randomness is Low(Ω), Ω being Δ^0_2. On the other hand, no FB-random set can be low for FB-randomness, hence not every Low(Ω) set is low for FB-randomness.

Chapter 9

Lowness for Demuth Randomness

The work in this chapter is joint with Rod Downey. It is submitted for publication.

9.1 Introduction

A fundamental theme in the study of computability theory is the idea of *computational feebleness*, which might be loosely defined as properties exhibited by noncomputable sets resembling computability. This is usually described in literature as a notion of lowness, and indicates weakness as an oracle. The classical example of sets exhibiting a property of this sort are the low sets. As mentioned in earlier chapters there is a plethora of results in the literature which suggest that low sets resemble computable sets, particularly for the computably enumerable (c.e.) sets.

In notions of lowness, one usually considers a certain set operation and says that A satisfies the notion of lowness if it does not give any extra power to the operation. In the above example of low sets, the operation concerned was the Turing jump operator. Slaman and Solovay demonstrated in [SS91] a relationship between the low sets, and another seemingly unrelated lowness notion from the theory of inductive inference. Hence their result says that lowness for various notions of computation can be intertwined. In a similar vein, Bickford and Mills [BM] considered the concept of a *superlow* set. The standard construction of a low c.e. set by the preservation of jump computations already made the constructed set superlow (as in the low basis theorem). At first blush we might be tempted to think that the low and superlow sets are very similar, or even the same. However the low and superlow c.e. sets have turned out to be not even elementarily equivalent. Recent examples have suggested that the dynamic properties of low and superlow c.e. sets are very different. For instance Diamondstone [Dia] showed that there was a low cuppable c.e. set A (i.e. $\emptyset' \equiv_T A \oplus L$ for some low c.e. set L), which was not superlow cuppable (i.e. $\emptyset' \equiv_T A \oplus Q$ failed for every superlow c.e. set Q). Another result demonstrating the inequivalence of the two theories is given in Chapter 10, where we show that there is a low c.e. degree which is not the join of any two superlow c.e. degrees.

Recent development in algorithmic randomness have revealed that the theory of low and superlow c.e. sets is much deeper than originally thought. Various lowness notions for Kolmogorov complexity and other operations arising in algorithmic randomness have suggested a deep connection with subclasses of the low sets. Several subclasses of the superlow sets have sprung up, and have been shown to be even better candidates for studying properties resembling computability. A central theme in these classes is the notion of *traceability*, and we have explored and given a survey on this concept in Chapters 2 to 5.

We have already seen in Theorem 2.2.1 that jump traceability and superlowness were the same for n-c.e. sets. When we consider the next level on the Ershov hierarchy, the ω-c.e. sets, these two notions separate. If we consider non-Δ^0_2 sets, the situation becomes even more bizarre. There is a perfect Π^0_1 class of sets which are jump traceable, via an exponential bound. Such a phenomenon highlights an important inherent property of being traceable; we are only able to enumerate possible values of $A\upharpoonright n$, but beyond that we are given no additional information to suggest which one of the enumerated values is correct. Indeed Kjos-Hanssen and Nies [KHN] recently showed that jump traceable sets could even be superhigh.

Traceability plays a very important role in understanding lowness notions arising in algorithmic information theory. If R is a notion of effective randomness, then *low for* R would denote all the sets A for which $R^A = R$ (i.e. every random Z is still random relative to A). The work of Terwijn and Zambella [TZ01], Kjos-Hanssen, Nies and Stephan [KHNS05], and Bedregal and Nies [BN03] showed that the low for Schnorr randomness sets were exactly the computably traceable sets, revealing an interesting interaction between "predictability" in terms of traceability, and simplicity in terms of Kolmogorov complexity.

Several other lowness notions (apart from low for Martin-Löf randomness) have been studied in the literature. We list a few notable examples. Downey, Greenberg, Mikhailovich and Nies [DGMN08] showed that the computably traceable sets were exactly those which are low for computable measure machines. Here, a computable measure machine is a prefix-free machine with a computable halting probability, and A is low for computable measure machines (c.m.m.) if for each c.m.m. M relative to A, there is a c.m.m. N and a constant c such that $K_M^A(\sigma) \geq K_N(\sigma) - c$ for every σ.

Nies [Nie05b] showed that the only sets which are low for computable randomness[1], are the computable sets. The combined work of Greenberg, Miller, Stephan and Yu [GM09, SY06] revealed that the sets which are low for Kurtz randomness, are exactly the hyperimmunefree and non-d.n.c. degrees. These are also the sets which were low for weak 1-genericity, which demonstrate yet another interaction between lowness notions in classical computability, and randomness. For more examples we refer the reader to Chapter 8 of Nies' book [Nie09].

In the next section, we contribute with another result in this direction. We consider lowness with respect to a less well-known notion of randomness, known as Demuth randomness. This was introduced by Demuth [Dem88] and was originally motivated by topics in constructive analysis. This appears to be a very natural (strong) randomness notion to study, and not much work has yet been done on this class.

Definition 9.1.1. A Demuth test is a sequence of c.e. open sets $\{W_{g(x)}\}_{x\in\mathbb{N}}$ such that $\mu W_{g(x)} < 2^{-x}$ for every x, and every ω-c.e. function g. We say that Z passes the test if $\forall^\infty x Z \notin W_{g(x)}$. A real is Demuth random if it passes every Demuth test.

A Demuth test is a sequence of c.e. open sets, but the function giving the weak indices is an ω-c.e. function. Informally if we were building such a test (to try and catch some real number) we have additional power over building a ML-test because we can change the name of $W_{g(x)}$ a bounded number of times. That is, we can remove a certain part (or even all) of what we have enumerated into $W_{g(x)}$ so far,

[1]A real is computably random if it succeeds on every computable martingale.

as long as we only do it a computably bounded number of times. The definition of passing a Demuth test is as in the Solovay sense, and we cannot always require that $W_{g(x)} \supseteq W_{g(x+1)}$. There is no universal Demuth test, although there is a single special test $\{W_{\hat{g}(x)}\}$ which is universal in the sense that every real passing the special test is Demuth random. However the function $\hat{g}(x)$ has to emulate every ω-c.e. function, and so $\hat{g}(x)$ is Δ^0_2. Hence the special test is not a Demuth test.

Clearly the Demuth randoms lie between 2-randomness and ML-randomness. It is not hard to construct a Δ^0_2 Demuth random using the special test, and obviously no ω-c.e. set can be Demuth random. Hence the containments are proper. Demuth randoms exhibit properties which can be found in both 1- and 2-random reals. For instance every Demuth random (like every 2-random) is weakly jump traceable (hence GL_1). This means they are of hyperimmune degree by a result of Miller and Nies (Theorem 8.1.19 of [Nie09]). Since there are hyperimmunefree weakly 2-randoms, this implies that Demuth randomness and weak 2-randomness are incomparable notions. However unlike the 2-randoms, the Demuth test notion is essentially computably enumerable.

We contribute two theorems to the understanding of this notion of randomness. First, we prove that every Demuth random is array computable. This notion was introduced by Downey, Jockusch and Stob [DJS90] to describe the class of reals below which certain multiple permitting arguments could not be carried out. This again suggests that Demuth randoms are like the 2-randoms, having low computational strength. Recall that every 2-random real is low for Ω and hence array computable; however not every Demuth random is low for Ω (in particular no Δ^0_2 Demuth random can be low for Ω). The Δ^0_2 Demuth random sets form an interesting class, being both low *and* array computable but not superlow.

Theorem 9.1.2. *Each Demuth random Z is array computable.*

Proof. We observe the proof that every Demuth random is GL_1 already does it; this can be found in Chapter 3 of Nies [Nie09]. The proof actually produces an ω-c.e. function g which dominates the function $\Theta^Z(x) :=$ least s such that $\Phi^Z_x(x)[s] \downarrow$ (which of course implies that $Z' \leq_T Z \oplus \emptyset'$). By usual convention the output value of $\Phi^Z_x(x)$ is $< \Theta^Z(x)$. Since every function computable in Z can be coded into the diagonal, we have a computable function p such that for every e, and almost every $y > e$, we have $\Phi_e^Z(y) = \Phi_{p(e,y)}^Z(p(e,y)) < \Theta^Z(p(e,y)) < g(p(e,y)) < \tilde{g}(y)$, where $\tilde{g}(y) := \max\{g(p(0,y)), g(p(1,y)), \cdots, g(p(y,y))\}$ is ω-c.e. as well. $\square$

Next, we study the notion of lowness with respect to Demuth randomness. A relativized Demuth test involves full relativization. That is, a Demuth test relative to A is a sequence $\{W_{g(x)}^A\}$ where $\mu W_{g(x)}^A < 2^{-x}$ for every x. Here, $g(x) = \lim \tilde{g}(x,s)$ for some A-computable function $\tilde{g}$, and the number of $\tilde{g}$-mind changes is bounded by an A-computable function. A real Z is Demuth random relative to A if it passes every Demuth test relative to A. We say that A is *low for Demuth randomness* if every Demuth random is Demuth random relative to A. In the next section we prove that every real low for Demuth randomness is of hyperimmunefree degree.

However it is unknown if there is any set which is noncomputable and low for Demuth randomness. A construction of such a set will have to build a hyperimmune-free degree, and if one uses the standard forcing method then one has to address the issue of constructing the effective objects required in the proof. We conjecture that:

Conjecture 9.1.3. *Every set low for Demuth randomness is computable.*

9.2 Each set low for Demuth randomness is hyperimmunefree

We work in the Cantor space 2^ω with the usual clopen topology. The basic open sets are of the form $[\sigma]$ where σ is a finite string, and $[\sigma] = \{X \in 2^\omega \mid X \supset \sigma\}$. We fix some effective coding of the set of finite strings, and identify finite strings with their code numbers. We treat W_x as a c.e. open set, consisting of basic clopen sets. We say that $[\sigma] \in W_x$ to mean that the code number of σ is in W_x, and we say that a string $\tau \in W_x$ if $\tau \supseteq \sigma$ for some $[\sigma] \in W_x$. Equivalently we say that τ is *captured by* W_x. The same definition holds if we replace τ by an infinite binary string. We prove:

Theorem 9.2.1. *No set of hyperimmune degree can be low for Demuth randomness.*

Proof. Suppose A is of hyperimmune degree. Let h^A be a function total computable in A and nondecreasing, which escapes domination by all total computable functions.

That is, for all total computable g, $\exists^\infty x(g(x) < h^A(x))$. We build a $Z \leq_T A'$ which is Demuth random, but not Demuth random relative to A. To do this, we give an A-computable approximation $\{Z_s\}$ to Z. The construction will try to achieve two goals. The first is to make Z Demuth random by making Z avoid all Demuth tests. The second goal is to ensure that for infinitely many x, there are at most $h^A(x)$ many mind changes of $Z_s \upharpoonright x$. Hence we can easily use the approximation Z_s to build a Demuth test relative to A capturing Z infinitely often. Hence Z cannot be Demuth random relative to A.

9.2.1 The motivation

Before we describe the strategy used to prove Theorem 9.2.1, let us see why an attempted construction of a c.e. set A which is low for Demuth randomness fails. Let us consider a single (relativized) Demuth test $\{V_x^A\}$, played by the opponent, where the index for V_x^A can change $h^A(x)$ times. Now we have to cover $\{V_x^A\}$ with a plain Demuth test $\{U_x\}$, by making sure that $V_x^A \subseteq U_x$ for every x. If $h^A(x) = 0$ for all x, then we could just follow the construction of a c.e. set which is low for random. We would enumerate y into A (to make A noncomputable), if the penalty we have to pay for making the enumeration of y is small. Even when h^A is computable, we can always arrange the enumerations so that $V_x^A \subseteq U_x$ eventually, because we could use $h^A(x)$ as the bound for the index change of U_x.

The problem is that an enumeration into A not only increases the amount we have to put into U_x, but also gives the opponent a chance to redefine $h^A(x)$. Suppose he has defined $h^A(x)$ with use b_x. At some stage we will have to commit ourselves to a number $g(x)$, and promise never to change the index for U_x more than $g(x)$ times. We would of course declare that $g(x) > h^A(x)$, but once we do that, the opponent could challenge us to change $A \upharpoonright b_x$ to ensure the noncomputability of A. We have to eventually change $A \upharpoonright b_x$ at some x, and allow the opponent to make $h^A(x) > g(x)$, and then we are stuck.

Note that the opponent will be likely to have a winning strategy, if h^A escapes domination by all computable functions. He could then carry out the above for each e, patiently waiting for an x such that $h^A(x) > \varphi_e(x)$, and then defeat the e^{th} Demuth test. This is the basic idea used in the following proof, where we will play the opponent's winning strategy.

9.2.2 Listing all Demuth tests

In order to achieve the first goal, we need to specify an effective listing of all Demuth tests. It is enough to consider all Demuth tests $\{U_x\}$ where $\mu(U_x) < 2^{-3(x+1)}$. Let $\{g_e\}_{e\in\mathbb{N}}$ be an effective listing of all partial computable functions of a single variable. For every g in the list, we will assume that in order to output $g(x)$, we will have to first run the procedures to compute $g(0), \cdots, g(x-1)$, and wait for all of them to return, before attempting to compute $g(x)$. We may also assume that g is nondecreasing. This minor but important restriction on g ensures that:

(i) $dom(g)$ is either $\mathbb{N}$, or an initial segment of $\mathbb{N}$,

(ii) for every x, $g(x+1)$ converges strictly after $g(x)$, if ever.

By doing this, we will not miss any total nondecreasing computable function. It is easy to see that there is a total function $k \leq_T \emptyset'$ that is universal in the following sense:

1. if $f(x)$ is ω-c.e. then for some e, $f(x) = k(e,x)$ for all x,
2. for all e, the function $\lambda x k(e,x)$ is ω-c.e.,
3. there is a uniform approximation for k such that for all e and x, the number of mind changes for $k(e,x)$ is bounded by

$$\begin{cases} g_e(x) & \text{if } g_e(x)\downarrow, \\ 0 & \text{otherwise.} \end{cases}$$

Let $k(e,x)[s]$ denote the approximation for $k(e,x)$ at stage s. Denote $U_x^e = W_{k(e,x)}$, where we stop enumeration if $\mu(W_{k(e,x)}[s])$ threatens to exceed $2^{-3(x+1)}$. Then for each e, $\{U_x^e\}$ is a Demuth test, and every Demuth test is one of these. To make things clear, we remark that there are two possible ways in which $U_x^e[s] \neq U_x^e[s+1]$. The first is when $k(e,x)[s] = k(e,x)[s+1]$ but a new element is enumerated into $W_{k(e,x)}$. The second is when $k(e,x)[s] \neq k(e,x)[s+1]$ altogether; if this case applies we say that U_x^e has a *change of index at stage* $s+1$.

9.2.3 The strategy

Now that we have listed all Demuth tests, how are we going to make use of the function h^A? Note that there is no single universal Demuth test; this complicates matters slightly. The e^{th} requirement will ensure that Z passes the first e many (plain) Demuth tests. That is,

$$\mathcal{R}_e : \text{ for each } k \leq e,\ Z \text{ is captured by } U^k_x \text{ for only finitely many } x.$$

$\mathcal{R}_e$ will do the following. It starts by picking a number r_e, and decides on $Z\restriction r_e$. This string can only be captured by U^k_x for $x \leq r_e$, so there are only finitely many pairs $\langle k, x\rangle$ to be considered since we only care about $k \leq e$. Let S_e denote the collection of these open sets. If any $U^k_x \in S_e$ captures $Z\restriction r_e$, we would change our mind on $Z\restriction r_e$. If at any point in time, $Z\restriction r_e$ has to change more than $h^A(0)$ times, we would pick a new follower for r_e, and repeat, comparing with $h^A(1), h^A(2), \cdots$ each time. The fact that we will eventually settle on a final follower for r_e, will follow from the hyperimmunity of A; all that remains is to argue that we can define an appropriate computable function *at each* $\mathcal{R}_e$, in order to challenge the hyperimmunity of A.

Suppose that $r^0_e, r^1_e, \cdots$ are the followers picked by $\mathcal{R}_e$. The required computable function P would be something like $P(n) = \sum_{k\leq e}\sum_{x\leq r^n_e} g_k(x)$, for if $P(N) < h^A(N)$ for some N, then we would be able to change $Z\restriction r^N_e$ enough times on the N^{th} attempt. There are two considerations. Firstly, we do not know which of $g_0, \cdots, g_e$ are total, so we cannot afford to wait on non-converging computations when computing P. However, as we have said before, we can have a different P at each requirement, and the choice of P can be nonuniform. Thus, P could just sum over all the total functions amongst $g_0, \cdots, g_e$.

The second consideration is that we might not be able to compute $r^0_e, r^1_e, \cdots$, if we have to recover r^n_e from the construction (which is performed with oracle A). We have to somehow figure out what r^n_e is, externally to the construction. Observe that however, if we restrict ourselves to nondecreasing $g_0, g_1, \cdots$, it would be sufficient to compute an upper bound for r^n_e. We have to synchronize this with the construction: instead of picking r^n_e when we run out of room to change $Z\restriction r^{n-1}_e$, we could instead pick r^n_e the moment enough of the $g_k(x)$ converge and demonstrate that their sum exceeds $h^A(r^{n-1}_e)$. To recover a bound for say, r^1_e externally, we compute the first stage t such that *all of the* $g_k(x)[t]$ have converged for $x \leq r_e^0$ and g_k total.

9.2.4 Notations used for the formal construction

The construction uses oracle A. At stage s we give an approximation $\{Z_s\}$ of Z, and at the end we argue that $Z \leq_T A'$. The construction involves finite injury of the requirements. $\mathcal{R}_1$ for instance, would be injured by $\mathcal{R}_0$ finitely often while $\mathcal{R}_0$ is waiting for hyperimmune permission from h^A. We intend to satisfy $\mathcal{R}_e$, by making $\mu(U_x^e \cap [Z \restriction r])$ small for appropriate x, r. At stage s, we let $r_e[s]$ denote the follower used by $\mathcal{R}_e$. At stage s of the construction we define Z_s up till length s. We do this by specifying the strings $Z_s \restriction r_0[s], \cdots, Z_s \restriction r_k[s]$ for an appropriate number k (such that $r_k[s] = s - 1$). We adopt the convention of $r_{-1} = -1$ and $\alpha \restriction -1 = \alpha \restriction 0 = \langle\rangle$ for any string α. We let $S_e[s]$ denote all the pairs $\langle k, x\rangle$ for which $\mathcal{R}_e$ wants to make Z avoid U_x^k at stage s. The set $S_e[s]$ is specified by

$$S_e[s] = \{\langle k, x\rangle \mid k \leq e \ \wedge \ r_{k-1}[s] + 1 \leq x \leq r_e[s]\}.$$

Define the sequence of numbers

$$M_n = \sum_{j=n}^{2n} 2^{-(1+j)};$$

these will be used to approximate Z_s. Roughly speaking, the intuition is that $Z_s(n)$ will be chosen to be either 0 or 1 depending on which of $(Z_s \restriction n)^\frown 0$ or $(Z_s \restriction n)^\frown 1$ has a measure of $\leq M_n$ when restricted to a certain collection of U_x^e.

If P is an expression we append $[s]$ to P, to refer to the value of the expression as evaluated at stage s. When the context is clear we drop the stage number from the notation.

9.2.5 Formal construction of Z

At stage $s = 0$, we set $r_0 = 0$ and $r_e \uparrow$ for all $e > 0$, and do nothing else. Suppose $s > 0$. We define $Z_s \restriction r_k[s]$ inductively; assume that has been defined for some k. There are two cases to consider for $\mathcal{R}_{k+1}$:

1. $r_{k+1}[s] \uparrow$: set $r_{k+1} = r_k[s] + 1$, end the definition of Z_s and go to the next stage.

2. $r_{k+1}[s]\downarrow$: check if $\sum_{\langle e,x\rangle\in S_{k+1}[s]} 2^{r_{k+1}} g_e(x)[s] \le h^A(r_{k+1}[s])$. The sum is computed using converged values, and if $g_e(x)[s]\uparrow$ for any e, x we count it as 0. There are two possibilities:

   (a) *sum* $> h^A(r_{k+1})$: set $r_{k+1} = s$, and set $r_{k'}\uparrow$ for all $k' > k+1$. End the definition of Z_s and go to the next stage.

   (b) *sum* $\le h^A(r_{k+1})$: pick the leftmost node $\sigma \supseteq Z_s\restriction r_k[s]$ of length $|\sigma| = r_{k+1}[s]$, such that $\sum_{\langle e,x\rangle\in S_{k+1}[s]} \mu(U_x^e[s]\cap[\sigma]) \le M_{r_{k+1}[s]}$. We will later verify that σ exists by a counting of measure. Let $Z_s\restriction r_{k+1}[s] = \sigma$.

We say that $\mathcal{R}_{k+1}$ has *acted.* If 2(a) is taken, then we say that $\mathcal{R}_{k+1}$ has *failed the sum check.* This completes the description of Z_s.

9.2.6 Verification:

Clearly, the values of the markers $r_0, r_1, \cdots$ are kept in increasing order. That is, at all stages s, if $r_k[s]\downarrow$, then $r_0[s] < r_1[s] < \cdots < r_k[s]$ are all defined. From now on when we talk about Z_s, we are referring to the fully constructed string at the end of stage s. It is also clear that the construction keeps $|Z_s| < s$ at each stage s.

Lemma 9.2.2. *Whenever step 2(b) is taken, we can always define $Z_s\restriction r_{k+1}[s]$ for the relevant k and s.*

Proof. We drop s from notations, and proceed by induction on k. Let Υ be the collection of all possible candidates for $Z_s\restriction r_{k+1}$, that is, $\Upsilon = \{\sigma : \sigma \supseteq Z\restriction r_k \wedge |\sigma| = r_{k+1}\}$. Suppose that $k \ge 0$:

$$\begin{aligned}
&\sum_{\sigma\in\Upsilon}\sum_{\langle e,x\rangle\in S_{k+1}} \mu(U_x^e\cap[\sigma]) \;=\; \sum_{\langle e,x\rangle\in S_{k+1}}\sum_{\sigma\in\Upsilon}\mu(U_x^e\cap[\sigma])\\
\le\;& \sum_{\langle e,x\rangle\in S_{k+1}} \mu(U_x^e\cap[Z\restriction r_k]) \;\le\; \sum_{\langle e,x\rangle\in S_k}\mu(U_x^e\cap[Z\restriction r_k]) + \sum_{x=r_k+1}^{r_{k+1}}\sum_{e\le k+1}\mu(U_x^e)\\
\le\;& M_{r_k} + \sum_{x=r_k+1}^{r_{k+1}} 2^{-2x} \text{ (since } k\le r_k) \;\le\; M_{r_k} + \sum_{x=2r_k+1}^{r_k+r_{k+1}} 2^{-(1+x)}\\
=\;& \sum_{x=r_{k+1}}^{2r_{k+1}} 2^{-(1+x)}2^{r_{k+1}-r_k} \text{ (adjusting the index } x) \;=\; M_{r_{k+1}}|\Upsilon|.
\end{aligned}$$

Hence, there must be some σ in Υ which passes the measure check in 2(b) for $Z\upharpoonright r_{k+1}$. A similar, but simpler counting argument follows for the base case $k = -1$, using the fact that the search now takes place above $Z\upharpoonright r_k = \langle\rangle$. □

Lemma 9.2.3. *For each e, the follower $r_e[s]$ eventually settles.*

Proof. We proceed by induction on e. Note that once $r_{e'}$ has settled for every $e' < e$, then $\mathcal{R}_e$ will get to act at every stage after that. Hence there is a stage s_0 such that

(i) $r_{e'}$ has settled for all $e' < e$, and

(ii) r_e receives a new value at stage s_0.

Note also that $\mathcal{R}_e$ will get a chance to act at every stage $t > s_0$, and the only reason why r_e receives a new value after stage s_0, is that $\mathcal{R}_e$ fails the sum check. Suppose for a contradiction, that $\mathcal{R}_e$ fails the sum check infinitely often after s_0.

Let $q(n-1)$ be the stage where $\mathcal{R}_e$ fails the sum check for the n^{th} time after s_0. In other words, $q(0), q(1), \cdots$ are precisely the different values assigned to r_e after s_0. Let $\mathcal{C}$ be the collection of all $k \leq e$ such that g_k is total, and d be a stage where $g_k(x)[d]$ has converged for all $k \leq e$, $k \notin \mathcal{C}$ and $x \in dom(g_k)$. We now define an appropriate computable function to contradict the hyperimmunity of A. Define the total computable function p by: $p(0) = 1 + \max\{s_0, d,$ the least stage t where $g_k(r_e[s_0])[t]\downarrow$ for all $k \in \mathcal{C}\}$. Inductively define $p(n+1) = 1+$ the least t where $g_k(p(n))[t]\downarrow$ for all $k \in \mathcal{C}$. Let $P(n) = \sum_{k\leq e}\sum_{x\leq p(n)} 2^{p(n)} g_k(x)[p(n+1)]$, which is the required computable function.

One can show by a simple induction, that $p(n) \geq q(n)$ for every n, using the fact that $\mathcal{R}_e$ is given a chance to act at every stage after s_0, as well as the restrictions we had placed on the functions $\{g_k\}$. Let N be such that $P(N) \leq h^A(N)$. At stage $q(N+1)$ we have $\mathcal{R}_e$ failing the sum check, so that $h^A(N) < h^A(q(N)) < \sum_{\langle k,x\rangle\in S_e} 2^{q(N)} g_k(x)$, where everything in the last sum is evaluated at stage $q(N+1)$. That last sum is clearly $< P(N) \leq h^A(N)$, giving a contradiction. □

Let $\hat{r}_e$ denote the final value of the follower r_e. Let $Z = \lim_s Z_s$. We now show that Z is not Demuth random relative to A. For each e and s, $Z_{s+1+\hat{r}_e}\upharpoonright\hat{r}_e$ is defined, by Lemma 9.2.2.

Lemma 9.2.4. *For each e, $\#\{t \geq 1+\hat{r}_e : Z_t{\upharpoonright}\hat{r}_e \neq Z_{t+1}{\upharpoonright}\hat{r}_e\} \leq h^A(\hat{r}_e)$.*

Proof. Suppose that $Z_{t_1} \upharpoonright \hat{r}_e \neq Z_{t_2} \upharpoonright \hat{r}_e$ for some $1+\hat{r}_e \leq t_1 < t_2$. We must have $r_{e'}$ already settled at stage t_1, for all $e' \leq e$. Suppose that $Z_{t_2} \upharpoonright \hat{r}_e$ is to the left of $Z_{t_1} \upharpoonright \hat{r}_e$, then let e' be the least such that $Z_{t_2} \upharpoonright \hat{r}_{e'}$ is to the left of $Z_{t_1} \upharpoonright \hat{r}_{e'}$. The fact that $\mathcal{R}_{e'}$ didn't pick $Z_{t_2} \upharpoonright \hat{r}_{e'}$ at stage t_1, shows that we must have a change of index for U^a_b between t_1 and t_2, for some $\langle a,b\rangle \in S_{e'} \subseteq S_e$. Hence, the total number of mind changes is at most $2^{\hat{r}_e} \sum_{\langle a,b\rangle \in S_e} g_a(b)$, where divergent values count as 0. $2^{\hat{r}_e}$ represents the number of times we can change our mind from left to right consecutively without moving back to the left, while $\sum_{\langle a,b\rangle \in S_e} g_a(b)$ represents the number of times we can move from right to left. Since $\mathcal{R}_e$ never fails a sum check after $\hat{r}_e$ is picked, it follows that the number of mind changes has to be bounded by $h^A(\hat{r}_e)$. □

By asking A' appropriate 1-quantifier questions, we can recover $Z = \lim_s Z_s$. Hence Z is well-defined and computable from A'. To see that Z is not Demuth random in A, define the A-Demuth test $\{V^A_x\}$ by the following: run the construction and enumerate $[Z_s{\upharpoonright}x]$ into V^A_x when it is first defined. Subsequently each time we get a new $Z_t{\upharpoonright}x$, we change the index for V^A_x, and enumerate the new $[Z_t{\upharpoonright}x]$ in. If we ever need to change the index $> h^A(x)$ times, we stop and do nothing. By Lemma 9.2.4, Z will be captured by $V^A_{\hat{r}_e}$ for every e.

Lastly, we need to see that Z passes all $\{U^e_x\}$. Suppose for a contradiction, that $Z \in U^e_x$ for some e and $x > \hat{r}_e$. Let δ be such that $Z \in [\delta] \in U^e_x$, and let $e' \geq e$ such that $\hat{r}_{e'} > |\delta|$. Go to a stage in the construction where δ appears in U^e_x and never leaves, and $r_{e'} = \hat{r}_{e'}$ has settled. At every stage t after that, observe that $\langle e,x\rangle \in S_{e'}$, and that $\mathcal{R}_{e'}$ will get to act, at which point it will discover that $\mu(U^e_x \cap [Z{\upharpoonright}\hat{r}_{e'}]) = 2^{-\hat{r}_{e'}} > M_{\hat{r}_{e'}}$. Thus, $\mathcal{R}_{e'}$ never picks $Z{\upharpoonright}\hat{r}_{e'}$ as an initial segment for Z_t, giving us a contradiction. □

Part III

Turing Degree Theory

Chapter 10

Splitting into degrees with low computational strength

The work in this chapter is joint with Rod Downey.

10.1 An ultrahigh c.e. degree which cannot be split into two superlow c.e. degrees

The Sacks splitting theorem says that every c.e. set can be split (in a set-splitting) into disjoint low c.e. sets. An examination of the proof will reveal that it is impossible to place a computable bound on the number of injuries to each lowness preservation strategy. Indeed if $A = A_0 \sqcup A_1$ then $A \equiv_{wtt} A_0 \oplus A_1$, and by a result of Bickford and Mills [BM], we cannot even have a set-splitting of any wtt-complete c.e. set into two superlow c.e. halves.

The general question of whether there was a degree split (into superlow degrees) was open. That is, given any c.e. degree $\mathbf{a}$, are there superlow c.e. degrees $\mathbf{a_0}, \mathbf{a_1}$ such that $\mathbf{a} = \mathbf{a_0} \cup \mathbf{a_1}$? This was true for the case $\mathbf{a} = \mathbf{0}'$. We answer the question in the negative by constructing an incomplete c.e. degree which cannot be split in such a way; in fact we can make the example very close to $\mathbf{0}'$ by making it ultrahigh. This shows that Turing completeness was, in a way, necessary to ensure the disengagement of coding markers required to create such a split. It is easy to see that our counterexample can also be made low.

We prove that:

Theorem 10.1.1. *There is an ultrahigh c.e. degree which cannot be split into two superlow c.e. degrees.*

10.1.1 Requirements

We build a c.e. set A satisfying the following requirements

$$\begin{aligned} \mathcal{N}_e &: \text{ If } A = \Phi_e^{W_e \oplus V_e} \text{ and } W_e \oplus V_e = \Delta_e^A, \text{ then one of } W_e \text{ or } V_e \\ &\quad \text{ is not superlow.} \\ \mathcal{P}_e &: \text{ If } \Phi_e^A \text{ is an order, make } \emptyset' \text{ } A\text{-jump traceable via } \Phi_e^A, \end{aligned}$$

Here, we let $\langle \Phi_e, \Delta_e \rangle$ denote the e^{th} pair of Turing reductions, and $J^{\emptyset'}(k)$ denote the value of the universal $\emptyset'$-partial computable function $\{k\}^{\emptyset'}(k)$. $\langle W_e, V_e \rangle$ is the e^{th} pair of c.e. sets. Both kinds of requirements above will be further divided into subrequirements; we will describe how to do this in the following section.

10.1.2 Making A not the join of superlow sets

Since the superlow c.e. sets are exactly the jump traceable c.e. sets, we will diagonalize against all the possible traces for the jump $J^{\emptyset'}$. The requirement $\mathcal{N}_e$ is divided into subrequirements $\mathcal{N}_{e,i,j}$, and $\mathcal{N}_e$ is satisfied in a nonuniform manner depending on the outcomes of the individual subrequirements (i.e. we ensure that one of W_e or V_e is not superlow, but we do not know which). Basically each $\mathcal{N}_{e,i,j}$ aims to make *either* J^{W_e} not traced by T_i, *or* it makes J^{V_e} not traced by T_j. Here, we let $T_0, T_1, \cdots$ be an effective list of all possible c.e. traces. That is, T_e is a uniform sequence of c.e. sets $T_e(0), T_e(1), \cdots$. Also associated with each sequence T_e is a partial computable function t_e of a single variable. We say that W is jump traceable via T_e, if for all x, $|T_e(x)| < t_e(x)$ and whenever $J^W(x) \downarrow$ implies that $J^W(x) \in T_e(x)$. Every superlow c.e. set will be jump traceable via some T_e.

If every $\mathcal{N}_{e,i,j}$ is satisfied, then clearly $\mathcal{N}_e$ itself will be satisfied: if $\forall i \exists j$ such that $\mathcal{N}_{e,i,j}$ succeeds in the first alternative, then W_e is not superlow. Otherwise V_e will not be superlow. We first describe the $\mathcal{N}_{e,i,j}$-strategy in isolation, then we will describe the ultrahigh strategy, and finally we will see how to put both strategies together. We see the $\mathcal{N}_{e,i,j}$-strategy as having primarily a negative role blocking elements from entering A. However, like many other strategies which force every set in a constructed Turing degree avoid a certain property, there is a certain amount of finite positive action. Examples include constructing a Turing degree free of semi- or hemi-maximal sets, and a d.c.e. degree which is not of c.e. degree.

Consider an arbitrary $\mathcal{N}$-requirement (we drop all indices, since we are considering it in isolation). Suppose $A \equiv_T W \oplus V$ via the reductions Φ, Δ. The basic plan to make either J^W not traced by T or J^V not traced by T is the following. By the Recursion Theorem, we control parts of the jump. To wit, we are supplied with an infinite list of indices x for which we are able to build the x^{th} Turing functional $J^X(x)$. We pick a number of indices $\eta_0, \eta_1, \cdots, \eta_m$ and we will use $J^X(\eta_i)$ to diagonalize against the trace $T(\eta_i)$. If we were merely trying to make A itself not superlow (this is weaker than $\mathcal{N}$), we could do the following. We wait for $t(\eta_0)\downarrow$ and then start the following cycle. Set $J^A(\eta_0)[s] \downarrow = s$ and wait for s to be enumerated in $T(\eta_0)$. When s enters $T(\eta_0)$ (at $t > s$) we change A to make the previous axioms invalid, and then set $J^A(\eta_0)[t] \downarrow = t$ and repeat. This cycle repeats at most $t(\eta_0)$ times, so we only need to change A (for the sake of a single requirement) finitely often.

However we need to make A not of the same Turing degree as the join of superlow sets. We need to simultaneously run two versions of the above strategy (one for W and one for V). Each change in A will make progress towards *one of* the two strategies. Hence we will just need to repeat the above cycle more times; in this case we need to freeze A in between cycles, so that any progress towards either side does not become undone. Specifically, the modified strategy is:

(1) pick a number of followers (targeted for entry into A) $x_1, \cdots, x_N$ such that $x_{i+1} > \delta(\varphi(x_i))$ for all i. These followers will be enumerated in decreasing order of magnitude. Freeze A below all these uses.

(2) define $J^W(\eta_0)\downarrow$ and $J^V(\eta_1)\downarrow$ both with use $\varphi(x_N)$. Wait for these values to enter $T(\eta_0)$ and $T(\eta_1)$ respectively.

(3) enumerate x_N into A. Then one of W or V has to change below $\varphi(x_N)$. If we get a W-change, repeat steps (2)-(3) with the next index $J^V(\eta_2)$ and next

follower x_{N-1}. If we get a V-change, repeat steps (2)-(3) with the same indices $J^W(\eta_0)$ and $J^V(\eta_1)$, and next follower x_{N-1}.

If we have many consecutive V-changes without a W-change, then we would have made $J^V(\eta_i) \not\in T(\eta_i)$ for some i. If on the other hand we are interrupted with a W-change at each index $J^V(\eta_1), \cdots J^V(\eta_m)$, then we have made $J^W(\eta_0) \not\in T(\eta_0)$. It is important that once we start on the diagonalization, we freeze A below the uses, and also we enumerate the followers $x_N, \cdots, x_1$ in decreasing order of magnitude; this is to ensure that we keep the W-use of $J^W(\eta_0)$ above $\varphi(x)$ for any follower x not yet in A.

We draw the reader's attention to the following fact. Step 1 consists of many individual actions. Namely we first pick x_0 and wait for $\delta(\varphi(x_0)) \downarrow$, before selecting x_1, and so on. We could have increased the A-restraint to $\delta(\varphi(x_0))$ the moment we see it converge, so that we never need to re-select x_1. However, for technical reasons which will be explained later, we will not do this. Observe that we only need to increase A-restraint once all of the x_i have been picked. For instance if A changes below $\delta(\varphi(x_0))$ while we were waiting for $\delta(\varphi(x_1))$ to converge, then we would pick a new value for x_1 above the new use $\delta(\varphi(x_0))$. If infinitely often we do this, then $\delta(\varphi(x_0))$ would tend to ∞, and we would not need to act for $\mathcal{N}$ after all.

It is easy to see that we can combine all of the different $\mathcal{N}$-requirements in a finite injury argument. These requirements guarantee that A is of intermediate Turing degree ($\mathbf{0} < deg_T(A) < \mathbf{0}'$). It is easy to see that we can throw in lowness requirements to make A low, but A clearly cannot be superlow (because we do not know a priori the number of injuries). This strategy also admits variations; for instance it is not hard to see that one can make A not of the same Turing degree as the join of two sets W, V where W is low and V is superlow. This will involve considering all the possible lowness indices, and a straightforward modification of the above strategy together with a standard infinite injury argument will settle the issues.

10.1.3 Making A ultrahigh

The construction takes place on a tree, which grows downwards. We order the nodes lexicographically. We use $\supset$ for string extension. We let $\alpha <_L \beta$ denote that α is strictly to the left of β. If $\mathcal{R}$ is a requirement, we say that α is an $\mathcal{R}$-node, if α is assigned the requirement $\mathcal{R}$. A negative node is a $\mathcal{N}_{e,i,j}$-node for some e, i, j, while an ultrahigh node is a $\mathcal{P}_{e,k}$-node for some e, k. A top node or a mother node is a $\mathcal{P}_e$-node for some e. We call the $\mathcal{N}$-nodes negative, even though their actions are not solely to prevent numbers from entering A; they do themselves enumerate numbers into A (though only finitely often before being injured).

The basic strategy to make A ultrahigh has already been described in Sections 4.3 and 5.2.3. Here we only outline the important points which are peculiar to this construction. If Φ_e^A is an order, the requirement $\mathcal{P}_e$ will build an A-u.c.e. sequence $\{V_k^A\}_{k \in \mathbb{N}}$ such that for all k, $|V_k^A| \leq \Phi_e^A(k)$ and $J^{\emptyset'}(k) \in V_k^A$. To do this, $\mathcal{P}_e^A$ will divide the task into infinitely many substrategies, or modules. The k^{th} module will be responsible for building V_k^A. To build V_k^A, we define a functional $\Psi^A(k, n)$ for $n < \Phi_e^A(k)$, and let $V_k^A = \cup_n \Psi^A(k, n)$.

As before, the negative nodes always wait for a *believable computation* before proceeding with their basic strategy. We arrange for several ultrahigh modules to be grouped into a *block*. The b^{th} block is denoted by $\mathcal{B}_b$. A key feature which ensured the successful combination of the ultrahigh strategy with a negative requirement, was the following: at an entire level devoted to the negative requirement, we needed to ensure that we only had one active restraint at any time. This was achieved in Section 4.3 by transferring the restraint (held by a minimal pair node) sideways; in Section 5.2.3 any restraint imposed by a tt-minimal pair node meant success for the entire level.

We discuss how to implement this feature in our construction. Suppose $\beta_0, \cdots, \beta_M$ are all the negative nodes on some level arranged from left to right, and $\Psi^A(k, 0), \Psi^A(k, 1)$ belong to a block of lower priority. Once some negative node β_m puts up a restraint, we ensure that every β_i to the right of β_m will also hold the same restraint as β_m, and does not put up any new restraint of its own. Consequently every β_i to the right will run the basic β_m-strategy, and enumerate the β_m-followers (instead of its own followers). This happens even though we might be currently to the right of β_m.

We call this *acting on β_m's behalf.* This remains true until we visit left of β_m, in which case a new restraint may be put up (by some β to the left of β_m). Thus if β_0 is the leftmost node visited infinitely often, and if β_0 ever puts up any restraint, then this restraint will be in force forever, so that $\Psi^A(k,1)$ will record the correct $\{k\}^{\emptyset'}(k)$-value. If β_0 never gets to put up a restraint, then $\Psi^A(k,0)$ will record the correct $\{k\}^{\emptyset'}(k)$-value. Each time β_0 is visited we will clear $\Psi^A(k,1)$ of any value it is currently recording. This is the reason why in the atomic strategy for $\mathcal{N}$, we only put up an A-restraint when all the followers have been picked.

The above extends to the n^{th} block as well, and generally we need $\Psi^A(k,0)$, $\cdots, \Psi^A(k, 2^n-1)$ for every k in $\mathcal{B}_n$, since we need to record all possible restraint-states that the higher priority negative requirements are in. Note that due to the arrangement of the blocks, generally the n^{th} negative requirement will be placed at a level much larger than $2n$, and so there can be much more than 2^{2n} many different versions of the negative requirement. Hence the number of mistakes which a module in $\mathcal{B}_n$ will make does not depend on the level at which it is operating at (as is the case in a high or superhigh construction), but rather on the *number of levels of negative requirements* which are placed before it.

10.1.4 Technical considerations on the tree

In this section we describe the technical aspects when combining the ultrahigh strategies with the negative strategies on the tree. Suppose β_0 is on the true path, and β_i to the right of β_0 is acting on behalf of β_0. Because β_0 and β_i are at different positions on the construction tree, we have to consider issues which will arise from enumerating β_0-followers when we are to the right of β_0. The first issue is the difference in believable computations. Remember that β_0 only puts up a restraint if all the β_0-followers have seen computations which are β_0-believable. In future when β_i enumerates these followers on behalf of β_0, we cannot expect that upon recovery, these followers will have β_0-believable computations. However note that the only reason why we enumerate any β_0-follower is to produce $W \oplus V$-changes. Hence even if the recovered computations were not β_0-believable (and may never become β_0-believable), it does no damage to β_0's strategy, since we would have obtained our desired $W \oplus V$-change.

We point out another situation where nodes on the right are allowed to injure nodes to the left. Suppose β is a negative node to the right of another negative node $\tilde{\beta}$ on the same level, and β was allowed to act on behalf of $\tilde{\beta}$. Because $\tilde{\beta}$-followers were picked very early, this action will injure nodes of every kind which are of lower $\tilde{\beta}$-priority. This can include nodes which are to the left of β, and even β itself. However this only happens finitely often to nodes on the true path (once all historical followers have been enumerated, we are safe).

10.1.5 Tree layout and notations

We now define the priority tree. If $|\alpha| = 2\langle e, i, j\rangle$ we assign the requirement $\mathcal{N}_{e,i,j}$ to it. There are two outcomes for α, w to the left of d. The main requirement $\mathcal{N}_e$ is satisfied via the actions of $\mathcal{N}_{e,i,j}$, and we do not require its presence on the tree. Nodes α of length $2\langle e, k\rangle + 1$ are assigned requirement $\mathcal{P}_e$ if $k = 0$, and to the subrequirement $\mathcal{P}_{e,k}$ if $k > 0$. $\mathcal{P}_e$-nodes are called *mother nodes*, with two outcomes $\infty <_L f$. If α is an $\mathcal{P}_{e,k}$-node and $\tau \subset \alpha$ is a $\mathcal{P}_e$-node, we say that τ is the *mother node of* α, and we denote $\tau = \tau(\alpha)$. If $\alpha <_L \beta$ are both $\mathcal{P}_{e,k}$-nodes such that $\tau(\alpha) = \tau(\beta)$, we say that α is a *left sibling node* of β.

The outcomes of a $\mathcal{P}_{e,k}$-node are $0 <_L 1 <_L 2 <_L \cdots$. Outcome 0 is a distinguished outcome, placed to the extreme left signifying the Σ^0_3-outcome of a global win. The other outcomes each represent the code number of a finite sequence of pairs $(n_0, x_0), (n_1, x_1), \cdots, (n_j, x_j)$ where the n_i's are distinct natural numbers, and $x_i \in \{\infty, f\}$, in some effective coding $\langle\cdot\rangle$ with range $\mathbb{N} \setminus \{0\}$. The only restriction we demand on the coding is to ensure that if σ and η are two such sequences, then $\langle \sigma, (n, \infty), \eta\rangle < \langle \sigma, (n, f), \eta\rangle$. For instance, we could just code the sequence $(0, f), (1, \infty), (2, f)$ as $2^1 3^0 5^1$. As before we say that *m specifies the pair* (k, x), if m is an outcome where (k, x) appears. We say that *m specifies the finite set* $A \subset \mathbb{N}$ if $A = \{k \mid m \text{ specifies } (k, f) \text{ or } (k, \infty)\}$.

To measure if Φ^A_e is an order, we define *the nondecreasing length of convergence* for Φ^A_e at stage s to be the usual

$$\ell_e[s] = \max\{x < s : (\forall y \le x)\Phi^A_e(y)[s] \downarrow \ge \Phi^A_e(y-1)[s] \downarrow\}.$$

This is measured at mother nodes, and we also write $\ell_\tau[s]$ in place of $\ell_e[s]$. We say that s is a *τ-expansionary stage* if $s = 0$ or $\ell_\tau[s] > \ell_\tau[s^-]$, where $s^- < s$ is the largest τ-expansionary stage before s. Generally if s is an α-stage, then we let s^- denote the previous α-expansionary stage, if α is a mother node, and the previous α-stage otherwise.

If τ is an $\mathcal{P}_e$-node, we define $m^\tau_b = \langle e, b\rangle - \langle e, 0\rangle$. This is basically the number of levels j between τ and its b^{th} block devoted to a negative requirement. A b-daughter node α of τ (i.e. a daughter node assigned $\mathcal{B}^\tau_b$) will be allowed $2^{m^\tau_b}$ many boxes at its level. We let $M^\tau_b = \{x : 2^{m^\tau_b} < x \leq 2^{m^\tau_{b+1}}\}$. Again if α is a $\mathcal{P}_{e,b}$-node with top τ, we write m^τ_α and M^τ_α in place of m^τ_b and M^τ_b. Note that $M^\tau_1, M^\tau_2, \cdots$ gives a fixed partition of a cofinite segment of $\mathbb{N}$. The b^{th} block $\mathcal{B}^\tau_b$ will contain all the modules k such that $\Phi^A_e(k) \in M^\tau_b$; this changes with time. Hence with each $\mathcal{B}^\tau_b$-node, we associate the parameter

$$L_\alpha[s] = \{k < \ell_\tau[s] : \Phi^A_e(k)[s] \in M^\tau_\alpha\}.$$

That is, $L_\alpha[s]$ gathers all the modules of τ which should be put into block $\mathcal{B}^\tau_b$ at stage s. All sibling nodes of α will have the same list L_α at all times, so they will all run the same modules.

If α is a $\mathcal{P}_{e,b}$-node, we denote the *label of* α by its position on the tree. This is denoted by lb_α, which is a finite string of length $m^{\tau(\alpha)}_\alpha$ with word $\{w, d\}$, and is determined by the outcomes of the negative nodes lying between $\tau(\alpha)$ and α, when read off starting from $\tau(\alpha)$. For example, if $\mathcal{N}_2, \mathcal{N}_3$ and $\mathcal{N}_4$ lie between $\tau(\alpha)$ and α, with outcomes w, d and w respectively, then $lb_\alpha = wdw$. Again we can partially order all such labels in $\{w, d\}^{<\omega}$ lexicographically, with w to the left of d; we also use $<_L$ for this ordering. If τ is a top node, and $\delta \supset \tau$ is any node (including negative nodes), we can extend the above definitions to define m^τ_δ and lb^τ_δ. In the above, lb_α is just a special case of $lb^{\tau(\alpha)}_\alpha$.

The point of introducing labels for an ultrahigh node α, is to easily express the following actions. Each α will be in charge of defining $\Psi^\tau(k, lb_\alpha)$ for every $k \in L_\alpha$. Here, Ψ^τ is a functional built at τ; we drop all mention of oracle A. In the end we let $V^\tau_k = \{\Psi^\tau(k, lb_\alpha) : \tau = \tau(\alpha) \text{ and } k \in L_\alpha\}$, so that $|V_k| \leq 2^{m^\tau_\alpha}$. We refer to each value $\Psi^\tau(k, \sigma)$ as a *box*, which we will fill with a number. Note that in this construction, the label or pointer lb_α is fixed; unlike the construction of a cappable ultrahigh set in

Section 4.3, where the label changes dynamically. Each ultrahigh node α will start work only if it is active. The label lb_α specifies the particular box at which α will fill or clear for every $k \in L_\alpha$. α will fill the box $\Psi^\tau(k, lb_\alpha)$ each time it plays an outcome specifying (k, f), and will clear $\Psi^\tau(k, lb_\alpha)$ each time the outcome specifies (k, ∞).

At some stage s, when we say that we *fill* $\Psi^\tau(k,\sigma)$ *with use* u, we mean that we enumerate the axiom $\langle\langle k,\sigma\rangle, \{k\}^{\emptyset'}(k)[s], A_s\lceil u+1\rangle$ into Ψ^τ. The s^{th} stage use of the computation $\Psi^\tau(k,\sigma)[s]$ is denoted by $u^\tau_{k,\sigma}[s]$. To *clear*, or to *empty* $\Psi^\tau(k,\sigma)$ *at a stage* s, means that we enumerate $u^\tau_{k,\sigma}[s]$ into A. As usual we associate finite strings with their code numbers. During a stage s of the construction in which we visit an ultrahigh node α, and we filled some box for the sake of α, we will say that the box was *filled by* α *at stage* s. This remains true until the box is next emptied.

We say that α *is active* at stage s, if τ allows α to run the modules in its instruction list L_α. As before, this is defined to be $\alpha \supseteq \tau^\frown\infty$ and we have all of the following:

- (AC.1): $\Phi_e^A(\ell_\tau)[s] \not\in M^\tau_\alpha$.
  [Hence we do not expect new numbers to appear in the instruction list $L(\alpha)$, unless there is an A-change.]

- (AC.2): $L_\alpha[s] \neq \emptyset$.
  [Otherwise α has nothing to do.]

- (AC.3): there is some largest $t < s$ which are both α-stages, $L_\alpha[t] = L_\alpha[s]$, and for every $k \leq 1 + \max L_\alpha[s]$, the computation $\Phi_e^A(k)[s]$ has persisted for at least two visits to α (i.e. $A\lceil\varphi_e(k)[t] = A\lceil\varphi_e(k)[s]$).
  [This ensures that the true path is well-defined. That is, we ensure that α switches from being active to inactive each time there is a change in some use in L_α.]

If α is active then it will play an outcome specifying L_α. We also say that α is *permanently active* at stage s, if it is active at every visit to α after stage s. We now define what it means for a computation measured by a negative node α to be believable.

If α is active, then α will continue tracing $\{k\}^{\emptyset'}(k)$-values for all $k \in L_\alpha[s]$. The uses of these traces are chosen fresh (in particular, larger than the Φ_e^A-uses), so that when α next plays the outcome 0, all of the previous traces would have been automatically cleared. This is important because each time the $\mathcal{B}_b^\tau$-node α plays outcome 0, there would be some rearrangement of τ-modules amongst $\mathcal{B}_b^\tau, \mathcal{B}_{b+1}^\tau, \cdots$. For instance, the k^{th} module may have moved from $\mathcal{B}_{b+1}^\tau$ to $\mathcal{B}_b^\tau$. It is important that when the $\mathcal{B}_b^\tau$-nodes begin the next tracing phase, they start with a clean slate.

Definition 10.1.2. For a negative node α, we say that a computation $\Delta^A(n)[s]$ with use u is *α-believable* at stage s, if all the following hold:

- for every ultrahigh node $\beta \subset \alpha$, and every k such that $\alpha(|\beta|)$ specifies (k, ∞), the box $\Psi^{\tau(\beta)}(k, lb_\beta)$ is either empty or has use $> u$.
- for every ultrahigh node β which has been visited prior to s, such that $\tau(\beta)^\frown\infty \subset \beta^\frown 0 \subseteq \alpha$, and all k, the box $\Psi^{\tau(\beta)}(k, lb_\beta)$ is either empty or has use $> u$.
- every box of lower $\alpha^\frown w$-priority is either empty, or has use $> u$.

At the negative node α, we measure the *believable length of agreement* for $A \equiv_T W_e \oplus V_e$ to be:

$$\ell_\alpha^b[s] = \max\{y < s \mid (\forall x < y)\ A_s(x) = \Phi_e^{W_e \oplus V_e}(x)[s]\ \wedge \\ (\forall z < \varphi_e(x)[s])\Delta_e^A(z)[s] = (W_e \oplus V_e)(z)[s]\}.$$

We require that all the Δ_e^A-computations mentioned are α-believable at s. We distinguish this from the *plain length of agreement*, which we denote simply as $\ell_\alpha[s]$, where the definition is as above except that we do not require the Δ_e^A-computations to be α-believable. If $x < \ell_\alpha[s]$ or $\ell_\alpha^b[s]$, then we denote the nested use by $\delta_e(\varphi_e(x))$. This is not to be confused with the length of convergence when τ is a mother node, which we also denote by ℓ_τ.

By the Recursion Theorem, we are supplied with an infinite list of indices x for which we are able to build the x^{th} Turing functional $J^X(x)$. When a negative node α *picks a fresh index* at stage s, we will pick for α an index from this list which is a fresh number. As usual, a fresh number at stage s is a number larger than s, and larger than anything mentioned so far. We now define the parameters needed for a negative node α. The indices used are: the special index η_0^α aimed at W_e', and the others denoted by $\eta_1^\alpha, \eta_2^\alpha, \cdots, \eta_{t_i(\eta_0^\alpha)}^\alpha$ which are all aimed at V_e'. The followers are denoted by $x^\alpha_1, x^\alpha_2, \cdots, x^\alpha_N$ where $N = t_i(\eta^\alpha_0)t_j(\eta^\alpha_{t_i(\eta^\alpha_0)})$. Note that N will be used throughout the construction and verification if everything converges. The indices and followers are picked in increasing order, i.e. $x^\alpha_p < x^\alpha_q$ and $\eta^\alpha_p < \eta^\alpha_q$ whenever $p < q$. We assume that $t_0, t_1, \cdots$ are all nondecreasing functions. Associated with each negative α is a state F_α. This may be either 1, 2 or 3 depending on how far along the basic strategy α is. $F_\alpha = 1$ means that α is still picking the indices η^α_n. State 2 means that α has picked its indices and is now in the process of setting up its followers. In state 3 means that α has begun diagonalization. Note that if α is in state 3, then it cannot tolerate any small A-change unless it is initialized.

During the construction, if s is an α-stage and some computation is convergent at s, then we say that this computation has *persisted for at least two visits to* α, if this computation is also convergent at the previous visit to α, and there has been no change below the use. Given a node α on the tree, and a box $\Psi^\tau(x, \sigma)$, we say that the box *is of lower* α*-priority*, if $\tau^\frown\infty \subseteq \alpha$ and $\sigma >_L lb^\tau_\alpha$.

When we initialize a node α at s, we do the following. If α is a mother node we make $\Psi^\tau(-)$ and u^τ_- undefined at all arguments. Ultrahigh nodes are not initialized in this way; the boxes they control will be cleared when a negative node with a higher priority label is visited. If α is a negative node, we make all the η^α_- and x^α_- undefined, and set $F_\alpha = 1$.

10.1.6 The construction

At stage 0, initialize every node and clear every box. At stage $s > 0$, we inductively define the stage s approximation to the true path, δ_s of length s, and we state the actions taken by the nodes on δ_s. Suppose that $\alpha = \delta_s \upharpoonright d$ has been defined for $d < s$. We will also define $o = \delta_s(d)$, the α-outcome. There are three cases:

1. *α is a $\mathcal{P}_e$-node.* We let $o = \infty$ if s is α-expansionary, and $o = f$ otherwise. Initialize all the nodes to the right of $\alpha^\frown o$ and clear every box of lower $\alpha^\frown o$-priority.

2. *α is a $\mathcal{P}_{e,i}$-node.* We first figure out what the outcome is at the current stage s. If α is currently not active, we let $o = 0$. Otherwise, for each $k \in L_\alpha[s]$, we have to let o specify either (k, ∞) or (k, f). Check if the following holds:

- (OUT.1): the box $\Psi^{\tau(\alpha)}(k, lb_\alpha)$ is either unoccupied, or it is occupied by the correct value (i.e. equals to $J^{\emptyset'}(k)[s]$).
- (OUT.2): $J^{\emptyset'}(k)[s]\downarrow$, and this computation has persisted for at least two visits to α. We let $t < s$ be the least stage such that the same computation $J^{\emptyset'}(k)[s]\downarrow$ also applies at t (i.e. $J^{\emptyset'}(k)[t]\downarrow$ with $\emptyset'_t\upharpoonright j(k)[t] = \emptyset'_s\upharpoonright j(k)[t]$).
- (OUT.3): we also require that for every $k' \in L_\alpha[s] - \{k\}$, such that $J^{\emptyset'}(k')[t]\downarrow$ and $\emptyset'_t\upharpoonright j(k')[t] \neq \emptyset'_s\upharpoonright j(k')[t]$, that the current stage s is not the first α-stage after the change. That is, there is a smaller α-stage u such that $t < u < s$, and $\emptyset'_t\upharpoonright j(k')[t] \neq \emptyset'_u\upharpoonright j(k')[t]$.

If all of the above hold for k, we let o specify (k, f), otherwise we let o specify (k, ∞). Specifically we let o = code of the pairs $(k_0, x_0), \cdots, (k_p, x_p)$, where k_i's are precisely all the distinct elements of $L_\alpha[s]$, and the x_i's are determined above. We explain the choices for the conditions OUT.1-3 above - these are implemented purely for technical reasons. OUT.1 ensures that if the box is occupied by the wrong value, then we immediately force outcome (k, ∞) to clear it. The combination of OUT.2 and 3 ensures that the true path is consistent with the "truth of outcome", as explained in the earlier section. We now take actions for α. Do all the following

- we clear all the boxes which need clearing. For each k such that o specifies (k, ∞), we clear $\Psi^{\tau(\alpha)}(k, lb_\alpha)$. Also clear every box of lower $\alpha^\frown o$-priority. Finally for each ultrahigh node $\beta >_L \alpha$ such that $\tau(\beta) \subset \alpha$, and $lb_\beta \not<_L lb_\alpha^{\tau(\beta)}$, and x such that the $\Psi^{\tau(\beta)}(x, lb_\beta)$-box is currently full and was filled by β, we clear it. Basically this last action clears all the boxes filled by nodes to the right.
- fill all the boxes which need filling, unless some box which is expected to be occupied is not. That is, if there is some ultrahigh node $\xi \subset \alpha$ such that the ξ-outcome along α specifies (k', f) for some k', but the box $\Psi^{\tau(\xi)}(k', lb_\xi)$ is currently empty, then we skip the following action for α: for every k such that o specifies (k, f), we fill $\Psi^{\tau(\alpha)}(k, lb_\alpha)$ with a fresh use (unless the box is already full). We do this to ensure that the use of α-boxes are always larger than the use of ξ-boxes.

- initialize all the nodes to the right of $\alpha^\frown o$. If this is not the first α-stage, we also initialize all the negative nodes to the left of α which have believed in incorrect computations. For each sibling node $\beta <_{left} \alpha$ or $\beta = \alpha$, and for each outcome $o' > 0$ such that
  - either o' and o do not specify the same set of numbers, or
  - there is some k such that o' specifies (k, f) and o specifies (k, ∞),

  we initialize every node extending $\beta^\frown o'$. Note that this is possible even if o' is placed to the left of o. We injure nodes to the left via this action; we have to show during the verification that nodes on the true path are injured by action to the right only finitely often.

3. *α is an $\mathcal{N}_{e,i,j}$-node.* If no left sibling of α is in state 3, we do the following, depending on the state F_α:

   - $F_\alpha = 1$: look for the least n such that η_n^α is undefined. If $n = 0$, we pick a fresh index for η_0^α. If $n > 0$ and $t_i(\eta_0^\alpha)\downarrow$, pick fresh indices for $\eta_1^\alpha < \cdots < \eta_{t_i(\eta_0^\alpha)}^\alpha$ and set $F_\alpha = 2$.
   - $F_\alpha = 2$: if one of $t_j(\eta_1^\alpha), \cdots, t_j(\eta_{t_i(\eta_0^\alpha)}^\alpha)$ has not yet converged, do nothing. Otherwise do the following. We want to get rid of the followers x where there has been a change in A below $\delta_e(\varphi_e(x))$. Look for the smallest n such that $x_n^\alpha \downarrow$ and $\ell_\alpha^b[s^-] > x_n^\alpha$ and $A{\upharpoonright}\delta_e(\varphi_e(x_n^\alpha))[s^-]$ has changed between s^- and s. Such a change is possible due to lower priority requirements acting. If such n exists, pick the least one and make all $x_{n+1}^\alpha, x_{n+2}^\alpha, \cdots$ undefined. Note that we *do not* undefine x_n^α itself.

     Next, pick the least $n \leq N$ such that x_n^α is undefined. If $n = 0$ we give x_n^α a fresh value, otherwise we give x_n^α a fresh value only if $\ell_\alpha^b[s] > x_{n-1}^\alpha$. Finally if there is no such n (i.e. all the followers have been defined) and $\ell_\alpha^b[s] > x_N^\alpha$, we set $F_\alpha = 3$, initialize every node extending α and clear every box of lower $\alpha^\frown w$-priority.
   - $F_\alpha = 3$: if $\ell_\alpha[s] \leq x_N^\alpha$, do nothing else (we wait for recovery). Also if $|T_i(\eta_0^\alpha)| \geq t_i(\eta_0^\alpha)$ or $|T_j(\eta_m^\alpha)| \geq t_j(\eta_m^\alpha)$ for some $m > 0$, we do nothing else (since the traces do not obey the bounds). Otherwise we let $k =$

$|T_i(\eta_0^\alpha)|+1$, and we will play the indices η_0^α and η_k^α. Also let x be the largest of the α-followers that has not yet entered A.

We check if either of $J^{W_e}(\eta_0^\alpha)[s]$ or $J^{V_e}(\eta_k^\alpha)[s]$ is undefined; if so define it (or them) to give output s with use $\varphi_e(x)[s]$, and do nothing else. Otherwise we have both convergent. In this case, we check if both $J^{W_e}(\eta_0^\alpha)[s] \in T_i(\eta_0^\alpha)$ and $J^{V_e}(\eta_k^\alpha)[s] \in T_j(\eta_k^\alpha)$. If both are in the respective traces, we enumerate x into A and initialize every node extending α. Otherwise do nothing.

Note that we use the believable length of agreement when setting up the followers in state 2, and we only talk about the plain length of agreement while diagonalizing in state 3. This is because in state 3 all we really want are $W_e \oplus V_e$-changes that follow the enumeration of a follower x into A, and we are not really interested in the actual recovery of the computation $\Delta_e^A(\varphi_e(x))$. Now we decide the α-outcome. If $F_\alpha = 3$ let $o = d$, otherwise the outcome is w. Initialize all the nodes to the right of $\alpha^\frown o$ and clear every box of lower $\alpha^\frown o$-priority.

If there is some left sibling of α in state 3, we let $\tilde{\alpha}$ be the left-most one. We take actions under state 3 above, replacing α by $\tilde{\alpha}$. The only difference is that if we enumerate some x into A, we will initialize every node $\beta \supset \tilde{\alpha}$ as well as $\beta >_L \tilde{\alpha}$. This may include nodes β of higher priority than α, and can even be α itself. In the verification we will show that this injury is finite if α is on the true path. We will call this action α *enumerates* x *on behalf of* $\tilde{\alpha}$. When α acts on behalf of $\tilde{\alpha}$, we are only waiting for the plain length of agreement $\ell_{\tilde{\alpha}} = \ell_\alpha$ to recover; it is unreasonable to wait for all these computations to become $\tilde{\alpha}$-believable while we are at α (nor is it necessary to do so). Let the outcome o of α be d, initialize every node to the right of $\alpha^\frown o$ and clear every box of lower $\alpha^\frown o$-priority.

10.1.7 Verification

It is easy to see that there is a leftmost path visited infinitely often: if α is an ultrahigh node and is never permanently active, then 0 is the leftmost outcome it plays infinitely often; on the other hand if α does become permanently active, then L_α eventually settles and α will have to play one of the $2^{|L_\alpha|}$ many possible outcomes specifying L_α. Let TP be the true path of the construction, defined as usual to be the leftmost path visited infinitely often. We let $True_\alpha$ be the true stage of α, i.e. least α-stage s such that $(\forall t > s)(\delta_t \not<_L \alpha)$. Clearly if $\alpha \subseteq \beta \subset TP$ then $True_\alpha \leq True_\beta$. If we ever visit left of a node α we will initialize α in the same stage. It is not hard to verify also that if $\alpha \subset \beta$ and α is initialized, then β is initialized by the same action as well.

We have two (possibly conflicting) concepts of "priority" for a node α on the construction tree. The first is determined by the position of α on the tree. The second is determined by the label of α (relative to some top node τ). It is possible for a node α to be physically left of another node α', but yet has a lower priority label than the label of α'. In the next lemma, we reconcile these conflicts; in particular we show that the only way for us to first visit α, and then visit α' later in the construction, is for us to injure α in the meantime.

Lemma 10.1.3. *Suppose α, α' are negative nodes which are siblings, s is a $\alpha^\frown d$-stage and $t > s$ is a $\alpha'^\frown w$-stage, and α has been visited prior to s. Then there is some $s \leq s' \leq t$ such that α is initialized at s'.*

Proof. If $\alpha' <_L \alpha$ then we are done, so assume that $\alpha' \geq_L \alpha$. Observe that the only way for any negative node to get out of state 3 is for it to be initialized. At stage s either $F_\alpha = 3$ or some left sibling of α is in state 3. If the former holds then α has to be initialized to make $F_\alpha \neq 3$ before α' is visited. Suppose the latter holds, and at s α acts on behalf of $\tilde{\alpha} <_L \alpha$. Similarly $\tilde{\alpha}$ has to be initialized at some s' between s and t. If this initialization of $\tilde{\alpha}$ was *not* because $\tilde{\alpha}$ believed in an incorrect computation, then it is trivial to verify that the same action would also initialize α to the right of $\tilde{\alpha}$. Therefore the initialization to $\tilde{\alpha}$ was because $\tilde{\alpha}$ had previously believed in an incorrect computation. That is, $\tilde{\alpha}$ extends some $\tilde{\beta}^\frown o'$ for some ultrahigh node $\tilde{\beta}$ and outcome $o' > 0$, which is deemed to be incorrect by $\delta_{s'} \upharpoonright |\tilde{\beta}|$ visited at s'. Now α also extends some $\beta^\frown o$ for $|\beta| = |\tilde{\beta}|$. If $\tau(\beta) \neq \tau(\tilde{\beta})$ then by the fact that the nodes $\delta_{s'} \upharpoonright |\beta|$ and $\tilde{\beta}$ are siblings, we have α initialized at s' since α is strictly right of $\delta_{s'}$. Hence we may assume that $\tau(\beta) = \tau(\tilde{\beta})$. We analyze o' and o. At s when $\beta^\frown o$ was visited we had not initialized $\tilde{\alpha}$, which implies that o and o' specify the same set (hence $o > 0$), and whenever o' specifies (k, f) we must also have o specifies (k, f). This means that at s', $\delta_{s'} \upharpoonright |\tilde{\beta}|$ would also consider β and o to be incorrect, and so α extending this will be initialized at s'. $\square$

For the next lemma 10.1.4, we let α be a $\mathcal{P}_{e,i}$-node and $\tau = \tau(\alpha)$ such that $\alpha \supseteq \tau^\frown\infty$. Assume α lies on the true path. Let $L := \lim_s L_\alpha[s]$, which may not exist. It is not hard to see that

- Φ_e^A is an order $\Rightarrow L$ exists and $L = \{k \mid \Phi_e^A(k) \in M_i^\tau\}$.
- α becomes permanently active iff L exists, $L \neq \emptyset$ and for all $k \leq 1 + \max L$, $\Phi_e^A(k) \downarrow$.
- If α never becomes permanently active, then 0 is the true outcome of α.

In the next lemma, we show that the true α-outcome is consistent with the "truth of outcome". This is instrumental in showing later that nodes on the true path are injured by action on the right only finitely often. In (ii) we show that if $J^{\emptyset'}(k) \downarrow$, then not only will the true outcome specify (k, f), but there will also be only finitely many outcomes (k, ∞) being played at the whole level. This ensures that boxes filled by a positive node on the true path, never become emptied if $J^{\emptyset'}(k) \downarrow$.

Lemma 10.1.4. *Suppose that α becomes permanently active.*

(i) Let $L = \{k_0, \cdots, k_p\}$. The true outcome of α is $(k_0, x_0), \cdots, (k_p, x_p)$ where for each i, $x_i = \infty$ iff $J^{\emptyset'}(k_i) \uparrow$.

(ii) For each $k \in L$ such that $J^{\emptyset'}(k) \downarrow$, there can only be finitely many stages t such that $\delta_t \supseteq \tau^\frown\infty$ and $\delta_t(|\alpha|)$ specifies (k, ∞).

(iii) For all but finitely many stages t, whenever $\delta_t \supseteq \tau^\frown\infty$ then

- *if t is the first visit to $\delta_t \upharpoonright |\alpha|$ then $\delta_t(|\alpha|) = 0$, and*
- *if t is not the first visit to $\delta_t \upharpoonright |\alpha|$ then $\delta_t(|\alpha|)$ specifies L, and furthermore $\delta_t(|\alpha|) \not<_L (k_0, x_0), \cdots, (k_p, x_p)$.*

Proof. (i) We partition L into the two parts, $L_\infty := \{k \in L \mid J^{\emptyset'}(k) \uparrow\}$, and $L_f := L - L_\infty$. We let $s_0 > True_\alpha$ be a large enough α-stage, such that

- L_α has settled,
- α is always active when visited,
- for each $k \in L_f$, $J^{\emptyset'}(k)\downarrow$ has stabilized on the correct use, and
- for each $k \in L_f$, the box $\Psi^\tau(k, lb_\alpha)$ is not occupied by an incorrect value. This is possible because if the box is occupied by a number other than $J^{\emptyset'}(k)$, then α will play an outcome specifying (k, ∞) every time it is visited until the box is cleared.

At every visit to α after s_0, α will play an outcome specifying L. To prove that the true outcome of α is $(k_0, x_0), \cdots, (k_p, x_p)$, we do it in two parts. Firstly, we show that for each $k \in L_f$, there are only finitely many stages such that α plays an outcome specifying (k, ∞). Secondly, we will show that there are infinitely many stages such that α plays an outcome specifying *all of* $\{(k, \infty) \mid k \in L_\infty\}$.

To prove the first part, we fix a $k \in L_f$. Let $s_1 \leq s_0$ be the least such that $J^{\emptyset'}(k)[s_1]\downarrow$ on the correct use. For each $k' \in L_\infty$ such that $J^{\emptyset'}(k')[s_1]\downarrow$, there must be a change in $\emptyset'$ below the k'-use after stage s_1. Wait until all these changes occur, for all these $k' \in L_\infty$, and we let $s_2 > s_0$ be a large enough α-stage after these changes. We claim that after stage s_2, any outcome played by α has to specify (k, f): Pick $k' \in L - \{k\}$, such that $J^{\emptyset'}(k')[s_1]\downarrow$ with $\emptyset'_{s_1} \upharpoonright j(k')[s_1] \neq \emptyset'_{s_2} \upharpoonright j(k')[s_1]$. We want to show that s_2 is not the first α-stage after the change. If $k' \in L_f$ then s_0 would be a better candidate instead of s_2, since any such change has to occur by stage s_0; on the other hand if $k' \in L_\infty$ then it follows by the choice of s_2 large.

To prove the second part, we let $s_3 > s_0$ be an α-stage. We show that there is a stage $s_4 \geq s_3$ such that α plays an outcome specifying *all of* $\{(k, \infty) \mid k \in L_\infty\}$. We may assume that $\tilde{L} := \{k \in L_\infty \mid J^{\emptyset'}(k)[s_3]\downarrow\} \neq \emptyset$ (otherwise we can let $s_4 = s_3$). For each $k \in \tilde{L}$, there is a least α-stage $s_4^k > s_3$ such that $\emptyset'$ would have changed below the use of $j(k)[s_3]$. Let $\tilde{k}$ be some member of $\tilde{L}$ with the largest $s_4^{\tilde{k}}$, and let $s_4 = s_4^{\tilde{k}}$. Thus s_4 is the largest α-stage amongst all the first changes. We argue that this choice of s_4 works, in particular we fix a $k \in L_\infty$ such that $J^{\emptyset'}(k)[s_4]\downarrow$, and we want to show that the outcome played by α at stage s_4 specifies (k, ∞). Assume the computation $J^{\emptyset'}(k)[s_4]\downarrow$ has persisted for two visits (otherwise it is trivial). Hence

$k \neq \tilde{k}$, and therefore condition (OUT.3) in the construction is not met when deciding the outcome for k, because s_4 is the first α-stage after a $j(\tilde{k})$-change. Hence (k, ∞) is played at stage s_4.

(ii) This is proved in a similar way as in the first part of (i). Fix a $k \in L$ such that $J^{\emptyset'}(k) \downarrow$, and let s_1, s_2 be as in (i). The stage s_2 works for (i), but not for us in this case; we have to wait until every right sibling node α' of α which had been visited prior to stage s_2, is visited again at least one more time (if ever). Say this happens by stage $s_5 > s_2$. Now the argument in (i) can be used to show that at every stage $t > s_5$ such that $\delta_t \supseteq \tau^\frown\infty$, we either have $\delta_t(|\alpha|) = 0$, or else $\delta_t(|\alpha|)$ specifies (k, f).

(iii) Immediate from (ii). □

The next lemma tells us that each time some computation in L_α changes, certain τ-boxes will automatically be cleared. That is, if k is not eventually in L_α, then all the $\Psi^\tau(k, \gamma)$-boxes with $|\gamma| = m_i^\tau$, which are incorrect coding locations for the k^{th} row, will be unoccupied in the limit. Part (ii) covers the case when k eventually does enter L_α (and is in L_α at every α-stage), but yet α has true outcome 0. This might be possible if we have infinitely many uses for $\Phi_e^A(k)[t]$, so that $\Phi_e^A(k) \uparrow$, but $\Phi_e^A(k)[t] \in M_i^\tau$ whenever it converges. Every time α plays outcome 0, this box might be occupied; unlike in (i), because some sibling node of α to the right might fill the box after each change in the use for $\Phi_e^A(k)[t]$. In this case we get a global win at τ, and τ would not care which boxes are filled; however this is important for the negative nodes in the construction tree extending $\alpha^\frown 0$ which might be on the true path. These negative nodes might see Δ-computations which they would like to freeze, but there is currently a use of some τ-box below the δ-use, which will later be cleared (by the lemma below). Hence these negative node would know not to believe in these Δ-computations until these τ-boxes are cleared.

Lemma 10.1.5. *Let α be a $\mathcal{P}_{e,i}$-node and $\tau = \tau(\alpha)$ such that $\alpha \supset \tau^\frown\infty$. Assume α lies on the true path and s is an α-stage which is not the first one.*

(i) If α plays outcome 0 at stage s, then for every number $k \notin L_\alpha[s]$ and string γ of length m_i^τ, the box $\Psi^\tau(k, \gamma)$ has to be empty at stage s.

(ii) For every k and string γ of length m_i^τ, if the box $\Psi^\tau(k,\gamma)$ gets filled by the actions at stage s (if not already full), then the box will have to be emptied at least once before α can next play outcome 0*.*

Proof. (i) Suppose the contrary, let $k \notin L_\alpha[s]$ and γ of length $|\gamma| = |lb_\alpha|$ be such that the box is full with use $u^\tau_{k,\gamma}[s]\downarrow$. This box must have been filled at some stage $t < s$, in which $\delta_t(|\alpha|)$ was active and $k \in L_\alpha[t]$. The fact that k is no longer in $L_\alpha[s]$, means that A has to change below t. Hence this change will happen after $u^\tau_{k,\gamma}[s] > t$ was set, a contradiction. In fact, we don't need that α plays outcome 0; this will always be true if k goes out of L_α.

(ii) Again the box must have been filled when $k \in L_\alpha[t]$. Before α can next play outcome 0 after stage s, there must be a change some computation in $L_\alpha[t]$, which would clear the box before the next $\alpha^\frown 0$-visit. Note that we are not claiming that the box is *empty* at the next $\alpha^\frown 0$-stage after s; just that the box would be emptied at least once in between. □

Now we will show that despite the fact that nodes can be injured by action on the right, this only happens finitely often to nodes on the true path.

Lemma 10.1.6. *Every node on the true path is initialized finitely often.*

Proof. We proceed by induction on the length. Let α be on true path, and suppose no node $\beta \subset \alpha$ gets initialized anymore. We consider all possibilities, i.e. we suppose α gets initialized by the actions of some node β. We may assume that $\beta \not<_L \alpha$. Since β cannot be a top node, we divide the proof into two parts. Firstly suppose that β is a negative node. If β is not acting on behalf of some sibling node, then we must have $\beta \subset \alpha$ and so this happens only finitely often (because β never picks new followers). Suppose β was acting on behalf of some sibling node $\tilde{\beta} <_L \beta$. We must have either $\tilde{\beta} \subset \alpha$ or else $\tilde{\beta} <_L \alpha$. If we let $X[t]$ be the collection of all numbers x such that x is a follower (at stage t) of some negative node in state 3, which is either to the left of α or it is extended by α. Observe that for all large enough t, $X[t]$ is nonincreasing, i.e. $X[t+1] \subseteq X[t]$. Now each time α is initialized by β acting on behalf of some $\tilde{\beta}$, this event will also correspond to some number in $X[t]$ being enumerated into A. So, this only happens finitely often (since $X[t]$ is finite).

Now for the second part we assume that β is an ultrahigh node. Hence β initializes α because it thought that α had believed in incorrect computations. So α extends some $\tilde{\beta}^\frown o'$ for some outcome $o' > 0$ and some sibling $\tilde{\beta} <_L \beta$ or $\tilde{\beta} = \beta$. Let o be the β-outcome played at stage t, when this initialization took place (at a very large stage t). Since $o' > 0$ and is the true outcome of $\tilde{\beta}$, it follows that $\tilde{\beta}$ will become permanently active, and by Lemma 10.1.4, and the fact that t is not the first visit to β, we must have both o and o' specify the same set. Hence there is some k where o' specifies (k, f) and o specifies (k, ∞), which is a contradiction. So after a large enough t no ultrahigh node can initialize α. $\square$

As an important consequence to Lemmas 10.1.3 and 10.1.6, we have that if $i < j$ then for almost every stage t, we have

$$lb^{TP\upharpoonright i}_{TP\upharpoonright j} \leq_L lb^{\delta_t\upharpoonright i}_{\delta_t\upharpoonright j}.$$

This will be assumed in the rest of the proof.

Lemma 10.1.7. *Along the true path, all the negative requirements are satisfied.*

Proof. Take an $\mathcal{N}_{e,i,j}$-node α on the true path, and suppose that α is never initialized after some stage s_0. Assume also that t_i and t_j are total, and $|T_i(x)| < t_i(x)$ and $|T_j(x)| < t_j(x)$ for all x. Also assume that $A = \Phi_e^{W_e \oplus V_e}$ and $W_e \oplus V_e = \Delta_e^A$. If any of the assumptions above are not met, then $\mathcal{N}_{e,i,j}$ is satisfied vacuously. It is easy to see that $\ell_\alpha \to \infty$.

Claim 1. There is a leftmost node β, such that $|\beta| = |\alpha|$ and $F_\beta = 3$ at almost all stages. If there is a left sibling node of α with this property, then we are done. Otherwise at almost every visit to α, we must in fact have that *no* left sibling of α is at state 3. Since α is never initialized after s_0, its state never decreases. We claim that $\lim F_\alpha = 3$. If not, then $\lim F_\alpha = 1$ or 2. It is obvious that $\lim F_\alpha = 1$ is not possible, so at some point all the indices will be picked and $\lim F_\alpha = 2$.

We argue inductively that for every $0 \leq i \leq N$, x_i^α will become eventually defined forever, and also that $\liminf \ell_\alpha^b > x_i^\alpha$. x_i^α will clearly be defined (and never again undefined) when $\Delta_e(\varphi_e(x_{i-1}^\alpha))$ has settled, and is believable. To see that ℓ_α^b has $\liminf$ greater than x_i^α, we suppose that there is there is some box with use $u_{k,\sigma}^\tau < \delta_e(\varphi_e(x_i^\alpha))$ where $\delta_e(\varphi_e(x_i^\alpha))$ is the correct use. We show that this box must be emptied at the next visit to α (if not before), which gives a contradiction. To see this, suppose the second option in Definition 10.1.2 fails (that the first option fails is trivial). We apply Lemma 10.1.5(ii) to get a contradiction. Suppose the third option fails. Since α always plays outcome w when visited, at the next α-stage this box will be cleared. This contradiction says that ℓ^b_α is almost always larger than x^α_i. Hence there will be a stage where we give F_α state 3, a contradiction.

Let s_2 be the final stage where β moves from state 2 to 3. It is easy to see that for all i, $x^\beta_{i+1} > \delta_e(\varphi_e(x^\beta_i))[s_2]$ and $\ell^b_\beta[s_2] > x^\beta_i$. By Lemma 10.1.3, for any $i < |\beta|$ and all $t \geq s_2$, we must have $lb^{\beta\restriction i}_{\beta^\frown d} \leq_L lb^{\delta_t\restriction i}_{\delta_t\restriction 1+|\beta|}$. We can in fact say something more. Suppose $\tau \subset \beta$ and $\xi \supset \tau$ is visited at some stage after s_2 such that $|\xi| > |\beta|$; then we have $lb^\tau_{\beta^\frown w} <_L lb^\tau_\xi$. This is because $lb^\tau_{\beta^\frown d} = (lb^\tau_\beta)^\frown d >_L (lb^\tau_\beta)^\frown w = lb^\tau_{\beta^\frown w}$.

Claim 2. After s_2, no A-change below $q := \delta_e(\varphi_e(x^\beta_N))[s_2]$ is possible apart from an enumeration of some x^β_i. Any such change has to be produced by the actions of some node ξ at stage $t \geq s_2$. Let o be the outcome played by ξ at t. Suppose ξ was clearing a box of lower $\xi^\frown o$-priority. The box would be $\Psi^\tau(k, \sigma)$ for some $\tau \subset \xi$ and k, and $\sigma >_L lb^\tau_{\xi^\frown o}$. We may assume that $\tau \subset \alpha, \beta$, otherwise the box is only filled after s_2 and poses no threat to us. By the above observation about the priority of labels, we must either have $lb^\tau_{\xi^\frown o} >_L lb^\tau_{\beta^\frown w}$ or else $lb^\tau_{\xi^\frown o} \subset lb^\tau_{\beta^\frown w}$. Thus σ is strictly to the right of $lb^\tau_{\beta^\frown w}$, which means that the box $\Psi^\tau(k, \sigma)$ is of lower $\beta^\frown w$-priority. Hence it would be cleared by β at s_2 itself, and the only possibility is if $\xi = \beta$ and $t = s_2$. By β-believability this action does not change A below q.

Suppose on the other hand, ξ is a negative node which changed $A\restriction q$ because it was clearing a box of lower $\xi^\frown w$-priority (as it moved from state 2 to 3). At t, ξ is visited and it moves from state 2 to 3. Also, the outcome of ξ at t is d. As above, the box being cleared is $\Psi^\tau(k, \sigma)$ for some $\tau \subset \xi, \alpha, \beta$ and k, and $\sigma >_L lb^\tau_{\xi^\frown w} = lb^\tau_\xi{}^\frown w$. If $|\xi| > |\beta|$ then it is easy to see once again that the box $\Psi^\tau(k, \sigma)$ is of lower $\beta^\frown w$-priority. We suppose on the other hand that $|\xi| \leq |\beta|$; and show that this is impossible. Since at t, ξ would initialize every node extending it, we must in fact have $\beta <_L \xi$. We may assume $lb^\tau_{\xi^\frown d} \subseteq lb^\tau_{\beta^\frown d}$ (because the alternative $lb^\tau_{\xi^\frown d} >_L lb^\tau_{\beta^\frown d}$ is treated as above), it is not hard to verify that after s_2, ξ to the right of β can never play outcome w again. However observe that at s_2, ξ would be initialized since it is to the right of β. The only way for us to have stage t after s_2, is for ξ to promote its state first from 1 to 2 at some stage t' strictly between s_2 and t. At such a stage t', ξ must be acting for its own sake, and be having outcome w, a contradiction. This shows that the box $\Psi^{\tau}(k,\sigma)$ is of lower $\beta^\frown w$-priority, and will be cleared at s_2, a contradiction to the existence of ξ and t.

Now we can assume ξ changed A for a different reason (other than clearing boxes of lower priority). Hence ξ has to be either an ultrahigh or a negative node. Suppose first that ξ is a negative node. At t, ξ can either be acting for itself, or it is acting on behalf of some sibling $\tilde{\xi}$. Consider first the case when ξ is acting for itself. Since ξ cannot be initialized at s_2, $\xi \not\supset \beta$ and $\xi \not>_L \beta$. ξ also cannot be to the left of β, and if $\xi \subset \beta$ then it would initialize β after s_2. The only possibility is $\xi = \beta$, which means that one of the x_i^β would be enumerated. If ξ was acting on behalf of $\tilde{\xi}$ then a similar argument applies with $\tilde{\xi}$ in place of ξ.

Now suppose that ξ is an ultrahigh node, so that $u_{k,lb_\xi}^{\tau(\xi)}[t]$ is enumerated into A, for some k such that o specifies (k,∞). Again we may assume that $\tau(\xi) \subset \beta$ otherwise $u_{k,lb_\xi}^{\tau(\xi)}[t]$ is picked after s_2. If $|\xi| > |\beta|$ then we have $lb_\xi >_L lb_{\beta^\frown w}^{\tau(\xi)}$, so it would have been cleared by β at s_2, and so has use at stage t larger than q. On the other hand suppose $|\xi| < |\beta|$. Let $\tilde{\xi} = \beta\upharpoonright|\xi|$. Since β cannot be initialized by ξ at stage t, the outcome of $\tilde{\xi}$ along β is either 0 or else it specifies (k,∞). In both cases, the use cannot be less than q, again due to β-believability.

Lastly suppose that ξ is an ultrahigh node clearing a box filled by some ultrahigh node ξ' to its right. The box is again of the form $\Psi^{\tau(\xi')}(x, lb_{\xi'})$ for some x and $lb_{\xi'} \not<_L lb_\xi^{\tau(\xi')}$. Note that ξ' must have filled this box at a stage strictly before s_2. Again we may assume that $\tau(\xi') \subset \beta$; we divide into three cases. Firstly if $|\xi| < |\beta|$, then $\beta\upharpoonright|\xi|$ will be strictly to the left of ξ', and at s_2 when $\beta\upharpoonright|\xi|$ is visited we cannot have $lb_{\xi'} <_L lb_{\beta|\xi|}^{\tau(\xi')}$, otherwise we will also have $lb_{\xi'} <_L lb_\xi^{\tau(\xi')}$. This means that the box $\Psi^{\tau(\xi')}(x, lb_{\xi'})$ will be cleared by the action of $\beta\upharpoonright|\xi|$ at s_2, which is a contradiction. For the second scenario we assume that $|\xi| > |\beta| > |\xi'|$. Then we have $lb_\xi^{\tau(\xi')} >_L lb_{\beta^\frown w}^{\tau(\xi')}$. In this case, $\beta\upharpoonright|\xi'|$ is visited at s_2 and is strictly to the left of ξ', and again we cannot have $lb_{\xi'} <_L lb_{\beta|\xi'|}^{\tau(\xi')}$, so that the box $\Psi^{\tau(\xi')}(x, lb_{\xi'})$ will also be cleared by the action of $\beta\upharpoonright|\xi'|$ at s_2. Lastly both $|\xi'|, |\xi| > |\beta|$. Then neither $lb_{\xi'} <_L lb_{\beta^\frown w}^{\tau(\xi')}$ nor $lb_{\xi'} \supseteq lb_{\beta^\frown w}^{\tau(\xi')}$ is possible since $lb_{\beta^\frown w}^{\tau(\xi')} <_L lb_\xi^{\tau(\xi')}$. Since ξ' is long, we also cannot have $lb_{\xi'} \subset lb_{\beta^\frown w}^{\tau(\xi')}$. The last case is $lb_{\beta^\frown w}^{\tau(\xi')} <_L lb_{\xi'}$, in which case the box $\Psi^{\tau(\xi')}(x, lb_{\xi'})$ is of lower $\beta^\frown w$-priority, so that it will be cleared by β at s_2. This concludes the verification of claim 2.

Note that we are not claiming that $\delta_e(\varphi_e(x^\beta_N))[t] = q$ for every t. In fact, the nested use $\delta_e(\varphi_e(x^\beta_N))[t]$ can become larger than q when x^β_N enters A. Subsequently we may have other nodes changing A below $\delta_e(\varphi_e(x^\beta_N))[t]$, but always doing so above q. By Claim 2 we have that at every stage t after s_2 and for every k, whenever $x^\beta_k \notin A_t$ then $\varphi_e(x^\beta_k)[t] = \varphi_e(x^\beta_k)[s_2]$ and $\delta_e(\varphi_e(x^\beta_k))[t] = \delta_e(\varphi_e(x^\beta_k))[s_2]$, and furthermore the respective sets have not changed up to these uses. This is because if $W_e \oplus V_e$ changes without being permitted by us, then β (or any sibling acting on behalf of β) immediately becomes inactive to preserve the disagreement.

We also have infinitely many chances to act for β, or act on behalf of β. Suppose some largest x^β_c is never enumerated into A. Let $k = |T_i(\eta^\beta_0)| + 1$ (the actual size), and $k \leq t_i(\eta^\beta_0)$. Since $\varphi_e(x^\beta_c)$ settles, it follows that both $J^{W_e}(\eta^\beta_0)$ and $J^{V_e}(\eta^\beta_k)$ are eventually defined. This means that one of the two final values will not be in the respective traces, so $\mathcal{N}_{e,i,j}$ is satisfied. We may therefore assume that every one of the followers is enumerated into A. Let t_c be the stage when x^β_c is enumerated into A. We prove the crucial claim:

Claim 3. Between t_{c+1} and t_c, either $T_i(\eta^\beta_0)$ has a new number enumerated into it, or else some number strictly between t_{c+1} and t_c is enumerated into one of $T_j(\eta^\beta_1), \cdots, T_j(\eta^\beta_{t_i(\eta^\beta_0)})$. Let $k = |T_i(\eta^\beta_0)[t_{c+1}]|+1$, and suppose that no new number enters $T_i(\eta^\beta_0)$ between the two stages. Now both $J^{W_e}(\eta^\beta_0)[t_{c+1}]$ and $J^{V_e}(\eta^\beta_k)[t_{c+1}]$ are defined, and are in the respective traces. Since these computations were defined before t_{c+1} (say at t'), it follows that they have use at least as large as the use $\varphi_e(x^\beta_{c+1})[t']$. By the remarks after Claim 2, it follows that $\varphi_e(x^\beta_{c+1})[t'] = \varphi_e(x^\beta_{c+1})[t_{c+1}]$. This is the crucial part where we use Claim 2. There must be a recovery of l_β some time between t_{c+1} and t_c (in which β or some sibling acting on behalf of β gets to act). At that stage we have either W_e-change (hence $J^{W_e}(\eta^\beta_0)\uparrow$) or V_e-change (hence $J^{V_e}(\eta^\beta_k)\uparrow$). Now we cannot have a W_e-change, because otherwise we give $J^{W_e}(\eta^\beta_0)$ a new value which will be in $T_i(\eta^\beta_0)[t_c]$. Hence we must have a V_e-change at recovery, and we will give it a new value which will be reflected in $T_j(\eta^\beta_k)$.

By a simple counting argument and Claim 3, it follows that we cannot have enumerated all of $x^\beta_1, \cdots, x^\beta_N$, giving a contradiction. This shows that $\mathcal{N}_{e,i,j}$ is satisfied. □

Lemma 10.1.8. *Along the true path of construction, all the positive requirements are satisfied.*

Proof. Let τ be a $\mathcal{P}_e^A$-node on the true path, such that Φ_e^A is an order. The true τ-outcome is clearly ∞. We will show that $\{V_k^\tau\}_{k\in\mathbb{N}}$ traces $J^{\emptyset'}$ correctly. We fix a number k, and let i be such that $\Phi_e^A(k) \in M_i^\tau$, and α be the version of $\mathcal{P}_{e,i}^A$ on the true path with true outcome o. Hence α will eventually be responsible for tracing $J^{\emptyset'}(k)$, when L_α settles.

We first of all show that $|V_k^\tau| < \Phi_e^A(k)$. Take a box in the k^{th} row, i.e $\Psi^\tau(k,\sigma)$ for some σ. If this is filled by a node β where $|\beta| \neq |\alpha|$, then β must have done so when k was in L_β. Since k has to leave L_β, the box must be cleared before that. Therefore any such box $\Psi^\tau(k,\sigma)$ on row k can only be defined permanently by a node at level $|\alpha|$. There are at most $2^{m_i^\tau}$ many distinct labels of the nodes at that level, so $\Psi^\tau(k,\sigma)$ is only defined permanently in at most $2^{m_i^\tau} < \Phi_e^A(k)$ many different σ. Next, suppose that $J^{\emptyset'}(k)\downarrow$. We want to show that $J^{\emptyset'}(k) \in V_k^\tau$. We know α will become permanently active and o will specify (k,f). It is enough to prove the following claim (by induction):

Claim 1. For any ultrahigh node α along the true path and any k such that the true α-outcome specifies (k,f), the box $\Psi^{\tau(\alpha)}(k,lb_\alpha)$ receives a permanent definition with value $J^{\emptyset'}(k)$. We proceed by induction on the length of α. Take α on the true path with true outcome o specifying (k,f) for some k, and $\tau = \tau(\alpha)$. Clearly $\{k\}^{\emptyset'}(k)\downarrow$; let s_0 be a large $\alpha^\frown o$-stage. By induction hypothesis at s_0, α would not be stopped from putting $J^{\emptyset'}(k)$ into the box $\Psi^\tau(k,lb_\alpha)$, unless of course the box $\Psi^\tau(k,lb_\alpha)$ itself is occupied. In that case it has to be occupied by the correct value, since α only plays an outcome specifying (k,∞) finitely often. This can in fact be said about every $\alpha^\frown o$-stage after s_0; we must have the box $\Psi^\tau(k,lb_\alpha)$ occupied (possibly filled by α). The only issue is whether we can get it to be occupied permanently without being made undefined infinitely often. We assume for a contradiction that infinitely often we have the box $\Psi^\tau(k,lb_\alpha)$ being made undefined. In particular, there are infinitely many α-stages s and some stage $t \geq s$ such that an enumeration of a number $p \leq u_{k,lb_\alpha}^\tau[s]$ is made at stage t. In the following, s is an arbitrary $\alpha^\frown o$-stage.

- p is the use of a box $\Psi^{\tau'}(k', \sigma)$ which is filled by a node ξ with $|\xi| > |\alpha|$. We claim that if $\Psi^{\tau'}(k', \sigma)$ was filled after s_0, then it would not be possible for it to do the above. In particular, assume that $p \leq u^{\tau}_{k,lb_\alpha}[s]$ is enumerated at t for some $s_0 \leq s \leq t$ and s is an $\alpha^\frown o$-stage. Say ξ is an ultrahigh node which filled p, which must be at or before s. ξ cannot be to the right of $\alpha^\frown o$, because at s, α would clear the ξ-box. $\xi \not\subseteq \alpha$ because ξ is long. So $\xi \supseteq \alpha^\frown o$ is the only possibility left. Since ξ would have filled p strictly before s, and also the fact that when ξ filled p (at $s' < s$), we ensured that $u^{\tau}_{k,lb_\alpha}[s'] \downarrow$, we have $u^{\tau}_{k,lb_\alpha}[s'] < p \leq u^{\tau}_{k,lb_\alpha}[s]$, which means an A-change below p will take place before s, a contradiction.

- p is the use of a box $\Psi^{\tau'}(k', \sigma)$ of lower $\beta^\frown o'$-priority (when $\beta^\frown o'$ is visited). We cannot have $\tau' <_L \alpha$ because we never visit left of α. We cannot have $\tau' >_L \alpha$ because τ' is initialized at s. We cannot have $\tau' \supset \alpha$ as well, because then the box will be filled by some τ'-daughter of longer length than α. Suppose now that $\tau' \subset \alpha$. We have $\sigma >_L lb^{\tau'}_{\beta^\frown o'}$. If $|\sigma| \geq |lb^{\tau'}_{\alpha^\frown o}|$, then σ has to be filled by a τ'-daughter node of a longer length than α, which is already considered above. So we must have $|\sigma| < |lb^{\tau'}_{\alpha^\frown o}|$, and by Lemma 10.1.3 we also have $\sigma >_L lb^{\tau'}_{\alpha^\frown o}$ and so the box $\Psi^{\tau'}(k', \sigma)$ also of lower priority than $\alpha^\frown o$. This means the box is cleared at s, and by considering labels, p cannot be picked by α itself. Hence p is picked strictly after s, which is a contradiction.

- Suppose p is enumerated by an ultrahigh node β. The ultrahigh node β can be enumerating p either because p is the use of some box filled by a node to its right, or it is enumerating p because p is the use of a box which its outcome is telling it to clear. Suppose the latter holds. p is the use of some box, which is filled by some ultrahigh node ξ at some stage $t' \leq s$. We claim that if p is filled after s_0, then this will lead to a contradiction. We draw attention once again to the fact that at s, α will always clear all boxes it needs to clear, before filling the box $\Psi^{\tau}(k, lb_\alpha)$. There are now three cases depending on the position of ξ; remember that the box $\Psi^{\tau(\xi)}(k', lb_\xi)$ is filled by ξ and has to be cleared by β which is a sibling node of ξ. We want to get a contradiction in all three cases:

(C1) $\xi \subseteq \alpha$: then ξ is along the true path, so by Lemma 10.1.5(ii) the true outcome of ξ cannot be 0. Hence ξ will become permanently active and the true outcome of ξ has to specify either (k', ∞) or (k', f). By Lemma 10.1.4, the true outcome of ξ has to specify (k', ∞), since β has to play (k', ∞). This means that at s itself p would have to be cleared, a contradiction. This applies even if $\xi = \alpha$.

(C2) $\xi >_{left} \alpha$: at s, we would have cleared the box with use p when α is visited. This happens before α fills any box, so is a contradiction.

(C3) $\xi \supset \alpha$: then $|\xi| > |\alpha|$, and we have been through this case.

Suppose now the former holds, i.e. β is enumerating p because it is the use of a box filled by some node ξ to its right. In this case, it is not hard to see that the only possibilities for ξ are (C2) and (C3) above, and the same argument there applies.

- Lastly p is enumerated by a negative node β. β can either be enumerating some follower (of its own or some left sibling), or it could be clearing a box of lower $\beta^\frown w$-priority as it moved from state 2 to 3. Suppose the latter holds, i.e. β is clearing the box $\Psi^{\tau'}(k', \sigma)$. If $|\beta| > |\alpha|$ then the argument will be similar to the case when a box of lower $\beta^\frown o'$-priority is cleared. Suppose $|\beta| < |\alpha|$, and in fact we may assume that $lb^{\tau'}_{\beta^\frown d} \subseteq lb^{\tau'}_{\alpha}$ (because the alternative $lb^{\tau'}_{\beta^\frown d} >_L lb^{\tau'}_{\alpha}$ can be treated as above). This is now dealt with in a similar fashion as one of the cases in Lemma 10.1.7. After s, β to the right of α can never play outcome w again. At s, β would be initialized since it is to the right of α. β must promote its state first from 1 to 2 at some stage t' strictly between s and t. At such a stage t', β must be acting for its own sake, and be having outcome w, a contradiction.

Assume further that s_0 is large enough such that p cannot be enumerated under the situations described above. We consider the following ordering of nodes on the construction tree, by letting $\zeta_1 <_P \zeta_2$ iff either $\zeta_1 <_L \zeta_2$ or else $\zeta_1 \subset \zeta_2$. Let ξ be the least negative node wrt $<_P$ extending $\alpha^\frown o$, with a follower enumerated into A after s_0 (this may be due to ξ or some sibling node acting on behalf of

ξ); if ξ does not exist then we are done. If only finitely many ξ-followers are enumerated into A, let $s_1 > s_0$ be the first $\alpha^\frown o$-stage after the last enumeration of a ξ-follower. Otherwise if infinitely many ξ-followers are enumerated into A, then ξ must be to the right of the true path; let $s_1 > s_0$ be an $\alpha^\frown o$-stage such that we moved left of ξ.

We claim that no p below $u^\tau_{k,lb_\alpha}[s_1]$ can ever enter A after s_1. This can only be due to the enumeration of some β-follower for some $\beta \supseteq \alpha^\frown o$ (again this can be due to either β or some β-sibling). If ξ was to the right of the true path then it is not hard to see that no β extending $\alpha^\frown o$ can have a follower enumerated at or after s_1. In the other case if the final ξ-follower was enumerated prior to s_1, then every negative node extending $\alpha^\frown o$ and $>_P \xi$ would be initialized when the ξ-follower was enumerated, and have to pick their followers larger than $u^\tau_{k,lb_\alpha}[s_1]$. This is because if a negative node is initialized, then within the same stage, it will not pick new followers (being in state 1).

Hence the box $\Psi^\tau(k, lb_\alpha)$ will receive a permanent definition. This completes the induction step for Claim 1. Hence A is ultrahigh. $\square$

Question 10.1.9. *Which c.e. degrees are the join of two superlow c.e. degrees? For instance, can any incomplete non-superlow c.e. degree be the join of two superlow c.e degrees?*

10.2 Every high c.e. degree can be split into two array computable c.e. degrees.

Following the theme in the previous section, we investigate which degrees can be split into smaller degrees with low computational strength. Note that neither the low c.e. sets nor the array computable c.e. sets are contained in each other; however every superlow set is array computable. Therefore a natural direction one could take is to ask which c.e. degrees are the join of two smaller c.e. degrees which are low and array computable (this is a weaker property than being the join of two superlow c.e. degrees). We show that highness is enough to guarantee the ability to split in this way. In particular, high permitting allows us to disengage the coding markers almost every time, and will be enough to ensure that $\mathbf{a_0}$ and $\mathbf{a_1}$ are both low and array computable. In contrast, as we have seen in the previous section, the failure of $\mathbf{a}$ to even permit a small number of times (even in an "ultrahigh permitting") was enough to destroy superlow requirements. This again highlights an important fundamental difference between being superlow and being low+array computable; the former is much less tolerant to injuries.

Theorem 10.2.1. *Every high c.e. degree is the join of two array computable c.e. degrees. Furthermore the two degrees can be made low.*

As a corollary we mention that this theorem (together with Theorem 10.1.1) gives us a c.e. set which is low and array computable, but not superlow.

10.2.1 Requirements

Let C be a maximal c.e. set. We build two c.e. sets A_0 and A_1, and a Turing reduction Δ such that $C = \Delta^{A_0 \oplus A_1}$. As a convention, we let lowercase Greek letters denote use functions. The reduction Δ is built by movable markers, with the usual marker rules. That is, $\delta(n)[s] \downarrow$ iff there is a $\Delta(n)$-computation which currently applies at s. Before a marker is lifted or moved, it has to be first made undefined. A marker $\delta(n)$ which is defined at s will be made undefined if some number $x \leq \delta(n)[s]$ is enumerated into A_0 or A_1.

To make $A_i \leq_T C$ we use high permitting via the fact that the principal function $p(n)$ of $\overline{C}$ is dominant. We ensure that if we enumerate x into A_i at stage s then $p_{s-1}(n) \neq p_s(n)$ for some $n \leq x$; this clearly ensures that $A_i \leq_T C$. Since every c.e. traceable set is array computable, we meet the requirements

$$\mathcal{R}_{e,i}: \text{ If } \Phi_e^{A_i} \text{ is total, then build a c.e. trace } T(x) \text{ for } \Phi_e^{A_i}(x)$$
$$\text{with at most } h(x) \text{ many mind changes.}$$

Here h is a computable function to be defined later. We do not arrange for priority amongst the requirements $\mathcal{R}_{e,i}$. Each $\mathcal{R}_{e,i}$ is split into modules $M_{e,i,0}, M_{e,i,1}, \cdots$, with $M_{e,i,k}$ being in charge of tracing $\Phi_e^{A_i}(k)$. We write $M_{\langle e,i,k\rangle}$ instead of $M_{e,i,k}$. We arrange instead for priority amongst the modules, and we order the modules in the order $M_0 < M_1 < \cdots$ (lower numbers have higher priority). The module M_n

will disengage the $\delta(n)$-marker before tracing a convergent computation. In order to distinguish between two computations $\Phi_e^{A_i}(x)[s]$ and $\Phi_e^{A_i}(x)[t]$ with different use, but yet having the same output, we will trace in $T(x)$ the use of computations (as finite strings), instead of their output values.

10.2.2 Description of strategy

The key driving force behind many theorems which degree split $\emptyset'$, is the idea of *disengagement of markers.* For instance, if we wanted to show that $\emptyset' \equiv_T A_0 \oplus A_1$ where A_0 and A_1 are superlow, we would wait for jump computations $J^{A_0}(x)$ to converge, and then enumerate the marker $\delta(x)$ into the other side A_1 to lift the use $\delta(x) >$ use of $J^{A_0}(x)$. Only then do we believe the jump computation $J^{A_0}(x)$ and trace it; this ensures that we make computably boundedly many mistakes in tracing the jump computations. The completeness of $\emptyset'$ allows us to enumerate $\delta(x)$ whenever we want.

Let us suppose that we wanted to show instead that *every* c.e. degree C was the join of two superlow degrees (which is of course false). In addition to having to disengage markers, we now have to build A_0 and A_1 below C. In particular, before we are allowed to disengage $\delta(x)$ from below the use of some jump computation $J^{A_i}(x)$, we have to wait for a C-change. The reader might imagine that if C was close to the Halting problem, such as being high, then this would be enough. Informally the highness of C allows us to force C to change below almost every challenge we issue it. A plausible plan would then be the following: when we see a jump computation $J^{A_i}(x)$ converge and we need to disengage $\delta(x)$, we will issue a challenge for C to change below $p(\delta(x))$. Since C will permit at almost all of these challenges, we will be able to disengage almost every marker we wish, so that almost every convergent jump computation is traced.

The above plan cannot work, since of course not every high c.e. set is the join of superlow sets. The problem is, for instance $J^{A_i}(x)$ converges before $J^{A_i}(y)$ for $x > y$. When we challenge $p(\delta(x))$ to change, we have to simultaneously challenge $p(\delta(y))$ to change. This is because we have to define some total computable function f which p will dominate, so we cannot leave $f(\delta(y)) \uparrow$. The problem is that we are forced to issue a challenge for $p(\delta(y))$ when we were not ready; when $p(\delta(y))$ changes later to witness the domination of p, we are left with a dilemma: if we choose not to lift $\delta(y)$ to a fresh value, and keep $\delta(y)$ the same, then we lose the ability to challenge $p(\delta(y))$ to change again, if $J^{A_i}(y)$ converges later. On the other hand if we always lift $\delta(y)$ to a new value and $J^{A_i}(y)$ never converges, then $\delta(y)$ will be driven to infinity.

However note that the above problem is really due to the fact that we were trying to trace partial functions. If we were only required to trace total functions, then this issue does not arise. We would challenge $p(\delta(y))$ to change only when $\Phi^{A_i}(x)$ has converged for every $x \leq y$. Each module $M_{e,i,x}$ will run the following basic strategy: first it waits for $\Phi_e^{A_i}(z)$ to converge for every $z \leq x$. Then we enumerate a challenge $f_{e,i}(\delta(e,i,x)) \downarrow$ and larger than $p(\delta(e,i,x))$. Then we wait for C to permit, namely for $p(\delta(e,i,x))$ to enter C. When this happens we enumerate $\delta(e,i,x)$ into A_{1-i} to disengage the marker and then trace the computation.

The basic strategy is clear. We now address the technical issues arising from combining the different modules. The only numbers we enumerate into A_0 or A_1 are marker values. There are two reasons why we enumerate $\delta(n)$; the first is due to coding when n has entered C. This happens at most once for each $n = \langle e,i,x\rangle$. The other reason is due to the module M_n when C has permitted on M_n's challenge. Note that C can take as long as it likes before permitting M_n's challenge. In the meantime M_n will have a convergent computation $\Phi_e^{A_i}(x)$ which is waiting for disengagement; we call computations of this type *pending computations*. Each time a pending computation is destroyed, we might get C-permission to disengage $\delta(n)$ before $\Phi_e^{A_i}(x)$ has re-converged. In this case we have to move $\delta(n)$ to a fresh value anyway, because we want to able to allow M_n to issue a new challenge when $\Phi_e^{A_i}(x)$ converges again. Hence we have to ensure that M_n preserves pending computations while it is waiting for disengagement.

There are two reasons why a module M_n wants to limit the changes in A_0 or A_1. The first is for us to count the number of injuries to a traced computation; this is to make A_i c.e. traceable. The second is to ensure that pending computations are destroyed only finitely often; as mentioned above this is to ensure that δ-markers settle. We only care about having an effective bound for injuries of the first type. We do not need, and in fact cannot have an effective bound for injuries of the second type.

During the construction each module acts according to the basic strategy above. A traced computation can be destroyed by any action. A pending computation can only be destroyed by actions of lower priority modules. To determine an upper bound on the number of times a traced computation may be destroyed, we calculate how many times each marker $\delta(n)$ is moved, where $n = \langle e, i, x\rangle$. It is moved once due to coding. Suppose it is moved by M_n. If this action successfully disengages $\delta(n)$ from below a pending M_n-computation, then we get a bound by working inductively. On the other hand when $\delta(n)$ was moved, it might be that $\Phi_e^{A_i}(x) \uparrow$. In this case we can also proceed inductively since pending computations are only destroyed by the actions of higher priority requirements. Lastly when $\delta(n)$ is enumerated it might be that $\Phi_e^{A_i}(x) \downarrow$, but we are not able to disengage $\delta(n)$ because some higher priority requirement $M_{n'}$ has a pending computation with use larger than $\delta(n)$. In this case we enumerate $\delta(n)$ into A_i destroying the convergent $\Phi_e^{A_i}(x) \downarrow$ without tracing it. We then lift $\delta(n)$ above the use of the pending $M_{n'}$-computation. If this computation is never again injured, then $M_{n'}$ will never block M_n from subsequent disengagement attempts. If on the other hand the pending $M_{n'}$-computation is injured it has to be through the movement of one of $\delta(0), \cdots, \delta(n-1)$; and we can use the inductive bounds on these.

10.2.3 Notations

We write $\delta(e, i, k)$ instead of $\delta(\langle e, i, k\rangle)$. Each requirement $\mathcal{R}_{e,i}$ defines a computable function $f_{e,i}$ that serves as a challenge for C-permitting. We also build the trace $T_{e,i}(x)$ for $\Phi_e^{A_i}(x)$. We define $\ell_{e,i}[s] = \max\{y < s \mid (\forall x < y)\Phi_e^{A_i}(x)[s] \downarrow\}$.

A computation is said to be *traced* at a stage s, if it is $\Phi_e^{A_i}(x)[s] \downarrow$ for some e, i, x, and the use of the computation is in $T_{e,i}(x)$. A computation $\Phi_e^{A_i}(x)[s]$ is *pending* at a stage s, if $\ell_{e,i}[s] > x$ for some e, i, x, and $f_{e,i}(\delta(e, i, x))[s] \downarrow$ and it is not yet traced. Basically, pending computations are those convergent computations where a challenge has already been issued for C to change, and we are waiting for the chance to disengage the appropriate δ-marker from below the use. In both cases we also say that $M_{e,i,x}$ has been traced or is pending, for the appropriate $M_{e,i,x}$. The use of a pending $M_{e,i,x}$-computation is simply $\varphi_e^{A_i}(x)[s]$.

At a stage s, we say that $M_{e,i,x}$ *requires attention* if $\ell_{e,i}[s] > x$, $\delta(e, i, x)[s] \downarrow$,

$M_{e,i,x}$ is not traced and $f_{e,i}(\delta(e,i,x))[s]\uparrow$. A fresh number at stage s is a number larger than s, and not yet mentioned so far.

10.2.4 The construction

At stage s, the actions are divided into the following phases. Only the modules M_n for $n < s$ are considered.

1. *challenge phase:* we look for the least module $M_{e,i,x}$ requiring attention (if any). For every $z \leq \delta(e,i,x)[s]$ such that $f_{e,i}(z)$ is not yet defined, we set $f_{e,i}(z)[s]\downarrow$ to a fresh number and larger than $p(\delta(e,i,x))[s]$.

2. *coding phase:* look for the least n such that $p_{s-1}(n) \neq p_s(n)$ (i.e. $p_{s-1}(n)$ has entered C). We need to enumerate $\delta(p_{s-1}(n))$ into either A_0 or A_1: find the highest priority $M_{e,i,x}$ that is pending, such that $\delta(p_{s-1}(n)) <$ the use of the pending computation. Enumerate $\delta(p_{s-1}(n))$ into A_{1-i}, and into A_0 if $M_{e,i,x}$ does not exist.

3. *disengagement phase:* let n be as above, i.e. least such that $p_{s-1}(n)$ has entered C. Find the highest priority $M_{e,i,x}$ such that $\delta(e,i,x)\downarrow\geq n$, $M_{e,i,x}$ is not traced and $f_{e,i}(\delta(e,i,x))[s]\downarrow$. If such $M_{e,i,x}$ exists, we have to enumerate $\delta(e,i,x)$ into either A_0 or A_1; the decision as to which side to enumerate into, is based on the following: find some $M_{e',i',x'}$ (of the highest priority) which is of priority no lower than $M_{e,i,x}$ such that $M_{e',i',x'}$ is pending, and $\delta(e,i,x) <$ the use of the pending computation. If $M_{e',i',x'}$ exists, enumerate $\delta(e,i,x)$ into $A_{1-i'}$ otherwise enumerate it into A_0.

4. *tracing phase:* for every module $M_{e,i,x}$ such that $\ell_{e,i}[s] > x$, and either $\delta(e,i,x)\uparrow$ or $\delta(e,i,x)\downarrow> \varphi_e^{A_i}(x)$, we enumerate the use of the computation into $T_{e,i}(x)$.

5. *extension phase:* find the first m such that $\delta(m)$ is undefined. Give $\delta(m)$ a fresh value and set $C_s(m) = \Delta^{A_0\oplus A_1}(m)$ with use $2\delta(m)+1$.

10.2.5 Verification

It is obvious that $A_0 \oplus A_1 \leq_T C$ by permitting via $p(n)$, because $\delta(p_{s-1}(n))[s] > p_{s-1}(n) \geq n$ for any n and s. We next prove the crucial lemma.

Lemma 10.2.2. *There is a computable function $\tilde{h}$ such that the number of times $\delta(n)$ is enumerated into A_0 or A_1 is at most $\tilde{h}(n)$.*

Proof. We proceed by induction on $n = \langle e, i, x\rangle$, and define $\tilde{h}$ inductively. Let $\tilde{h}(0) = 1 + 1 = 2$; it is not hard to see (by following a similar argument as the inductive step below) that this bound is sufficient. This is because $\delta(0)$ is enumerated under the coding phase at most once, and enumerated under case (i) or (ii) below at most once. Case (iii) is not possible.

Now consider $n = \langle e, i, x\rangle > 0$, and suppose that $H = 1 + \tilde{h}(0) + \cdots + \tilde{h}(n-1)$ have all been defined. $\delta(n)$ can be enumerated under the coding phase at most once. Suppose $\delta(n)$ is enumerated under the disengagement phase, at stage s. At s one of the following three scenarios must hold:

(i): there is some $M_{n'}$ of highest priority $n' \leq n$ which is pending at s, and $\delta(n) <$ use of the $M_{n'}$-pending computation.

(ii): not (i) and $\ell_{e,i}[s] > x$.

(iii): otherwise.

If we enumerate $\delta(n)$ under scenario (i) and for some n', then in order for us to enumerate $\delta(n)$ again under (i) and the same n', we must have one of $\delta(0), \cdots, \delta(n-1)$ being enumerated. Hence the total number of times we enumerate $\delta(n)$ due to (i) is at most $(n+1)H$. Note that $\ell_{e,i}[s]$ might not be larger than x. Suppose we enumerate $\delta(n)$ under scenario (ii). Then $M_{e,i,x}$ is pending at s since $\ell_{e,i}[s] > x$, so the enumeration of $\delta(n)$ does not kill the pending $M_{e,i,x}$-computation, because of the failure of (i). Hence we would trace $\Phi_e^{A_i}(x)[s]$ under the tracing phase at s , and we never enumerate $\delta(n)$ under (ii) again unless one of $\delta(0), \cdots, \delta(n-1)$ is enumerated. The total number of times we enumerate under scenario (ii) is at most H. Consider scenario (iii); we claim that this is not possible.

Let t_1 be a stage where we enumerate $\delta(n)$ under (iii), and we want to get a contradiction. At t_1, $f_{e,i}(\delta(e,i,x))[t_1]$ is defined. Since uses are always chosen fresh, we let $t_0 < t_1$ be the stage where $f_{e,i}(d)$ is defined, where $d = \delta(e,i,x)[t_1]$. The action of defining $f_{e,i}(d)$ immediately makes $M_{e,i,x}$ pending. This pending computation has to be destroyed, say at t' between t_0 and t_1, in order for (i) and (ii) to fail at t_1. At

t' there has to be a pending $M_{n'}$-computation of strictly higher priority $n' < n$ such that $\delta(n) <$ the use of the pending $M_{n'}$-computation. Otherwise M_n is the highest priority module being considered at t', and the action will not be allowed to injure M_n. We let N be the module of the highest priority with a pending computation with use larger than d between t' and t_1. Clearly $N \geq n' > n$. Before t_1, no marker $\leq d$ can be enumerated, lest $\delta(n)$ is moved. Hence at t_1, M_N cannot be traced, $f_{b_0,b_1}(\delta(N))$ is defined where $N = \langle b_0, b_1, b_2\rangle$, and furthermore $\ell_{b_0}^{A_{b_1}}[t_1] > b_2$ also holds because M_N is the highest priority module with the above-mentioned properties (hence cannot be injured). Hence at t_1, M_N is pending and in fact will still have pending computation with use larger than d. This shows that (i) must hold at t_1. The resulting contradiction shows that stage t_1 does not exist, and (iii) cannot be possible. Totalling up the figures obtained from (i) and (ii), we can safely define $\tilde{h}(n) = 1 + (n+2)H$. □

Therefore each $\delta(n)$ eventually becomes defined and settles. Hence $\Delta^{A_0 \oplus A_1}$ is total, and equals C.

Lemma 10.2.3. *Both A_0 and A_1 are low.*

Proof. Let $Q(z,t) = 1$ if $\ell_{e,i}[t] > x$, where $z = \langle e, i, x\rangle$, and let $Q(z,t) = 0$ otherwise. As usual we assume the hat trick. Suppose for every $z' < z$, both $\delta(z')$ and $Q(z',t)$ have settled. Also assume that no number less than any convergent $\Phi_{e'}^{A_{i'}}(x')$ is ever enumerated (in either side) where $\langle e', i', x'\rangle < z$. Let $z = \langle e, i, x\rangle$. Suppose $\ell_{e,i}[t] > x$ at infinitely many stages t. If $M_{e,i,x}$ is ever traced then the traced computation would be preserved forever (since $\delta(0), \cdots, \delta(z-1)$ have settled). Hence $Q(z) = 1$ forever. Suppose $M_{e,i,x}$ is never traced. Then $M_{e,i,x}$ will be pending eventually at one of these stages t. It is simple to verify that A_i cannot change below the use of the pending $M_{e,i,x}$-computation at t ever again. Hence $Q(z)$ settles.

The fact that A_0 and A_1 are low follows from the fact that $Q(\langle g(e), i, g(e)\rangle)$ settles, where we define $\Phi_{g(e)}^X$ to be total if $e \in X'$ and is empty if $e \notin X'$. □

Again we draw the reader's attention to the following fact. We are only interested in getting an effective bound for the number of times a traced computation can be destroyed. We do not need to compute a bound for the number of times a *pending* *computation* can be destroyed; indeed there cannot be an effective bound for the number of changes in $Q(z)$, otherwise A_0 and A_1 would be superlow. Observe that $Q(z)$ can change from 1 to 0 because some higher priority pending computation is blocking a coding attempt; the number of times this happens, though finite, will depend on the use of higher priority pending computations which we cannot expect to know beforehand.

We now prove that A_0 and A_1 are both c.e. traceable. It is obvious that $|T_{e,i}(x)| \leq h(e,i,x)$ where $h(0) = 1$ and $h(n+1) = 1 + \tilde{h}(0) + \cdots + \tilde{h}(n)$. Finally suppose that $\lim_s \ell_{e,i}[s] = \infty$. We suppose there are infinitely many x such that $\Phi_e^{A_i}(x) \notin T_{e,i}(x)$ (more precisely the use). Fix one such x, and let $d = \lim \delta(e,i,x)$. Suppose d was picked under the extension phase at some stage s. Clearly $M_{e,i,x}$ is not traced at s because $\delta(e,i,x)$ has settled down at s. In fact $M_{e,i,x}$ can never be traced at any stage after s. At s we pick d fresh so $f_{e,i}(d)[s] \uparrow$. It is not hard to see that $M_{e,i,x}$ has to receive attention under the challenge phase at some time t after s, where we will define $f_{e,i}(d) > p_t(d)$. We claim that none of $p_t(0), \cdots, p_t(d)$ gets enumerated into C after t: if one of them enters C then we will enumerate d (or some number smaller than d) into A_0 or A_1, violating the fact that $\delta(e,i,x)$ is stable. This means that infinitely often $f_{e,i}(d)$ is defined and is $> p(d)$, which means that $f_{e,i}$ is total computable and witnesses the fact that p fails to be dominant. This contradiction shows that almost all $\Phi_e^{A_i}(x)$ have to be traced.

10.3 A high$_2$ c.e. degree which cannot be split into smaller totally ω-c.e. degrees

In the previous section we showed that every high c.e. degree could be split into smaller c.e. degrees which were array computable. One might ask if high permitting was necessary; a weak highness property such as nonlow$_2$ might suffice. Perhaps even every degree was the join of two array computable degrees. We show that:

Theorem 10.3.1. *There is a high$_2$ c.e. degree $\boldsymbol{a}$ such that whenever $\boldsymbol{a} = \boldsymbol{a_0} \cup \boldsymbol{a_1}$, then one of $\boldsymbol{a_0}$ and $\boldsymbol{a_1}$ is not totally ω-c.e.*

We divide the discussion of the proof of the theorem in the following parts. First we discuss in Section 10.3.1, the plan on how to build a c.e. set A and make $deg(A)$ not contain the join of totally ω-c.e. sets. We then discuss in Section 10.3.2 why injecting highness requirements into the construction will fail, and discuss why high$_2$ requirements would be compatible.

10.3.1 Making $deg(A)$ not contain the join of any totally ω-c.e. sets.

We now need to satisfy the requirements

$$\mathcal{N}_e : \text{If } A = \Gamma_e^{W_e \oplus V_e} \text{ and } W_e \oplus V_e = \Delta_e^A, \text{ then one of } W_e \text{ or } V_e \text{ is not totally } \omega\text{-c.e..}$$

We drop subscript e for the purpose of the following discussion. To satisfy $\mathcal{N}$ we need to ensure that there are *total* functions F^W, G^V computable from W and V respectively, such that one of F^W or G^V is not ω-c.e..

We fix an effective enumeration $\langle a_i, b_i \rangle$ of all possible ω-c.e. approximations. That is, each $a_i(-,-)$ is a total computable function, and $b_i(-)$ is a partial computable function. We say that a function F is *i-approximated* if b_i is total and for every x, the number of mind changes on $a_i(x,-)$ is bounded by $b_i(x)$, and $\lim_s a_i(x,s) = f(x)$. Every ω-c.e. function is i-approximated for some i.

The general strategy for making a set X not superlow is very similar to making X not totally ω-c.e.. Indeed it is easy to see that if X is a c.e. set, then X is superlow iff for every (partial) function Φ^X, there is some i such that Φ^X is i-approximated at every place it is defined. Our strategy in Section 10.1.2 was able construct *partial functions* F^W and G^V such that one of them was not i-approximated for every i.

The extra difficulty posed by having to ensure totality is the following: suppose we have selected n_1 and n_2 with the intention of making one of $F^W(n_1)$ or $G^V(n_2)$ change often. We have to define $F^W(n_1)$ and $G^V(n_2)$ even though the respective b_i-bounds might not converge. We might therefore set the use of $F^W(n_1)$ and $G^V(n_2)$ to be above $\varphi(x_1), \cdots, \varphi(x_k)$, but later after more of b_i has converged we might find that k was insufficient, because we might have to force changes in W or V more than k times. We have to again exploit the fact we are allowed a nonuniform argument;

that is, we are allowed to build more than one pair $\langle F^W, G^V\rangle$ of functions as possible candidates to meet $\mathcal{N}$.

The requirement $\mathcal{N}$ will build a pair of functions $\langle F^W, G^V\rangle$ which will be total in the case when the $\mathcal{N}$-hypothesis is true. The requirement $\mathcal{N}$ is divided into infinitely many subrequirements $\mathcal{N}_{i,j}$, and each of these subrequirements $\mathcal{N}_{i,j}$ will build a pair of functions $\langle F^W_{i,j}, G^V_{i,j}\rangle$. The subrequirement $\mathcal{N}_{i,j}$ will itself be divided into infinitely many sub-subrequirements $\mathcal{N}_{i,j,k,l}$, which are each aiming to ensure that, if the $\mathcal{N}$-hypothesis is correct, then either F^W is not i-approximated, or G^V is not j-approximated, or $F^W_{i,j}$ is not k-approximated, or $G^V_{i,j}$ is not l-approximated.

The atomic action for $\mathcal{N}_{i,j,k,l}$ is the following.

1. Pick a follower x for A, and fresh numbers n_1, n_2, n_3, n_4. Wait for $\gamma(\delta(x))[s]\downarrow$, and then define all of $F^W(n_1), G^V(n_2), F^W_{i,j}(n_3), G^V_{i,j}(n_4)$ convergent with use $\gamma(\delta(x))[s]$. Freeze A.

2. Wait for all of $b_i(n_1), b_j(n_2), b_k(n_3), b_l(n_4)$ to converge. Enumerate x into A and wait for $W \oplus V$-change.

3. Say W changed in step 2. Now pick $x_1, \cdots x_{b_i(n_1)}$ such that $x_{m+1} > \gamma(\delta(x_m))$ for all m; define $F^W(n_1)$ on new use $\gamma(\delta(x_{b_i(n_1)}))$. Pick a fresh follower $\tilde{n}_2 > n_2$, and define $G^V(\tilde{n}_2)$ with use $\gamma(\delta(x_{b_i(n_1)}))$. Leave $F^W_{i,j}(n_3)\uparrow$. Increase restraint on A.

4. Wait for $b_j(\tilde{n}_2)$ to converge. Once it converges run the basic strategy in Section 10.1.2 to try and make F^W not i-approximated at input n_1. This process will succeed unless it is interrupted by a V-change. Go to next step if this happens.

5. Now pick fresh followers $y_1, \cdots y_M$ (where $M = b_j(\tilde{n}_2) + b_k(n_3)$) such that $y_{m+1} > \gamma(\delta(y_m))$ for all m; define $F^W_{i,j}(n_3)$ and $G^V(\tilde{n}_2)$ with use $\gamma(\delta(y_{b_j(\tilde{n}_2)}))$. Increase restraint on A. Run the basic strategy in Section 10.1.2 to make either $F^W_{i,j}$ not k-approximated at input n_3, or G^V not j-approximated at input $\tilde{n}_2$.

If the $\mathcal{N}$-hypothesis holds, then all γ, δ-uses will converge and the appropriate followers will be chosen when required. Hence we will not wait forever in steps 1 and 3. If we wait forever in step 2 then we would succeed trivially at one of $F^W, G^V, F^W_{i,j}$

or $G^V_{i,j}$. If we wait forever in step 4 then we either succeed trivially at $G^V(\tilde{n}_2)$, or we succeed at $F^W(n_1)$. If we ever get to step 5 then we will succeed at either $F^W_{i,j}(n_3)$ or $G^V(\tilde{n}_2)$.

The basic module succeeds at one of $F^W, G^V, F^W_{i,j}$ or $G^V_{i,j}$ as described above. Note that regardless of the outcome of the basic strategy, F^W and G^V will be defined at every input picked by the strategy. Furthermore if the basic strategy succeeds at neither F^W nor G^V, then both $F^W_{i,j}(n_3)$ and $G^V_{i,j}(n_4)$ are defined. Therefore the functions F^W, G^V built at $\mathcal{N}$ will be total regardless of the outcomes of individual sub-subrequirements. We analyze how the $\mathcal{N}$-requirement will be met. If $\forall i \exists j, k, l$ such that $\mathcal{N}_{i,j,k,l}$ succeeds at F^W, then F^W is not ω-c.e.. If $\exists i \forall j \exists k, l$ such that $\mathcal{N}_{i,j,k,l}$ succeeds at G^V, then G^V is not ω-c.e.. If neither of the alternatives above hold, then we must have some i_0, j_0 such that $\forall k, l$, $\mathcal{N}_{i_0,j_0,k,l}$ succeed at neither F^W nor G^V; hence both $F^W_{i_0,j_0}$ and $G^V_{i_0,j_0}$ will be total. In this situation if $\forall k \exists l$ such that $\mathcal{N}_{i_0,j_0,k,l}$ succeeds at $F^W_{i_0,j_0}$, then $F^W_{i_0,j_0}$ is not ω-c.e.; otherwise $G^V_{i_0,j_0}$ is not ω-c.e..

10.3.2 Making A high$_2$.

Suppose we wanted to combine the requirements in Section 10.3.1 with highness requirements. Since each atomic node $\mathcal{N}_{i,j,k,l}$ only increases A-restraint finitely often, and changes A finitely often, it might seem trivial to combine this with highness requirements. However we know this to be impossible from Theorem 10.2.1, so what goes wrong? The conflict is between an $\mathcal{N}_{i,j,k,l}$-node σ, and a highness node η such that $\sigma \supseteq \eta^\frown\infty$; i.e. σ believes that η wants to code an infinite column into A. Now σ will pick its follower(s) x_σ only at an η-expansionary stage; at such a stage s η would have enumerated all the contents of a column up to a large number $s' > s$, and x_σ chosen less than s'. The trouble is that $\gamma(\delta(x_\sigma))$ may later converge to a value larger than s'. Remember we had to ensure that the functions $F, G, F_{i,j}, G_{i,j}$ had to be total; therefore we cannot afford to wait until σ is next visited and $\Gamma(\Delta(x_\sigma))$ becomes σ-believable before defining $F, G, F_{i,j}, G_{i,j}$ (note that we could do this in Theorem 10.1.1 because the functions built there were not required to be total). Hence at the next η-expansionary stage, η would be unable to code below $\gamma(\delta(x_\sigma))$ without injuring σ. This could happen for infinitely many different sub-subrequirement nodes extending $\eta^\frown\infty$; which would be bad for η.

The inability to combine highness requirements with certain other requirements can sometimes be avoided if *capricious destruction* helps us make overall progress in the global setting; by doing so we get a high$_2$ set instead of a high set. For instance, no nonbounding set can be high, but one can overcome the difficulty of coordinating the local permitting with high coding by capriciously destroying certain computations with pending action. We refer the reader to Downey, Lempp and Shore [DLS93] for a more detailed discussion of a high$_2$ construction. We will only briefly describe the main points here.

To make A high$_2$ we need to code $Cof = \{e : dom(\varphi_e) \text{ is cofinite}\}$ into A''. A typical $\mathcal{P}_e$-mother node τ in the construction will measure if $e \in Cof$, and constructs a functional $H^A_\tau(i,t)$ such that H^A_τ is total and computable from A, and $e \in Cof$ iff $\lim_i \lim_t H^A_\tau(i,t) = 1$, and $e \notin Cof \Leftrightarrow \lim_i \lim_t H^A_\tau(i,t) = 0$. If we could do this, then to compute if $e \in Cof$, we observe that A'' can compute the true path of construction; so we can find the version of τ on the true path and then ask about the double limit.

The mother node τ will guess at the three quantifier question "$e \in Cof$" by breaking it up into infinitely many two quantifier questions, and approximating these individual questions at sub-nodes below it. To wit, each sub-node $\eta = \eta(i)$ will measure if there is $x > i$ such that $\varphi_e(x)\uparrow$. η will have an expansionary stage whenever it is visited and sees more of φ_e convergent above input i. η will then change A below some $H^A_\tau(i,-)$-use to try and redefine more of $H^A_\tau(i,-)$ to 1. At non-expansionary stages, η will extend $H^A_\tau(i,t)$ to more inputs t with output 0. Eventually η will want to ensure that if there are infinitely many η-expansionary stages then it records the $\lim_t H^A_\tau(i,t)$ as 1, otherwise it wants to record $\lim_t H^A_\tau(i,t)$ as 0

Note that in a high construction, *every* σ has to record the limit faithfully. However in a high$_2$ construction, we only need the limit $\lim_t H^A_\tau(i,t)$ to be recorded correctly at almost every level i below τ. We now discuss how to exploit this to make our construction work. Recall that the conflict was between a $\mathcal{N}_{i,j,k,l}$-node σ, and a high$_2$ node η such that $\sigma \supseteq \eta^\frown\infty$. Let $\tau(\eta)$ be the mother node of η, and $\tau(\sigma)$ be the master $\mathcal{N}$-node of σ. We describe two typical situations that will arise in the construction:

Case 1. $\tau(\sigma) \subset \tau(\eta) \subset \eta$*:* in this situation σ picks its follower x_σ as usual at η-expansionary stages. However whenever $\tau(\sigma)$ is visited and witnesses the convergence of $\gamma(\delta(x_\sigma))$, it will pre-empt η and immediately enumerate all pending η-markers $< \gamma(\delta(x_\sigma))$ into A and make more of $H^A_{\tau(\eta)}(\eta, -) = 1$. That is, we capriciously destroy the $\Gamma(\Delta(x_\sigma))$-computation in the hope of clearing the η-markers from below the use. We then travel the link down to σ. This happens even at non-η-expansionary stages. If the link between $\tau(\sigma)$ and σ is travelled infinitely often and never removed, then we will be forced to make $\lim_t H^A_{\tau(\eta)}(\eta, t) = 1$ even though η might want to record the limit as 0; however observe that in this situation $\gamma(\delta(x_\sigma))$ tend to ∞, so that we have a global $\tau(\sigma)$-win. In the cone below σ, no other daughter node of $\tau(\eta)$ will be forced to have an incorrect limit. On the other hand once the link between $\tau(\sigma)$ and σ is removed, then before a new link can be formed between $\tau(\sigma)$ and another $\sigma' \supset \eta^\frown\infty$, we must have a new η-expansionary stage. Therefore if we skip over η infinitely often by jumping from τ to σ' (for different σ''s), then we forced to make $\lim_t H^A_{\tau(\eta)}(\eta, t) = 1$ without a global $\tau(\sigma)$-win; however in this case η also wants to record the limit as 1, so we are fine. The point is that $\lim_t H^A_{\tau(\eta)}(i, t)$ will be forced to record an incorrect value for finitely many i.

Case 2. $\tau(\eta) \subset \tau(\sigma) \subset \eta$*:* we ensure that the following feature is present in the tree architecture: suppose there are daughter nodes η_1, η_2 of $\tau(\eta)$ such that $\tau(\sigma) \subset \eta_1{}^\frown w_1 \subset \eta_2{}^\frown w_2$, and that $w_1 \neq w_2$. Whenever we see this we will restart a new version of $\tau(\sigma)$ below $\eta_2{}^\frown w_2$ (as we do in a global win); along the true path $\tau(\sigma)$ will be restarted finitely often because the sub-nodes of $\tau(\eta)$ can only have finitely much alternation in their true outcomes. Therefore if σ is an active sub-node of $\tau(\sigma)$, and $\tau(\sigma)$ is the final version of the requirement on the true path, then we may delay extending the definition of F^W, G^V until the next σ-stage when the uses become η-believable.

The formal construction will require links, and *scouting reports*. That is, before following a link from τ down to σ, we check to see where we would go if the link was not present; if we wanted to go left of σ, then we cancel the link and let the construction take us there. The full details and construction will appear in the paper currently in preparation.

Chapter 11

Deepness and cupping with strong reducibilities

11.1 An almost superdeep degree

In [BM], Bickford and Mills defined a c.e. degree $\boldsymbol{a}$ to be deep, if for every c.e. degree $\boldsymbol{b}$, we have $\boldsymbol{b}' = (\boldsymbol{a} \cup \boldsymbol{b})'$. That is, the degree $\boldsymbol{a}$ is deep in the sense that the operation of joining with $\boldsymbol{a}$ preserves the jump. They asked if a noncomputable deep degree exists. Lempp and Slaman [LS89] answered this question by showing that there are no noncomputable deep degrees.

Cholak, Groszek and Slaman [CGS01] tried to salvage this notion of deepness, by restricting the class of degrees whose jump is to be preserved under joining with $\boldsymbol{a}$. That is, they called a c.e. degree $\boldsymbol{a}$ *almost deep*, if for every low c.e. degree $\boldsymbol{b}$, $\boldsymbol{b}' = (\boldsymbol{a} \cup \boldsymbol{b})'$; equivalently $\boldsymbol{a} \cup \boldsymbol{b}$ is low for every low c.e. degree $\boldsymbol{b}$. Their interest in this problem was aroused by the ongoing search for (naturally) definable ideals in the c.e. degrees. They showed that there was indeed a noncomputable almost deep degree. They then asked if one could obtain a smaller nontrivial ideal by requiring the jump to be preserved on a larger class, say the low$_2$ degrees. This was answered by Jockusch, Li and Yang [JLY04], where they showed that for every noncomputable c.e. degree $\boldsymbol{a}$, there is a low$_2$ c.e. degree $\boldsymbol{b}$ such that $\boldsymbol{a} \cup \boldsymbol{b}$ is high. This implied that the low and low$_2$ c.e. degrees were not elementarily equivalent, and that the notion of almost deep is already the best way of salvaging the original concept of deepness.

In this chapter we consider a different variation on deepness, which we call *almost* *superdeepness.*

Definition 11.1.1. We define a c.e. set A to be *almost superdeep*, if for every superlow c.e. set W, the join $A \oplus W$ is superlow.

A c.e. degree $\boldsymbol{a}$ is almost superdeep, if it contains an almost superdeep member. In Theorem 11.1.2 we construct a noncomputable almost superdeep degree:

Theorem 11.1.2. *There is a noncomputable c.e. set A, such that for any superlow c.e. set W, the join $W \oplus A$ is superlow.*

We mention that since this result was announced, several other related results were obtained. Greenberg and Nies showed that every strongly jump traceable c.e. set was almost superdeep. Since the hyper jump traceable c.e. sets are never promptly simple (Theorem 3.7.1), it is tempting to make the following conjecture:

Conjecture 11.1.3. *Every hyper jump traceable c.e. set is of almost deep degree.*

Ng [Ngb] has shown this tempting conjecture to be false. In an upcoming paper, Diamondstone and Ng [DN] show that the none of the various ideals (of the almost deep c.e. degrees, the almost superdeep c.e. degrees, or the K-trivial c.e. degrees) were contained in any of the others.

We now turn to proving Theorem 11.1.2. There is a uniform enumeration $\{d_e\}_{e\in\mathbb{N}}$ of all possible Δ^0_2 indices. That is, every function d_e in the list is a total computable function of two variables, and if X is a Δ^0_2 set, then there is some e, such that $\forall i(\lim_{s\to\infty} d_e(i,s) = X(i))$. We fix an enumeration $\langle d_e, \delta_e, W_e\rangle$ of all triples such that δ_e is a partial computable function (of a single variable), and W_e is a c.e. set. We let $J^X(i)$ denote the universal partial X-computable function. We also assume the convention that the use of any convergent computation is positive. We construct a coinfinite c.e. set A satisfying the following requirements :

$\mathcal{P}_e$: If W_e is infinite, then $A \neq \overline{W}_e$,

$\mathcal{N}_e$: If δ_e is total, and d_e approximates W'_e with at most δ_e many mind changes, then $W_e \oplus A$ is superlow.

To satisfy $\mathcal{N}_e$, we will construct a function p_α at each node α assigned to the requirement $\mathcal{N}_e$. To make $W_e \oplus A$ superlow, we will try to ensure that for all i,

$$\lim_{s\to\infty} p_\alpha(i,s) = \begin{cases} 1 & \text{if } J^{W_e\oplus A}(i)\downarrow, \\ 0 & \text{if } J^{W_e\oplus A}(i)\uparrow, \end{cases}$$

with at most $\varphi_\alpha(i)$ many mind changes (where $\varphi_\alpha(i)$ is a computable function to be determined). In this case, we say that $p_\alpha(i,s)$ *is correct in predicting* $J^{W_e\oplus A}(i)$. Since d_e and p_α are approximations of sets which are themselves jumps of other sets, it is convenient to assume that these functions are binary-valued, with values from the set $\{\uparrow,\downarrow\}$. Thus we say $p_\alpha(i,s) =\uparrow$ if it has value 0, and $p_\alpha(i,s) =\downarrow$ otherwise. For a node α assigned to the requirement $\mathcal{N}_e$, there are infinitely many subrequirements

$\mathcal{N}_{e,i}$: If δ_e is total, and d_e approximates W_e' with at most δ_e many mind changes, then $p_\alpha(i,s)$ is correct in predicting $J^{W_e\oplus A}(i)$ with at most $\varphi_\alpha(i)$ many mind changes.

11.1.1 The Basic Strategy

Our construction is a modification of the construction of an almost deep degree in [CGS01]. We describe the basic strategy for $\mathcal{N}_{e,i}$. We stay as close as possible to the almost deep case, and at the end we will outline the modifications required to make our construction work. We will enumerate the Turing functional Θ as a test to see if d_e approximates W_e' correctly. The basic strategy consists of the following steps:

(1) Wait for $J^{W_e\oplus A}(i)[s]\downarrow$. Enumerate a computation $\Theta^{W_e}(n)[s]\downarrow$ with use $u =$ W_e-use of $J^{W_e\oplus A}(i)[s]\downarrow$. Restrain A. Go to (2).

(2) Wait for either $d_e(n,s) =\downarrow$, or W_e to change below u. If W_e changes below u, drop the restraint on A and go back to (1). Otherwise if $d_e(n,s) =\downarrow$, set $p(i,s) =\downarrow$ and go to (3).

(3) Wait for a $W_e\restriction u$ change. Drop the restraint on A, and go to (4).

(4) Wait for $d_e(n,s) =\uparrow$. If the number of mind changes in $d_e(n,-)$ so far is less than $\delta_e(n)$, set $p(i,s) =\uparrow$. Go back to (1) and repeat.

If we wait forever in (1), then $p(i)$ will correctly predict the divergence of $J^{W_e \oplus A}(i)$. If we wait forever in (2) or (4), then d_e will not approximate W_e' correctly since n was given by the Recursion Theorem. If the strategy cycles through (1) and (2) infinitely often, then $p(i)$ will correctly predict the divergence of $J^{W_e \oplus A}(i)$ (remember that up till this point we had not yet set $p(i,s) =\downarrow$). If we wait forever in (3) then we have correctly predicted the convergence of $J^{W_e \oplus A}(i)$ (we are still restraining A in state (3)). Note that it is not possible to cycle through all 4 stages infinitely often.

11.1.2 Coordination with the mother node

The strategies are arranged on a Σ_2/Π_2-guessing tree. The main strategy $\mathcal{N}_e$ is given a place τ on the tree. Its subrequirements are assigned to locations extending τ. There are different nodes ($\alpha_0^i, \alpha_1^i, \cdots$) assigned to the subrequirement $\mathcal{N}_{e,i}$ on the same level, and each α_k^i is making a guess as to whether the stronger requirements have a Σ_2 or a Π_2 outcome. Each α_k^i follows the same basic strategy as previously outlined, with the mother strategy $\mathcal{N}_e$ at node τ coordinating their actions.

In the construction of an almost deep degree, the strategy used by each α_k^i is divided into two phases. During the first phase, control is given to τ which takes over the execution of steps (1) and (2). During the second phase, control is transferred from τ to α_k^i, where the remaining steps (3) and (4) will be executed. α_k^i will communicate with τ by "connecting" with it. The purpose of forming connections is for α_k^i to tell τ that it has already dropped its A-restraint once, and is now ready to receive fresh restraint. The action of a node τ is the following:

Phase 1 (action by τ) :

(1) Wait for $J^{W_e \oplus A}(i)[s] \downarrow$, and at least one α_k^i is connected to it. Enumerate a computation $\Theta^{W_e}(n(\alpha_k^i))[s] \downarrow$ for every connected α_k^i. Restrain A with priority τ.

(2) Wait for either W_e to change, or $d_e(n(\alpha_k^i)) =\downarrow$ for all connected α_k^i. If W_e has changed, drop the restraint on A and go back to (1). Otherwise if $d_e(n(\alpha_k^i), s) =\downarrow$ for all connected α_k^i, set $p_\tau(i,s) =\downarrow$, and transfer control of the strategy to all connected α_k^i by doing the following : for each connected α_k^i, we disconnect it and start α_k^i in step (3). Drop the τ-restraint on A, and transfer the restraint

to α^i_k.

Finally, we make τ go back to step (1). Note that at this point in time, τ holds no restraint and is not connected to any version of $\mathcal{N}_{e,i}$. Its role in the $\mathcal{N}_{e,i}$-strategy is now temporarily over, and will now wait in step (1) for some α^i_k to connect to it again.

Phase 2 (action by α^i_k) :

(0) Connect to τ to tell it to take over, and wait for τ to finish steps (1) and (2). Meanwhile, α^i_k imposes no restraint of its own.

(3) Hold the transferred A-restraint with priority α^i_k. Wait for W_e to change. Drop the restraint on A, and go to (4).

(4) Wait for $d_e(n(\alpha^i_k), s) =\uparrow$. If the number of mind changes made by $d_e(n(\alpha^i_k))$ so far is less than $\delta_e(n)$, set $p_\tau(i, s) =\uparrow$, and back go to (0).

So far the above description was based on the construction of an almost deep degree. An important new feature in this construction is that we need to get a computable upper bound φ_τ for the number of mind changes for p_τ. To do this, τ needs to determine in advance, a bound on the number of times each node $\alpha^i_0, \alpha^i_1, \cdots$ is going to be initialized. We could then prepare beforehand, a finite list of indices I_i from which α^i_k could use as testing locations $n(\alpha^i_k)$. We can then set $\varphi_\tau(i) = \delta_e(\max I_i)$.

In order to do this, observe that in our construction, τ will only transfer restraint to α^i_k finitely often; in between any two such transfers, we must have a W_e-change and new mind changes of $d_e(n(\alpha^i_k), s)$. Therefore whenever τ transfers restraint to α^i_k, we will also transfer the restraint *to every node on level* $|\alpha^i_k|$ which extends τ. The only reason a node α^i_k needs to abandon the current choice of testing location $n(\alpha^i_k)$, is due to an A-change after $\Theta^{W_e}(n(\alpha^i_k))$ has been defined. If this happens in phase 1, then the same A-change will also initialize τ, since the restraint is currently held at τ. If this change is in phase 2, then it has to be due to a positive node extending τ of length $< |\alpha^i_k|$. Hence τ can determine I_i in advance. The entire level $|\alpha^i_k|$ can only have increased restraint finitely many times, so the positive nodes of longer length will eventually not be blocked from acting. Note that we cannot do this (i.e. transfer restraint to the entire level) if we were trying to make A almost deep; this is because if the $\mathcal{N}$-hypothesis was false and d_e was not the correct lowness index for W_e, then α_k^i might hold restraint under step (3) with infinite lim sup.

Since each α_k^i will have only finite activity during the construction, we may even simplify the tree design further by removing the subnodes α_k^i from the tree. We will require τ to have two outcomes, ∞ and f. Each time τ is visited we will run steps (1) and (2) for each $i < s$ such that $J^{A\oplus W_e}(i)\downarrow$. We wait for *all* of these modules to respond with either a W_e-change, or a switch in $d_e(n_i)$. If this never happens then τ wins (by failure of the $\mathcal{N}$-hypothesis) with a finite A-restraint. When all the modules have responded we then allow $\tau^\frown\infty$ to be accessed, and distribute the restraints down appropriately. We may arrange for the latter by having, say different restraint functions $r_\tau(i)$ for the i^{th} level below τ. The construction and verification are routine, and we omit them.

Question 11.1.4. *Is every promptly simple almost superdeep degree also strongly jump traceable?*

We ask a more general question: Is there a direct construction of a promptly simple almost superdeep degree? Diamondstone [Dia] constructed directly a promptly simple c.e. set which was not superlow cuppable. We draw attention to the strategy employed in [Dia] for making the set not superlow cuppable (there we also had different restraint functions to be obeyed by an entire level). One might adapt the strategy there to make the constructed set almost superdeep.

11.2 Strong cupping

We extend the result of Bickford and Mills [BM] that no superlow set is wtt-cuppable. We show that:

Theorem 11.2.1. *No totally ω-c.e. set is wtt-cuppable. That is, if A and D are c.e. sets such that $\emptyset' \leq_{wtt} A \oplus D$ and A is totally ω-c.e., then $\emptyset' \leq_{wtt} D$.*

Proof. Suppose A, D and Φ are given such that $\emptyset' = \Phi^{A\oplus D}$ and the use φ is computable. We first describe the strategy required if we assumed instead that A was superlow. We would want to compute $\emptyset'$ from D; let us consider only $\emptyset'(0)$. We would associate 0 with some $g(0)$ for some (partial) function $g \leq_T A$ which we get to build. Since A is superlow, we can request for certification that A is correct on the current use of $g(0)$; furthermore we have a computable bound on the number of wrong certifications which may be given, say with bound h. Therefore we could prepare in advance $h(0)$ many agitators $c_1, \cdots, c_{h(0)}$ targeted for $\emptyset'$ (more specifically we are building a c.e. set C which is hardwired into $\emptyset'$), and then compute $\emptyset'(0)$ from the first $\gamma(c_{h(0)})$ many bits of D. Namely, if 0 does enter $\emptyset'$ we want to force $D\restriction\gamma(c_{h(0)})$ to change. We do this by enumerating $c_1, c_2, \cdots, c_{h(0)}$ into $\emptyset'$ one at a time. At each single attack, if $D\restriction\gamma(c_{h(0)})$ changes then we are done; otherwise $A\restriction\gamma(c_{h(0)})$ has to change, and we can get to redefine a new value for $g(0)$ at every attack we make. Since the number of wrong certifications for $g(0)$ is at most $h(0)$, one of the attacks must result in a D change, and we are done.

In the above argument, note that we can obtain the bound h effectively from g. Since g obviously can be made total, the above argument works even when A is only array computable. If A is now totally ω-c.e., we no longer can obtain the bound h effectively from g. To overcome this, we will have to "guess" at the correct bound, and build infinitely many reductions attempting to witness $\emptyset' \leq_{wtt} D$, based on each guess.

Formal proof. We may assume that φ is actually the number of bits of A (or the number of bits of D, if this is larger) which are accessed during the computation. By the recursion theorem, we can build some c.e. set C as part of $\emptyset'$ which we control, and assume that $C = \Phi^{A\oplus D}$. We fix an effective list $\langle f_e, h_e\rangle$ of partial computable functions such that f_e is binary and h_e is unary.

We perform infinitely many constructions, and argue that one of them works. The e^{th} construction will produce (uniformly in e) a total function $g_e \leq_T A$, and partial computable function δ_e. The aim of the e^{th} construction is to satisfy the single requirement

$\mathcal{R}_e$: If $\lim f_e(\langle e, x\rangle, s) = g_e(x)$ for every x with at most $h_e(\langle e, x\rangle)$ many mind changes, then ensure $\emptyset' \leq_{wtt} D$ with computable use δ_e.

Construction. We now describe the e^{th} construction; for convenience we drop e from the notations. Since $C = \Phi^{A\oplus D}$, we will only act whenever $C = \Phi^{A\oplus D}$ looks correct on a sufficiently long initial segment. If we take any action in the construction that kills a current computation, we always wait until the relevant computations recover. We also assume that new Φ-computations converge instantly when required.

The parameters that are needed are the following: A list of agitators $c_{k,0}, c_{k,1}, \cdots$ targeted for entry into C, and we single $c_{k,0}$ out as the *leading k-agitator* for each k. We may assume that we fix the leading agitators $c_{k,0} = 2k$ for all k; the rest of the agitators will be picked during the construction from the odd integers. We also build a partial computable function δ, which will be total if the $\mathcal{R}$-hypothesis is correct. At stage $s = 0$, set everything to be undefined. At $s > 0$, we look for the least k such that one of the following holds.

(A1) $\delta(k)[s]\uparrow$ and $h(k)[s]\downarrow$. In this case pick fresh agitators $c_{k,1} < \cdots < c_{k,h(k)}$. Set $\delta(k) = \varphi(c_{k,h(k)})$.

(A2) *Correction needed*: there is some k such that $\delta(k)\downarrow$, k has entered $\emptyset'$ at some stage $t < s$ and $D\restriction\delta(k)[t] = D\restriction\delta(k)[s]$. We begin the sequence of attacks as follow (to force $D\restriction\delta(k)$ to change): the first attack starts by enumerating $c_{k,0}$ into C, and waits for either $A\restriction\varphi(c_{k,0})$ or $D\restriction\varphi(c_{k,0})$ to change. If D changes then the attack is successful, otherwise if A changes, we wait for $f(x)$ to switch to output the current value of $A\restriction\varphi(c_{k,h(k)})$ (which is different, since A has changed). We begin the second attack after $f(x)$ changes, and repeat with $c_{k,1}$. We stop whenever one of the attacks is successful.

If no such k exists, we do nothing.

Verification. We first of all verify that the e^{th} construction works. Let $\tilde{h}(k) := h_e(\langle e,k\rangle)$. Define $g_e(k)$ by the following. Go to the first stage s such that $A\restriction\varphi(c_{k,0})[s] = A\restriction\varphi(c_{k,0})$. If $\tilde{h}(k)[s]$ has not yet converged, then we output $A\restriction\varphi(c_{k,0})$. Otherwise output $A\restriction\varphi(c_{k,\tilde{h}(k)})$. Note that g_e is total computable in A, regardless of what happens in the construction. Suppose the hypothesis in $\mathcal{R}_e$ holds. First of all, we argue that for each k, whenever correction is needed under (A2) for k, one of the attacks must be successful. Suppose all of the $\tilde{h}(k)+1$ many attacks fail. We must have $g_e(k) = A\restriction\varphi(c_{k,\tilde{h}(k)})$, since the first attack fails. Each time a new attack fails, $f_e(\langle e,k\rangle)$ has to switch to give a new string, since f_e predicts $g_e(k)$ correctly and A is c.e. But $f_e(\langle e,k\rangle)$ will have to switch $\tilde{h}(k)$ many times, a contradiction.

It follows therefore that each k receives attention at most twice, and hence δ_e is total. To show that $\emptyset' \leq_{wtt} D$ with use δ_e, we fix k and let s be the first stage where $D \upharpoonright \delta_e(k)[s] = D \upharpoonright \delta_e(k)$. Clearly $k \in \emptyset' \Leftrightarrow k \in \emptyset'[s]$, because if k enters $\emptyset'$ after s, then correction will be needed for k, in which $D \upharpoonright \delta(k)$ will be forced to change after stage s.

Finally, we verify that the theorem holds. Let $g(\langle e, x\rangle) := g_e(x)$, which is total computable since g_e is generated effectively. There will be some e such that the pair f_e, h_e witnesses that g is ω-c.e., and hence satisfies the hypothesis in $\mathcal{R}_e$. As a final note, we remark that even though each $g_e \leq_{wtt} A$, but the amalgamation g is merely computable in A, as we might well expect. $\square$

Question 11.2.2. *Can a superlow c.e. set be tt-cuppable? That is, is there a superlow c.e. set A and a (c.e.) set $D \not\geq_{tt} \emptyset'$ such that $\emptyset' \leq_{tt} A \oplus D$?*

Chapter 12

On cupping and the jump hierarchy

The work in this chapter is joint with Noam Greenberg and Guohua Wu. Every set mentioned in this chapter is computably enumerable unless otherwise stated.

12.1 Introduction

Two of the most influential concepts in the study of computably enumerable sets are that of lowness and prompt simplicity. As we have seen, lowness is concerned with the intrinsic information content of a set (or rather, the lack thereof). Prompt simplicity was introduced by Maass [Maa83] in connection with automorphisms of the lattice of the c.e. sets. This is a dynamic property which describes how fast elements may be enumerated into the set A. A promptly simple set A is in some sense similar to $\emptyset'$ in its dynamic properties. Ambos-Spies, Jockusch, Shore and Soare [ASJSS84] proved the beautiful result which linked the dynamic property of a set with a degree theoretic property: A is promptly simple iff A can be cupped with a low set.

Several recent results have recast the spotlight on this class, and variations on cuppability have been examined. Li, Wu and Zhang [LWZ00] defined a hierarchy of cuppable sets $LC \subseteq LC_2 \subseteq LC_3 \subseteq \cdots$, where $LC_n = \{A : \exists \text{ low}_n \text{ set } L \text{ such that } A \oplus L \equiv_T \emptyset'\}$, and $LC = LC_1$. They called these sets *low$_n$-cuppable*. They also showed that $LC_2 \neq LC$ by constructing a low$_2$-cuppable set which was not promptly simple. It is open if the low$_n$-hierarchy collapses:

Question 12.1.1. *Is each level of the low$_n$-cupping hierarchy distinct from the next?*

A. Li has claimed the above to be true. In fact, he conjectured that every cuppable set is in LC_n for some fixed n, but no proof of either statement has appeared. A different question one could consider is:

Question 12.1.2. *Does $\cup_n LC_n =$ cuppable sets?*

In Section 12.2 we answer this question (as well as A. Li's conjecture) negatively by constructing a cuppable set which can only be cupped with high sets; certainly such a set is not low$_n$-cuppable for any n.

The result of Ambos-Spies, Jockusch, Shore and Soare demonstrated a certain robustness in the class of low-cuppable sets. Variations of this class have been studied, with the most notable ones being the superlow-cuppable sets (SLC), and the sets which can be cupped with an array computable set (AC-cuppable). A set A is superlow-cuppable if there is a superlow set L such that $A \oplus L \equiv_T \emptyset'$. Nies asked if SLC and LC were equal, and this was answered by Diamondstone [Dia] who constructed a promptly simple set which was not in SLC. In [DN], Diamondstone and Ng explored the dynamic properties of a superlow-cuppable set. They showed that a natural strengthening of prompt simplicity, a notion which they called *strongly prompt*, was sufficient for being superlow-cuppable. It is possible that there are characterizations of LC_2 (or even the higher levels of the hierarchy) in this direction:

Question 12.1.3. *Are there characterizations of LC_n in terms of dynamic properties (such as for LC)?*

Recently Downey, Greenberg, Miller and Weber [DGMW06] have investigated the class of AC-cuppable sets. In particular, they prove that

Theorem 12.1.4 ([DGMW06]). *If A is promptly simple, then there is an array computable set C which cups with A.*

They also showed that there was an AC-cuppable set which was not promptly simple. Since their proof of Theorem 12.1.4 used a priority tree, it was not immediately obvious that the constructed set C was low as well. They asked if every promptly simple set had a low and array computable cupping partner. In Section 12.3 we answer their question and show that this is the case.

Finally in Section 12.4 we examine the class AC-cuppable. We show that AC-cuppable is exactly the same as LC_2; as a corollary we get Theorem 12.1.4. A superlow set A can be viewed as a low set with a computable bound (on the number of mind changes witnessing the lowness of A). This analogy can be extended to compare an array computable set with a low$_2$ set in the same way; indeed array noncomputable sets share many of the properties of a nonlow$_2$ set with respect to lattice embedding. Our result in Section 12.4 says that the expected analogous result of Diamondstone does not hold.

Finally we remark that our results from Sections 12.3 and 12.4 show that the cupping classes low+AC-cuppable and AC-cuppable coincide with LC and LC_2 respectively and we in fact get nothing new:

$$SLC \Rightarrow LC = \text{low} + AC\text{-cuppable} \Rightarrow LC_2 = AC\text{-cuppable}.$$

12.2 A cuppable set which cups with only high sets.

In this section we prove the following.

Theorem 12.2.1. *There is a cuppable set A such that for any set W, if $A \oplus W \geq_T \emptyset'$, then W is high.*

We build a set C and a functional Γ and ensure that $\emptyset' = \Gamma^{A\oplus C}$; this is a global action and it has higher priority over all other actions. We also have to satisfy the following requirements

$$\begin{aligned}\mathcal{M}_e &: \ \emptyset' \neq \Phi_e^C \\ \mathcal{R}_e &: \ \text{If } \emptyset' = \Phi_e^{W_e \oplus A}, \text{ then } W_e \text{ is high.}\end{aligned}$$

Here $\langle \Phi_e, W_e \rangle$ is the e^{th} pair such that Φ_e is a Turing functional and W_e is a c.e. set. These requirements automatically ensure that A is neither computable nor complete.

The reduction Γ is built with the help of the moving markers $\gamma(x)$, which denote the current use of $\Gamma^{A\oplus C}(x)$. The markers $\gamma(x)$ have to obey the usual marker rules, which are:

1. if $\Gamma^{A\oplus C}(x)[s]\downarrow$ then $\gamma(x)[s]\notin A\cup C[s]$,
2. for any $x<y$ such that $\gamma(y)[s]\downarrow$, we also have $\gamma(x)[s]\downarrow$ and $\gamma(x)[s]<\gamma(y)[s]$,
3. any new definition for $\gamma(x)$ is fresh,
4. $\Gamma^{A\oplus C}(x)[s]$ is undefined if a number $y\leq\gamma(x)[s]$ is enumerated into $A\cup C$.

The global coding strategy has to do two things: first it has to ensure the *totality of* Γ, and does this by defining $\Gamma^{A\oplus C}(x)\downarrow=\emptyset'(x)[s]$ at each stage s, for the smallest x which is currently undefined. Second, it has to ensure the *correctness of* Γ. Whenever $\Gamma^{A\oplus C}(x)\downarrow\neq\emptyset'(x)$, it will enumerate $\gamma(x)$ into C to allow $\Gamma^{A\oplus C}(x)$ to be corrected. Note that the global coding strategy will never directly enumerate any number into A.

12.2.1 A single $\mathcal{M}$-strategy

The strategy $\mathcal{M}$ will be a variant of the standard Friedberg strategy. That is, we pick a fresh follower x and wait for $\Phi^C(x)\downarrow=0$. We then enumerate x into $\emptyset'$ (what we really mean is that we enumerate some number x' into E where E is a c.e. set we build; this change will be reflected by a change in $\emptyset'(x)$), and attempt to preserve C. However as the global coding has priority over $\mathcal{M}$, we might destroy a computation $\mathcal{M}$ wants to preserve due to the need to correct $\Gamma^{A\oplus C}(n)$ for some n. To limit the number of times each $\mathcal{M}$ is injured in this way, we associate a number k with each $\mathcal{M}$, called the *threshold of* $\mathcal{M}$. When $\mathcal{M}$ observes that $\Phi^C(x)\downarrow=0$, and before it enumerates x into $\emptyset'$, $\mathcal{M}$ will first enumerate $\gamma(k)$ into A. The purpose of doing this, is to disengage all the markers $\gamma(k),\gamma(k+1),\cdots$ from below the C-use of $\Phi^C(x)$; that is, after enumerating $\gamma(k)$ into A we may now redefine $\gamma(k),\gamma(k+1),\cdots$ to values much larger than $\varphi^C(x)$. Only after we have successfully disengaged $\gamma(k)$, do we allow $\mathcal{M}$ to put x into $\emptyset'$. Hence the only way for this $\mathcal{M}$-diagonalization attempt to become undone, is for one of $\gamma(0),\cdots,\gamma(k-1)$ to enter C; this only happens due to a number less than k entering $\emptyset'$ (rendering the $\Gamma^{A\oplus C}$-computation incorrect); this happens at most k times provided the threshold k remains fixed. We point out that when considering the interaction of $\mathcal{M}$ with other strategies, there are situations that might arise during the construction that will cause the threshold of $\mathcal{M}$ to change.

12.2.2 A single $\mathcal{R}$-strategy

Before describing this strategy, we would like to point out that this will be a variation on the noncuppable strategy. In a noncuppable strategy we had to make W Turing complete; here we are allowed more freedom in the sense that we only need to make W high. We will see why this fact is important.

In the construction $\mathcal{R}$ will be assigned to some mother node τ, which will measure the length of agreement of $\emptyset' = \Phi^{W \oplus A}$. τ also builds some approximation $f_\tau^W(x, s)$ attempting to witness the highness of W. At levels below τ, we have daughter nodes $\sigma = \sigma(x)$ wishing to make $\lim_s f_\tau^W(x, s) = Tot(x)$; σ guesses at whether $x \in Tot$, and according to current information will decide on outcome f or ∞. At σ-expansionary stages (where x looks like it is in Tot), σ will attempt to change W (as in a usual highness strategy) to redefine $f_\tau^W(x, -) = 1$ to reflect its current desire to have the limit at 1. At a non-expansionary stage σ simply defines more of $f_\tau^W(x, -) = 0$, with some use $u_\tau(x)$ on W, which it can later change at expansionary stages. This way τ ensures that $\lim_s f_\tau^W(x, s)$ exists, and will have value 1 iff $x \in Tot$.

Of course we do not directly control W, and we can only agitate changes in W via Φ. To do this, before $\sigma(x)$ defines more of $f_\tau^W(x, -)$, it first picks a number z which we call an *agitator for* σ. It keeps z out of $\emptyset'$ while waiting for $\Phi^{A \oplus W}(z) \downarrow = 0$. σ then defines more of $f_\tau^W(x, -)$ with W-use $u_\tau(x) > \varphi^{W \oplus A}(z)$. If σ needs to change W to adjust $f_\tau^W(x, -)$, it will enumerate z into $\emptyset'$ and wait for the resulting W-change (no small numbers are permitted to enter A in the meantime).

12.2.3 Interaction between strategies

So far we had not yet explained how the fact that we only need W to be high would help. This comes when considering the interaction between a node α working for $\mathcal{M}$, and a node $\sigma(x)$ working for the mother node τ. Suppose α was between τ and σ.

For the following discussion, the reader should think of $\mathcal{M}$ as being a general positive requirement (such as making A noncomputable) which wants to put numbers into A. The reader should also temporarily think of $\mathcal{R}$ as being a noncuppable strategy requiring W to be Turing complete.

α would at some point need to enumerate a number p into A; it might turn out that when α is ready to do so, we have $p < \varphi^{W \oplus A}(z)$ where z is the agitator for σ. If α enumerated p into A without doing anything else, it would cause $\varphi^{W \oplus A}(z)$ to be lifted above $u_\tau(x)$, and when σ next wants to change W, its agitator z would now be ineffective because the entry of z into $\emptyset'$ will cause W to change below $\varphi^{W \oplus A}(z)$ but possibly above $u_\tau(x)$. If τ and σ were trying to make W Turing complete (instead of merely high), then this damage is fatal to the actions of τ. The reader will recall that in a noncuppable construction, this problem was overcome by implementing *delayed enumeration by* α. Before enumerating p into A, α would first enumerate its own agitator number into $\emptyset'$ and wait for W to change; when that happens σ would also have got its use $u_\tau(x)$ undefined, and we would immediately put p into A before redefining a new use $u_\tau(x)$ for σ.

Why could we not adopt this strategy for our construction, and end up with a set A which is both cuppable, and noncuppable? The key point is that the above delayed enumeration can only be used in constructions where the positive requirements on A could tolerate such delays; for instance if we needed to make A noncomputable, or even high. On the other extreme it would be impossible to make A promptly simple as the positive requirements on A would require immediate attention. In this construction the $\mathcal{M}$-requirements now seek to make A cuppable; if we did the above, then when α (working for an $\mathcal{M}$-strategy) finally gets a permission to change A and disengage certain γ-markers, it might not be able to do so. This is because in between the two stages s_1 (when α has requested for permission from τ), and $s_2 > s_1$ (when τ finally gave permission), C might have changed due to the global coding so that $\varphi^C(x_\alpha)$ is now undefined. Therefore in our construction, we will have to allow α to change A immediately at s_1; by doing so α would render all the current agitators of every σ of lower α-priority ineffective, and they must now pick new agitators. This is fine because α only changes A finitely often (unless α itself is initialized), and so each σ of lower α-priority will be injured this way finitely often. Note that since we only needed to make W high, this is perfectly fine.

We next describe what happens if σ is of higher α-priority, that is $\tau \subset \alpha$ and σ is left of α or is extended by α. If $\alpha \supseteq \sigma^\frown\infty$ then α only acts at stages when σ has just observed a change in W; σ would not care about the A-changes which α might make. The more interesting case is otherwise. Since α is not allowed to injure σ, α has to pick a threshold k_α such that $\gamma(k_\alpha) > \varphi^{A\oplus W}(z_\sigma)$ where z_σ is the current agitator of σ. If W changes below one of these φ-uses, then α will have to pick a fresh threshold k_α to deal with the increased $\varphi^{A\oplus W}(z_\sigma)$-use. If α is on the true path, then the set of agitators z_σ it has to respect is going to be fixed eventually; hence the only way that α can be injured infinitely often this way is for one of the finitely many φ-uses to go to infinity. Thus in this case even though α is on the true path, its strategy will not be met as its threshold k_α will be driven to infinity; however this corresponds to a global τ-win, and we can arrange for α to have an outcome drawing attention to this fact; one could then restart the other requirements (including α itself) below this outcome.

This concludes the discussion. The details and full construction will appear in the paper currently in preparation.

12.3 Every promptly simple set has a low cupping partner below every cupping partner.

If A is promptly simple, then we know that there is a low cupping partner for A. In which Turing lower cones can we find a low cupping partner for A? Clearly if C bounds a low cupping partner for A, then C itself also cups with A. We show that this is exactly the condition needed; that is a c.e. set C computes (in fact, wtt-computes) a low cupping partner of A if and only if C itself cups with A.

Theorem 12.3.1. *Suppose A is promptly simple, and C is such that $\emptyset' \leq_T A \oplus C$. Then, there is a low set $B \leq_{wtt} C$ such that $\emptyset' \leq_T A \oplus B$.*

As an immediate corollary, we have that every promptly simple set can be cupped with a low array computable set.

Corollary 12.3.2. *If A is promptly simple, then there is a low array computable set B such that $\emptyset' \leq_T A \oplus B$.*

12.3.1 Requirements and notations

We are given a promptly simple set A, and a set C and a Turing functional Γ such that $\emptyset' = \Gamma^{A\oplus C}$. Our job is to build a low c.e. set B and ensure that $B \leq_{wtt} C$. We also need to build a Turing functional Φ and ensure that $\emptyset' = \Phi^{A\oplus B}$. To ensure the lowness of B, we will try and preserve the computation $J^B(e)$ each time it converges, where $J^B(e)$ is the universal partial B-computable function. We suppress all mention of the stage number from the parameters if the context is clear, and at other times we will append $[s]$ to an expression to denote the value of the expression at stage s. The use of the functionals Γ, J and Φ are denoted respectively by γ, j and φ. Since we get to build the functional Φ, we think of $\varphi(e)$ as a marker pointing at some number $x \notin A_s \oplus B_s$. Since there are two parts to the oracle $A \oplus B$, we will allow $\varphi(e)$ to move if either A or B changes below x (one can think of the actual Φ-use as being $2\varphi(e) + 1$). We build Φ as a c.e. set of axioms, in the usual way. When we say that we pick a fresh number x, we mean that we pick x to be the least number $x > s$ and $x >$ any number used or mentioned so far.

We make use of the prompt simplicity of A in the standard way. At each e, we will enumerate an auxiliary c.e. set U_e to try and force A to change. By a slowdown lemma and the Recursion Theorem, we may assume that if we put numbers $x_0 < x_1 < \cdots$ into an auxiliary set U at stages $s_0 < s_1 < \cdots$ respectively, then A has to promptly permit one of them; namely there is some i such that $A_{s_i}\upharpoonright x_i \neq A_{s_i+1}\upharpoonright x_i$. We may in fact assume that A has to promptly permit infinitely many of the x_i's.

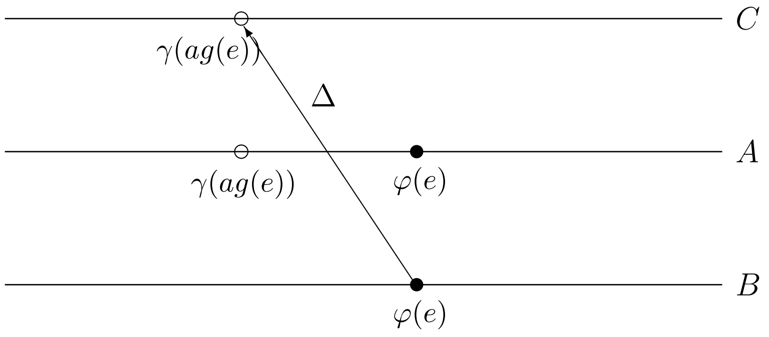
By the Recursion Theorem again, we have an infinite list of numbers $\{i_0, i_1, \cdots\}$ where we are able to control $\emptyset'(i_k)$ for all k. That is, we get to decide whether or not to enumerate i_k into $\emptyset'$. These numbers are called *agitators*. For each e, in order for us to set the uses of the computation $\Phi^{A\oplus B}(e)$ correctly, we need to pick an agitator from the list. We denote this appointed agitator by $ag(e)$. If we use up the appointed agitator (i.e. we enumerate it into $\emptyset'$ to force a change in $A \oplus C$), we will then appoint a fresh one for $ag(e)$, since the old one can no longer be used.

As mentioned previously we think of Φ as a c.e. set of axioms of the form

$\langle x, y, \sigma \oplus \tau \rangle$ where x represents the input, y represents the output and σ, τ represents the A and B use respectively. During the construction we will occasionally set $\Phi^{A\oplus B}(e)[s] \downarrow = r$ with use u. What this means is that we enumerate the axioms $\langle e, r, \sigma \oplus B_s\!\restriction u \rangle$ for every σ of length u, such that $\sigma \supset A_s\!\restriction\gamma(ag(e))$. Thus this axiom remains applicable until either $B\!\restriction u$ or $A\!\restriction\gamma(ag(e))$ changes. If $s < t$ are two stages such that $\Gamma^{A\oplus C}(x)[s] \downarrow$, then we say that the computation $\Gamma^{A\oplus C}(x)$ *is stable from s to t*, if $A\!\restriction\gamma(x)[s] = A\!\restriction\gamma(x)[t]$ and $C\!\restriction\gamma(x)[s] = C\!\restriction\gamma(x)[t]$ holds.

12.3.2 Description of strategy

We describe the plan here briefly. We want to build a wtt reduction $B = \Delta^C$ (Δ is not built explicitly in the actual construction; it is used here solely for illustrative purposes). Every marker $\varphi(e)$ is associated with an agitator number $ag(e)$, and we always ensure that we keep $\varphi(e) > \gamma(ag(e))$, so that in the event of any A-change, the $\varphi(e)$-marker is lifted and we may benefit from it. Once the appropriate $\varphi(e)$-uses are set we will define $B(\varphi(e)) = \Delta^C(\varphi(e))$ with C-use $\gamma(ag(e))$. The following is a helpful illustration of the situation.



There are two things to take care of in this construction: coding and lowness of B. If e ever enters $\emptyset'$ and coding needs to be done, we can simply enumerate the agitator $ag(e)$ into $\emptyset'$ (or at least, the part of $\emptyset'$ which we control). The opponent will either respond with an A-change (and coding is automatically done for us) or he responds with a C-change (in which case we are now allowed to enumerate $\varphi(e)$ into B for the sake of coding). Note we do not require the prompt simplicity of A to carry out this step.

The second thing we need to do is to ensure that $\exists^\infty s(J^B(e)[s] \downarrow) \Rightarrow J^B(e) \downarrow$ for every e. To this end we will ensure that each time $J^B(e)[s]$ converges with use

$j(e)[s]$, we will try and lift $\varphi(e)$ above $j(e)[s]$ by causing an A-change below $\varphi(e)[s]$. We do so by requesting a prompt change in $A \upharpoonright \gamma(ag(e))$. If the prompt change is given then the marker $\varphi(e)$ would have been lifted above $j(e)$ successfully. It is *only when prompt permission is denied*, that we enumerate the agitator $ag(e)$ into $\emptyset'$. The result would be either an A-change (in which case $\varphi(e)$ is lifted successfully) or a C-change. If the latter happens we will need to enumerate $\varphi(e)$ into B to set the marker $\varphi(e)$ above $\gamma(z)$ for a new agitator z. This destroys the computation $J^B(e)$, but in turn ensures that the auxiliary set U_e increases in size each time we request for prompt permission. Since prompt permission will eventually be given, it follows that $\varphi(e)$ will eventually be lifted above $j(e)$, which means that $J^B(e)$ will eventually be preserved forever provided no smaller φ-marker moves.

Note that the prompt simplicity of A is crucial in ensuring that the coding of $\emptyset' \leq_T A \oplus B$ is compatible with the lowness of B (as we might expect is the case). This is because if we rely solely on agitators to try and lift the marker $\varphi(e)$ above $j(e)$, the opponent can simply respond with a C-change every time we use up an agitator (this can be the case if C is merely incomplete). Also it is impossible to make B superlow because even after $\varphi(e)$ is lifted above $j(e)$ successfully, smaller $\varphi(e')$-markers may still move (unpredictably) due to $j(e')$.

12.3.3 The construction

At stage $s = 0$ we make all parameters undefined. Since $\Gamma^{A\oplus C} = \emptyset'$, at the end of each stage s, we may wait until $\Gamma^{A\oplus C}(z) \downarrow = \emptyset'(z)$ for every $z \leq$ the largest number used so far, before starting the next stage $s+1$. At stage $s > 0$ we pick the least $e < s$ that *requires attention*, i.e. one of the following holds:

(A1) $ag(e)$ undefined,

(A2) no axioms in $\Phi^{A\oplus B}(e)$ currently applies,

(A3) the computation $\Gamma^{A\oplus C}(ag(e)[s^-])$ is not stable from s^- to s, where $s^- < s$ is the stage where the current axiom in $\Phi^{A\oplus B}(e)$ was set,

(A4) $\Phi^{A\oplus B}(e) \downarrow \neq \emptyset'(e)$,

(A5) $J^B(e) \downarrow$ with use $j(e) \geq \varphi(e)$.

We then act for e; the action to be taken depends on the first item in the list above that applies to e. We perform the required action, and end the stage.

- If (A1) applies, pick a fresh agitator for $ag(e)$. If (A2) applies, we set $\Phi^{A\oplus B}(e)[s]\downarrow= \emptyset'(e)$ with fresh use $\varphi(e)$. If (A3) applies then enumerate $\varphi(e)$ into B to make the $\Phi^{A\oplus B}(e)$-computation not applicable.

- If (A4) applies then enumerate $ag(e)$ into $\emptyset'$ and pick a fresh $ag(e)$. Wait for $C\upharpoonright\gamma(ag(e))$ or $A\upharpoonright\gamma(ag(e))$ to change. If the latter happens first, do nothing else. If the former happens first, we enumerate $\varphi(e)$ into B.

- Finally if (A5) applies we enumerate $\gamma(ag(e))$ into U_e and request prompt permission. If we are given permission (i.e. $A\upharpoonright\gamma(ag(e))$ changes), then do nothing else. If we are denied prompt permission, then enumerate $ag(e)$ into $\emptyset'$ and pick a fresh $ag(e)$. Again wait for $C\upharpoonright\gamma(ag(e))$ or $A\upharpoonright\gamma(ag(e))$ to change. If the latter happens first, do nothing else. If the former happens first, we enumerate $\varphi(e)$ into B.

12.3.4 Verification

Firstly note that we never wait forever at some step of the construction. Next, we show that each e only requires attention finitely often. Suppose no $e' < e$ requires attention anymore. Clearly no $\varphi(e')$ is enumerated into B anymore for $e' < e$. Suppose on the contrary, e requires attention infinitely often. Hence there are infinitely many stages $s_0 < s_1 < \cdots$ such that at each of these stages s_i, $ag(e)[s_i]$ is enumerated into $\emptyset'$ and a new agitator is picked. This is because if there were only finitely many such stages, then $y = \lim_s ag(e)[s]$ exists and e will no longer require attention after $\Gamma^{A\oplus C}\upharpoonright y+1$ becomes stable.

Suppose s_i is large, such that $e \in \emptyset'[s_i]$ iff $e \in \emptyset'$. Furthermore at the end of each stage s_i, we must have $\Phi^{A\oplus B}(e)\uparrow$ because $ag(e)[s^-] = ag(e)[s_i]$ and $\Gamma^{A\oplus C}(ag(e))$ is stable from s^- to s_i, where s^- is the stage where the $\Phi[s_i]$-axioms were set. Consequently at stages s_j for all $j > i$ we must have that (A5) applies. If prompt permission is never given at any of these stages then the set U_e is infinite (since each time we reset $ag(e)$ fresh after being denied prompt permission), and this produces a contradiction. On the other hand if prompt permission is given at some stage s_j then the $\varphi(e)$ use will be lifted above $j(e)$, and so $J^B(e)[s_j]\downarrow$ on the correct use by the induction hypothesis, and e does not need to act at stage s_{j+1}, another contradiction.

So, each e only requires attention finitely often, and each $\varphi(e)$ settles. Consequently for each e, we have $\Phi^{A\oplus B}(e)\downarrow=\emptyset'(e)$ (Φ is clearly a consistent set of axioms, by the convention that $\gamma(x)[t]\leq\gamma(x)[s]$ if $t<s$). Also we have $\exists^\infty s(J^B(e)[s]\downarrow)\Rightarrow J^B(e)\downarrow$, so that B is low. Lastly we have to show that $B\leq_{wtt}C$: fix an x, and run the construction until stage x. If x is not yet picked as a φ-use, then $x\notin B$ (since uses are picked fresh). Otherwise there is some stage $s<x$ such that x is picked to be $\varphi(e)[s]$ for some e; at stage s we must have $y=\gamma(ag(e))[s]$ defined. Go to a stage $t>s$ where $C_t\upharpoonright y=C\upharpoonright y$, such that no $e'<e$ requires attention at t and $ag(e)[t]\downarrow$. Then it is not hard to see that we have $x\in B$ iff $x\in B_{t+1}$.

12.4 Every low$_2$-cuppable set is AC-cuppable

Theorem 12.4.1. *Suppose $\emptyset'\leq_T A\oplus C$ where C is low$_2$. Then, there is an array computable set B such that $\emptyset'\leq_T A\oplus B$.*

12.4.1 The method of exploiting low$_2$-ness

The usual way of using the low$_2$-ness of C, is to help us to successfully guess whether any reduction Γ^C we are considering is total, by using the fact that the totality of Γ^C can be described by a Δ^0_3-procedure. One can coordinate the answers to such questions on the construction tree, in a way such that the leftmost outcome gives the true answer to the totality of Γ^C. One can then convert certain infinite injuries into finite injuries, in a similar way as the Robinson's technique converts infinitary actions into finite activity for given low sets.

Recall that a c.e. set A is low$_2$ if and only if there is some $f\leq_T\emptyset'$ such that for every total $g\leq_T A$, we have $g(x)<f(x)$ for almost all x. The construction can be viewed as a game between ourselves and the opponent. The opponent has control over the function f, and the low$_2$ set C. He is responsible for giving us a computable approximation $f(x)[s]$ of $f(x)$. We will at the same time be building a reduction Δ^C (a different reduction will be built at each requirement), and we will

ensure that if we are unable to keep Δ^C total, then we would have an automatic win at the requirement. We will try and keep Δ^C total. Each axiom $\langle\sigma, x, y\rangle$ we enumerate into Δ will represent a *challenge* to the opponent, which can be viewed as a request for the certification of the correctness of σ as an initial segment of C. Note that if C is low (instead of merely low$_2$), then the opponent has to respond promptly to each of our challenges. This means that he has to either demonstrate that $\sigma \not\subset C$ by changing C below $|\sigma|$, or else he has to give us certification that $\sigma \subset C$. One of these two scenarios must occur, and we could wait for one of them to happen before proceeding with the construction. Since C might be noncomputable, the opponent could of course first certify that $\sigma \subset C$, and then later change C below $|\sigma|$, but he can only do so finitely often.

Now consider the situation we are in, that is, C is now merely low$_2$. Low$_2$ness is a weak form of lowness, and the opponent cannot after all change C too often. What sort of analogous game can we play with the opponent, which would still require him to give us certification of the correctness of initial C-segments? Each time we issue him a challenge by extending the domain of $\Delta^C(x)$, by enumerating an axiom $\langle\sigma, x, y\rangle$, he can now respond with one of the following:

(i) demonstrate $\sigma \not\subset C$ by changing $C\upharpoonright|\sigma|$,

(ii) certify that $\sigma \subset C$, in the sense that he switches the approximation of $f(x)$ to become larger than y,

(iii) do nothing at all for the time being.

The third outcome is annoying, because unlike in the low case, we now have to alter our strategies to live with the fact that the opponent can choose not to give us a definite answer (i) or (ii). Hence, as we proceed in the construction, we have to make sure that our future actions do not undo or destroy our previous attempts, because these might be pending the opponent's response in the future. The low$_2$-ness of C will force him to respond with (i) or (ii) at *almost every challenge we issue*, but possibly with a huge delay in between, conditional of course, on the fact that we keep Δ^C total.

The above description of the use of low$_2$-ness is slightly different from usual, which would normally entail using a Δ^0_3-guessing process to determine the totality of C-computable functions. However, the intuition behind our method is clear, in the sense that we want to force the opponent to show us "delayed certifications" of C-segments. By doing so, we are able to directly exploit the fact that "C should not change very often".

12.4.2 A related result

We will first discuss a related result of Downey, Greenberg, Miller and Weber [DGMW06]. The authors proved the following.

Theorem 12.4.2 (Downey, Greenberg, Miller and Weber [DGMW06]). *If A is promptly simple, there is an array computable set B such that $\emptyset' \leq_T A \oplus B$.*

From Theorem 12.4.2, we know that if $\emptyset' = \Gamma^{A\oplus C}$ where C is low, then we should be able to construct an array computable set B and a functional Φ such that $\emptyset' = \Phi^{A\oplus B}$. How would such a proof go?

Ishmukhametov [Ish97] showed that every c.e. array computable set is c.e. traceable. Using this, suppose now we are given a low C and Γ such that $\emptyset' = \Gamma^{A\oplus C}$. We want to build a c.e. traceable set B and a Φ such that $\emptyset' = \Phi^{A\oplus B}$. We need to satisfy the requirements

$$\mathcal{R}_e \quad : \quad \text{if } \Psi_e^B \text{ is total, then build a c.e. trace } \{T_x\}_{x\in\mathbb{N}} \text{ for } \Psi_e^B.$$

to ensure that B is c.e. traceable. The reduction $\emptyset' = \Phi^{A\oplus B}$ is built by movable markers. We maintain a set of markers $\varphi(0), \varphi(1), \cdots$ which should be viewed as the use of the functional $\Phi^{A\oplus B}$. When we enumerate a new axiom $\langle \sigma, x, y\rangle$ into the functional, we are actually setting the marker value $\varphi(x)\downarrow = |\sigma|$. The marker value (and the corresponding computation) persists until there is an A or B-change below $\varphi(x)$. When that happens, the previously enumerated computation becomes non-applicable, and $\varphi(x)$ becomes undefined. When this happens, we can move or lift the marker $\varphi(x)$ to a fresh value. Since all sets concerned are c.e., once a computation for input x becomes non-applicable, it will stay undefined until we decide to enumerate a new axiom and give a new marker value for input x. In the actual construction, we will need two separate sets of markers $\{\varphi_A(x)\}$ and $\{\varphi_B(x)\}$ for the A and B-use; however for simplicity we will assume that they are the same for the time being.

The global coding requirements work in the following way. Whenever some x enters $\emptyset'$, we will need to cancel our previously set axiom $\Phi^{A\oplus B}(x)$. We do this by enumerating $\varphi(x)$ into B, since this is the only thing we have direct control over. The strategy $\mathcal{R}_e$ wants to make B c.e. traceable. Hence, it wants to restraint B, each time after we had committed some current value $\Psi_e^B(x)[s]$ into the trace T_x. Each $\mathcal{R}_e$ will not be able to directly restraint the global coding requirements, so it will have to ensure that B does not change too often, by "disengaging" dangerous φ-markers, from below the B-use of Ψ^B-computations.

The construction takes place on a binary tree of strategies. The e^{th} level is devoted to the requirement $\mathcal{R}_e$. Each node α divides its main strategy into substrategies, which are run separately by individual α-modules $M_0^\alpha, M_1^\alpha, \cdots$. We divide $\mathbb{N}$ into infinitely many partitions, $zone_0^\alpha, zone_1^\alpha, \cdots$, and the module M_k^α looks after all the computations $\Psi_\alpha^B(x)$ for all $x \in zone_k^\alpha$. At the beginning, we set $zone_x^\alpha = \{x\}$ for all x. If x is currently in $zone_k^\alpha$, then the module M_k^α will only trace the value $\Psi_\alpha^B(x)[s]$ into T_x, if it manages to disengage $\varphi(k)$ from below the use of $\Psi_\alpha^B(x)[s]$ (i.e. it manages to make $\psi_\alpha(x)[s] < \varphi(k)$). If the computation is later on injured (either due to higher priority requirements acting or the global coding actions), we will have to lower the tolerance of x, and transfer control of x to a higher priority α-module. In particular, we will now set $zone_{j-1}^\alpha[s+1] = zone_j^\alpha[s]$ for every $j \geq k$. Thus, x previously had tolerance number k, but after the injury it would have a tolerance number of $k-1$, and it will now be handled by M_{k-1}^α. When x reaches a tolerance number of 0, then it would only be traced if M_0^α manages to disengage $\varphi(0)$, so $\Psi_\alpha^B(x)$ will be preserved forever. Hence, each trace T_x will have size at most x. There are two issues here, and we will describe how they are handled at a later stage; firstly we have to ensure that each $\varphi(k)$ is moved only finitely often so that $\Phi^{A\oplus B}$ is total. Secondly, we have to ensure that for almost all x, we have the $\Psi^B(x)$ values traced; after all it is useless if, for instance, every x reaches a tolerance number of 0.

Atomic strategy of a single M_k^α: The subscript α is dropped if the context is clear. We first describe how a single M_k^α intends to disengage $\varphi(k)$ from below the use $\psi(x)$ of every $x \in zone_k^\alpha$. We reserve infinitely many indices called agitators (by the Recursion Theorem), where we can specify if an index ag_k^α is in $\emptyset'$. We wait for $\Gamma^{A\oplus C}(ag_k^\alpha)[s]\downarrow$, and $\Psi^B(x)[s]\downarrow$, such that $\gamma(ag_k^\alpha) < \varphi(k)$ (in the latter case, we say that M_k^α is set correctly). If M_k^α is not set correctly, we will simply enumerate $\varphi(k)$ into B to clear the Φ-axioms, and move the $\varphi(k)$ marker above $\gamma(ag_k^\alpha)$ in order to set M_k^α correctly. Our ultimate aim is to force an $A\restriction\varphi(k)$-change so that we can lift the marker $\varphi(k)$ without damaging the $\Psi^B(x)$-computation. This is analogous to "requesting prompt permission". In this case, we enumerate a challenge $\Delta^C(p)[s]\downarrow$ with C-use $\sigma = C_s\restriction\gamma(ag_k^\alpha)$. As discussed in Section 12.4.1, the opponent has to either demonstrate that $\sigma \not\subset C$, or else he has to certify that $\sigma \subset C$. If the latter happens we could enumerate ag_k^α into $\emptyset'$ and wait for an A or C-change below $\gamma(ag_k^\alpha)$. If an A-change is given, then we could lift the $\varphi(k)$-marker (analogously, prompt permission has been given). If the opponent responds in any other way (i.e. prompt permission has been denied), then we would enumerate $\varphi(k)$ into B to kill the current $\Psi^B(x)$-computation, and wait for it to converge again. The point is that prompt permission can only be denied finitely often before M_k^α is finally successful, where we can then trace the $\Psi^B(x)$-computations.

Coordination between different nodes: Since each node α may make infinitely many enumerations into B (though only finitely often for each module), we have to equip α with two outcomes, ∞ to the left of f. A node $\beta_1 \supseteq \alpha^\frown f$ is not allowed to injure α, so at each stage when β_1 is visited, only the modules $M_j^{\beta_1}$ for $j >$ the previous α-expansionary stage, are allowed to act. However, a node $\beta_0 \supseteq \alpha^\frown\infty$ will wait for enough α-modules to be successful. That is, it will wait for all of $M_0^\alpha, \cdots, M_k^\alpha$ to successfully trace every computation they are looking after, before β_0 begins to act for $M_k^{\beta_0}$. By coordinating the actions of α and β_0 in this way, we ensure that α never injures any β_0-module, although it might be the case for instance, that $M_k^{\beta_0}$ acts and injures $M_{k+1}^\alpha, M_{k+2}^\alpha, \cdots$ with current traced computations. Note that M_k^α is safe from the actions of $M_k^{\beta_0}$, because every computation in $zone_k^\alpha$ has been successfully traced (and would have disengaged $\varphi(k)$). In the meantime, α will now start with the actions of $M_{k+1}^\alpha, M_{k+2}^\alpha, \cdots$. Therefore if $M_k^{\beta_0}$ itself is not injured anymore (remember that injury to $M_k^{\beta_0}$ can only come from below and never from above), it follows that M_{k+1}^α will eventually be successful and preserves its traced computations forever. We arrange for nodes of each length k to agree never to move $\varphi(k)$. In this way, we can ensure that each $\varphi(k)$ is only moved finitely often (since there are only finitely many nodes on the construction tree that can move it). Also this ensures that for instance, injury to $M_k^{\beta_0}$ will eventually cease, since the only modules which can injure it are of the form M_j^σ for some $j < k$ and $\sigma \supseteq \beta_0{}^\frown\infty$.

12.4.3 Incorporating low$_2$-ness into the strategy

In the proof above, C was low. The advantage was that we could run the atomic strategy for each α-module sequentially. That is, we could start with M_0^α, wait for it to successfully trace every computation in $zone_0^\alpha$, before starting M_1^α. If at any point in time, M_k^α is injured, we would drop back and start again sequentially beginning with M_k^α. Now suppose C is low$_2$. As discussed in Section 12.4.1, the opponent now has a third option available to him each time we issue him a challenge. He could choose to do nothing until a much later stage (he can wait for as long as he likes). There is still hope for us, since we do not actually need to build the c.e. trace for B in a sequential manner. We will now need to go through the modules in two passes. The first pass is done sequentially, where we go through $M_0^\alpha, M_1^\alpha, \cdots$ and issue a challenge to the opponent on behalf of each module. After the first pass through a module M_k^α, it will be *waiting* for the opponent to respond to the challenge. We cannot go though all the modules without the opponent responding to any of the challenges; if this happens we want to make sure that Δ_α^C is total which will then contradict the low$_2$-ness of C. On the other hand if the opponent goes back and responds to the challenge on some M_k^α, we will start the second phase and try to force a change in $A\lceil\varphi(k)$. If the opponent allows us to start the second phase on a module M_k^α, we will be guaranteed to at least make some progress at the opponent's expense. Note that the opponent is of course not obliged to respond to our challenges in a sequential fashion (even if Δ_α^C should turn out to be total). He could respond in a more or less random fashion, and he only has to make sure that in the limit, he responds to all but finitely many of our challenges.

Since the opponent does not have to respond promptly, a different situation arises. He could delay responding to the challenge at M_k^α, and allow us to first successfully trace the computations in M_j^α for some $j > k$. Suppose later on he responds to the challenge at M_k^α. He could force us to enumerate $\varphi(k)$ into C by certifying an incorrect C-segment (though he can only do this finitely often), which would in turn injure all the modules larger than k. Consequently, all the computations previously in $\cup_{j>k} zone_j^\alpha$ will have to lower their tolerance. In particular, $zone_k^\alpha$ itself might receive new values of x which were previously in $zone_{k+1}^\alpha$, but whose tolerance had been reduced. This is alright, because after all M_k^α has not yet been successful, and we could wait for $\Psi^B(x)$ to converge for all the new $x \in zone_k^\alpha$, before issuing a new challenge to the opponent. As mentioned earlier, the opponent can only respond incorrectly at each module finitely often, so each module can first become successful, and then only to have its traced computations injured by smaller modules later on, only finitely often.

There is a problem with this approach, for other requirements might be enumerating numbers into B, while M_k^α is waiting for the opponent to respond to an issued challenge. The reason why a promptly simple set A is so useful when combined with permitting, is because we can freeze the actions of all other requirements while waiting for the opponent to respond to an issued challenge. The point is that we will always benefit from any resulting A-change. We expect this to be a main issue here. Note that $\psi_\alpha(x)$ (for $x \in zone_k^\alpha$) could converge with a very large use, so that after a challenge is issued by M_k^α, we could change B below $\psi_\alpha(x)$. Now if $A \restriction \gamma(ag_k^\alpha)$ changes before $\Psi_\alpha^B(x)$ next converges, we would be unable to benefit from the A-change. The opponent could then certify that C was correct (on the old $\gamma(ag_k^\alpha)$-use), and never change C below that. This is bad for M_k^α, for it would have lost all progress it had previously made on the test $\Delta^C(p)$, and M_k^α would have to pick a new p for the purpose of issuing new challenges to the opponent. This might happen infinitely often, since $\Psi_\alpha^B(x)$ can converge on ever increasing use. Even though locally, α requires no definite action (Ψ_α^B being not total), the module M_k^α will end up moving the global marker $\varphi(k)$ infinitely often.

To deal with the above problem, we will do the following. We ensure that every time M_k^α issues a challenge on behalf of every $x \in zone_k^\alpha$, and then later on some computation in $zone_k^\alpha$ is injured (we cannot prevent these types of injuries), we will also have the corresponding challenge reset so that M_k^α can reuse the same test $\Delta_\alpha^C(p)$. To be more specific, we wait for $\Psi_\alpha^B(x)$ to converge believably (we will explain what it means to be a believable computation) for every $x \in zone_k^\alpha$, and then challenge the opponent (via $\Delta_\alpha^C(p)$) to certify that $C \restriction \gamma(ag_p^\alpha)$ is correct, where p is the largest such that $\varphi(p) < \psi_\alpha(x)$ for any $x \in zone_k^\alpha$. We will arrange for all $\beta \supset \alpha$ to respect all computations in $zone_k^\alpha$. Hence any action which can injure the computations in $zone_k^\alpha$ will have to be either due to the global actions (which are finite in nature), or due to the actions of some M_j^β where $\beta \subset \alpha$ and $j \leq p$. We say that a computation is M_k^α-believable, if all modules M_j^β where $\beta \subseteq \alpha$ and $j \leq p$ are set correctly for the relevant p. Therefore, the only reason why a module M_j^β enumerates $\varphi(j)$ into B and destroys the computation, must be due to one of the following:

(i) M_j^β is no longer set correctly, and it enumerates $\varphi(j)$ into B in an attempt to set itself correctly again. In this case, there must have been some change in C which now makes $\Delta_\alpha^C(p)$ now undefined, because of M_k^α-believability.

(ii) The opponent responded to some challenge put forward by M_j^β, where we would have enumerated ag_j^β in response, and observed a resulting $C\restriction\gamma(ag_j^\beta)$-change. In this case $\Delta_\alpha^C(p)$ is now undefined as well.

In either of the two cases above, M_k^α can now reuse the test $\Delta_\alpha^C(p)$ on the same p, so that previous progress on this test location is not wasted. Even if this happens infinitely often, M_k^α will only use finitely many agitators and hence only move $\varphi(k)$ finitely often. In this way, the actions of a larger module M_j^α may affect M_k^α, but no real injury is inflicted on M_k^α. Any Ψ_α^B-computation which is true, will eventually become believable. For each k, we need to arrange for two outcomes - $k\infty$ to the left of kf. Since α does not know what the nodes extending it are currently doing, we cannot require M_k^α-believability to include nodes $\beta_1 \supseteq \alpha^\frown kf$. Hence β_1 has to ensure that it never damages a convergent $\Psi_\alpha^B(x)$-computation which is pending the opponent's response, for every $x \in zone_k^\alpha$. Because of the possibility of $\psi_\alpha(x)$ going to ∞ for some $x \in zone_k^\alpha$, we need to have the outcome $k\infty$. A node $\beta_0 \supseteq \alpha^\frown k\infty$ is only visited if $\Delta_\alpha^C(p)$ is undefined, so M_k^α does not care what β_0 does during the $\alpha^\frown k\infty$-stages.

12.4.4 Requirements and conventions

We are given sets A and C and a Turing functional Γ such that $\emptyset' = \Gamma^{A\oplus C}$. Our job is to build a c.e. set B and Turing functional Φ such that $\emptyset' = \Phi^{A\oplus B}$. Since C is low$_2$, there is a function $f \leq_T \emptyset'$ such that for every total $g \leq_T A$, we have $g(x) < f(x)$ for almost all x. We let $\lim_s f(x)[s] = f(x)$ be a computable approximation to f.

We make B c.e. traceable by satisfying the requirements $\mathcal{R}_e$ for every e. If a set is c.e. traceable, the choice of the order does not matter. Hence we will make all the traces we build have a bound of identity size (any choice will do), i.e. $|T_x| \leq x$.

The usual convention regarding stage numbers and notations applies. The uses of the functionals Γ, Ψ and Φ are denoted respectively by γ, ψ and φ. The functional Φ is build as a c.e. set of axioms, in the same way as in Theorem 12.3.1. We also assume the usual convention that if $\Gamma^{A \oplus C}(x)$ (similarly $\Psi^B(x)$) converges at stage s, then the use satisfies $x < \gamma(x) < s$. During the actual construction, there may be several actions taken one after another in a single stage s. It is sometimes convenient to break down a single stage into substages where a single action is taken at each substage. In this construction we will not bother with distinguishing between a stage and its substages; when we refer to a stage s we actually mean the instance within the stage s (or the substage of s) where the action is taken.

We introduce a minor difference in notations for the Φ-use. We will have separate notations for the A and B-use, denoted respectively by $\varphi_A(e)$ and $\varphi_B(e)$. One can thus think of the actual Φ-use as being $2\max\{\varphi_A(e), \varphi_B(e)\} + 1$, where extra axioms are enumerated into Φ by padding.

12.4.5 The construction tree

The construction takes place on an infinite branching tree. Nodes at level e are devoted to the requirement $\mathcal{R}_e$. There are two groups of outcomes, and we will alternate between the two groups in the ordering: $1\infty <_L 1f <_L 2\infty <_L 2f <_L 3\infty <_L 3f <_L \cdots$. The first group of outcomes $1\infty, 2\infty, \cdots$ are called *infinite outcomes*, in order to distinguish these from the second group $1f, 2f, \cdots$, which we will call the *finite outcomes*. The choice of these names has little to do with the frequency of actions and when the respective outcomes are played; in the actual construction the finite outcomes do have infinitary actions, however they are tagged as "finite" simply because certain associated Ψ_e^B-computations converge in the limit.

The ordering amongst nodes is denoted by $\alpha <_L \beta$, which means that α is strictly to the left of β (i.e. there is some $i < \min\{|\alpha|, |\beta|\}$ such that $\alpha\restriction i = \beta\restriction i$ and $\alpha(i) <_L \beta(i)$). We use $\alpha \subset \beta$ to mean that β extends α properly, and $\alpha \subseteq \beta$ to mean $\alpha \subset \beta$ or $\alpha = \beta$. We also say that $\alpha \subset_\infty \beta$ if $\alpha^\frown n\infty \subseteq \beta$ for some n, and $\alpha \subset_f \beta$ if a similar situation holds with f instead of ∞. We also write $\alpha \subseteq_\infty \beta$ and $\alpha \subseteq_f \beta$ to have the obvious meaning.

We say that α *is an* $\mathcal{R}_e$*-node*, if α is assigned the requirement $\mathcal{R}_e$. As usual if α is an $\mathcal{R}_e$-node then we write Ψ^B_α and Ψ^B_e to denote the same thing. Each node α measures the totality of Ψ^B_α. As described previously, $\alpha^\frown k\infty$ will be visited if M^α_k has its current test reset, i.e. $\Delta^C_\alpha(p)$ undefined, so that M^α_k does not care what happens in the region $[\alpha^\frown k\infty]$. Whenever α plays this outcome, we initialize all nodes $\beta \supseteq \alpha^\frown o$ for any outcome o to the right of $k\infty$. Whenever M^α_k progresses in its atomic strategy, we will visit the outcome kf, and initialize all nodes extending an outcome $o >_L kf$. Any node $\beta \supseteq \alpha^\frown kf$ will coordinate its actions with α as described previously. That is, M^β_j will wait until all the modules $M^\alpha_k, M^\alpha_{k+1}, \cdots, M^\alpha_j$ are successful, before M^β_j is allowed to act (this prevents α-modules from injuring β-modules). To ensure the true path of construction exists, whenever a module M^α_k becomes successful, we will visit the outcome $k'f$ where k' is the least place $\leq k$ where a run of successful modules starts. For instance, using s to denote success, and w otherwise, we could have initially:

Module	M^α_0	M^α_1	M^α_2	M^α_3	M^α_4	M^α_5	M^α_6	M^α_7	M^α_8	$\cdots$
State	w	w	w	w	w	w	w	w	w	$\cdots$

Outcome played $= 0f$. Then we have:

Module	M^α_0	M^α_1	M^α_2	M^α_3	M^α_4	M^α_5	M^α_6	M^α_7	M^α_8	$\cdots$
State	w	w	s	w	w	w	w	w	w	$\cdots$

Outcome played $= 2f$.

Module	M^α_0	M^α_1	M^α_2	M^α_3	M^α_4	M^α_5	M^α_6	M^α_7	M^α_8	$\cdots$
State	w	w	s	w	w	s	w	w	w	$\cdots$

Outcome played $= 5f$.

Module	M^α_0	M^α_1	M^α_2	M^α_3	M^α_4	M^α_5	M^α_6	M^α_7	M^α_8	$\cdots$
State	w	w	s	w	w	s	w	w	s	$\cdots$

Outcome played $= 8f$.

Module	M_0^α	M_1^α	M_2^α	M_3^α	M_4^α	M_5^α	M_6^α	M_7^α	M_8^α	$\cdots$
State	w	w	s	w	w	s	s	w	s	$\cdots$

Outcome played $= 5f$.

Module	M_0^α	M_1^α	M_2^α	M_3^α	M_4^α	M_5^α	M_6^α	M_7^α	M_8^α	$\cdots$
State	w	w	s	w	s	s	s	w	s	$\cdots$

Outcome played $= 4f$.

Module	M_0^α	M_1^α	M_2^α	M_3^α	M_4^α	M_5^α	M_6^α	M_7^α	M_8^α	$\cdots$
State	w	w	s	s	s	s	s	w	s	$\cdots$

Outcome played $= 2f$.

Hence if Ψ_α^B is total, then Δ_α^C is total and the opponent has to respond at all but finitely many of the modules. In between, infinitary outcomes may be played from time to time, as different Ψ_α-computations are injured, but the above will describe the situation in the limit. In the above example we will visit α-outcome $2f$ infinitely often, and so a node $\beta \supset \alpha^\frown 2f$ can wait patiently to start any β-module. On the other hand if some infinite outcome $k\infty$ is played infinitely often then it must be that some $x \in zone_k^\alpha$ is divergent and Δ_α^C is not total, so that the opponent doesn't have to respond at all. However this is an automatic win for us at α.

12.4.6 Notations for the formal construction

Each α builds a c.e. trace $\{T_x^\alpha\}_{x\in\mathbb{N}}$, and ensures that for all x, $|T_x^\alpha| \leq x$. For each α and x, we have a parameter t_x^α which gives keeps track of an upper bound for $x - |T_x^\alpha[s]|$. At the beginning (each time α is initialized) we start with $t_x^\alpha = x$. Every time a value $\Psi_\alpha^B(x)[s]$ is traced in T_x^α and later $B\upharpoonright\psi_\alpha(x)$ is changed, we will decrease t_x^α by 1. When t_x^α reaches 1 (if ever), any current value traced must be preserved forever. This ensures that $|T_x^\alpha| \leq x$. During the construction we will sometimes say that we *lower* t_x^α. This simply means that we decrease the value of t_x^α by 1. When we lower t_x^α, we also lower t_y^α for every $y > x$ at the same time. The inverse of the parameter t_x^α is denoted by the parameter $zone_k^\alpha$, which is the set of all numbers x such that $t_x^\alpha = k$. Each node α will be building a Turing functional Δ_α, which is used as a test. Following conventions, the use of currently applicable computations $\Delta_\alpha^C(x)$ is denoted by $\delta_\alpha(x)$.

Every x in the same zone has the same goals; they all want to "disengage" the same φ_B-markers from below their use $\psi_\alpha^B(x)$. That is, every x in $zone_k^\alpha$ wants to ensure that $\psi_\alpha^B(x) < \varphi_B(k)$. The k^{th} module M_k^α will act on behalf of all the $x \in zone_k^\alpha$, and will elect a number p_k^α called a *follower*. The idea is that M_k^α will be enumerating computations for $\Delta_\alpha^C(p_k^\alpha)$. It also appoints a number ag_k^α called an *agitator*, and as the name suggests, this number will be used to force an $A \oplus C$-change.

We will introduce what we call global agitators, denoted by Ag_k. That is, Ag_k will be used to help in the coding of $\emptyset'(k)$. The primary aim of the global agitator is to ensure that if k enters $\emptyset'$ and we need to code, we will only change $B \restriction \varphi_B(k)$ if we have an accompanying $C \restriction \gamma(Ag_k)$-change. Since we only need to use up Ag_k if coding occurs, we may choose and fix the values for the global agitators in advance.

Each module M_k^α is in a particular state at any point in time. It can either be *unstarted*, *ready*, *waiting* or $successful$. Roughly speaking, being in the state *unstarted* means that the module has just been initialized and needs to pick a new follower and agitator. The state *ready* represents the fact that we are waiting for an appropriate chance to define $\Delta_\alpha^C(p_k^\alpha)$. When the module passes to state *waiting*, we have defined $\Delta_\alpha^C(p_k^\alpha)$ and are now waiting for the opponent to respond with $f(p_k^\alpha) > \Delta_\alpha^C(p_k^\alpha)$. Lastly the state $successful$ represents the fact that we have managed to disengage all dangerous markers, and have increased the trace size by 1 for all $x \in zone_k^\alpha$. The state of a module will retain its assigned value until we assign a new state to it. There is one exception to this: if a module M_k^α has state *waiting* and a C-change occurs so that we have $\Delta_\alpha^C(p_k^\alpha)$ becomes undefined, then we set the module to have state *ready* (this is assumed to be a background task which is done implicitly, we do not mention this step in the construction). The reason why we want to implement this is to be consistent with the fact that

$$\begin{aligned} M_k^\alpha \text{ has state } \textit{ready} &\;\Rightarrow\; \Delta_\alpha^C(p_k^\alpha) \text{ is undefined,} \\ M_k^\alpha \text{ has state } \textit{waiting} &\;\Rightarrow\; \Delta_\alpha^C(p_k^\alpha) \text{ is defined.} \end{aligned}$$

We say that the module M_k^α is *set correctly*, if its state is not *unstarted*, and $\gamma(ag_k^\alpha) \leq \varphi_A(k)$, with everything mentioned here defined. That is, a module is said to be set correctly if its agitator has its use in the correct place. Similarly, the global agitator Ag_k is said to be set correctly, if $\gamma(Ag_k) \leq \varphi_A(k)$. We define l_k^α to be the module ending the longest run of consecutive α-modules starting with M_k^α, that are in the state $successful$. That is, $l_k^\alpha =$ least number $j \geq k$ such that M_j^α is not in the state $successful$.

Definition 12.4.3 (Active modules)**.** With each node α we associate a collection of α-modules which we call *active modules.* At stage s, a module M_k^α is said to be active, if the following holds:

1. $k > |\alpha|$,
2. for every $\beta \subset \alpha$ and n such that $\beta^\frown nf \subseteq \alpha$, we have $n < k < l_n^\beta$,
3. for every $\beta \subset \alpha$ and n such that $\beta^\frown n\infty \subseteq \alpha$, we have $n < k$ and for every $n \leq k' \leq k$, we have $M_{k'}^\beta$ is either $successful$ or is set correctly,
4. Ag_n is set correctly for all $n \leq k$.

When we visit α during the construction, only the α-modules which are currently active get a chance to act. The idea is that once a module M_k^β enters the state $successful$, it never moves $\varphi_B(k)$ anymore, and so when $\alpha \supset_f \beta$ is visited we only allow α-modules which are currently active to act. This is to prevent injury to the α-modules by β-modules.

Definition 12.4.4 (Believable computation)**.** We define what we mean by a believable computation. A computation $\Psi_e^B(x)$ which converges at a stage s with B-use u, is said to be *M_k^α-believable*, if for every $z \geq k$ such that $\varphi_B(z)\downarrow< u$, we have M_z^α is currently active, and either $successful$ or set correctly.

When we *initialize a module M_k^α*, we set p_k^α and ag_k^α to be undefined, and declare the state of M_k^α to be *unstarted.* When we *initialize a node α*, we first initialize all of its modules. Then, we set $T_x^\alpha = \emptyset$ and $t_x^\alpha = x$ for all x, and set $\Delta_\alpha = \emptyset$.

12.4.7 The ordering amongst modules

The driving force behind the construction is the action of the individual modules (instead of the actions of nodes). During the construction when a node α is visited, we will take actions for some α-module. This action will affect and injure β-modules, possibly for all nodes β comparable with α. Due to these interactions between the modules of different nodes, we will introduce the following two concepts.

We first define a local priority ordering amongst the modules (of different nodes). Note that a module M_k^α only enumerates current $\varphi_B(k)$-marker values into B (under its individual strategy). If M_k^α is an α-module, we define the set of modules with a *lower local priority*, to be all the modules M_i^β for some $\beta \subset_f \alpha$, and $k \leq i$. We also include all the modules M_i^α for $i > k$ in this list (i.e. larger α-modules are also of a lower local priority).

We say that a computation $\Psi_\beta^B(x)[s]$ is *a current traced computation* at s, if $\Psi_\beta^B(x)[s] \downarrow \in T_x^\beta$, and there is no change in B below $\psi_\beta(x)$ since the time the value was traced in T_x^β. Furthermore we also require that t_x^β was not lowered since the time the value was traced. That is, current traced computations are the computations that we should not allow to be injured easily. Another minor technical point to note is the following. We will think of the trace T_x^β as tracing the use of $\Psi_\beta^B(x)[s]$, instead of the value of the output. Therefore if the computation $\Psi_\beta^B(x)$ is traced at two different times, we will have two different strings in the trace T_x^β. At stage s, we will also say that a computation $\Psi_\alpha^B(x)$ has *persisted for two visits to* α, if $\Psi_\alpha^B(x)[s^-] \downarrow$ and there has been no change in B below the ψ-use between s^- and s, where s^- is the previous visit to α. We say that a computation has persisted between two stages $s < t$ if the above holds with the obvious modifications. The point of making this definition is to identify the following situation: a module M_i^β may currently have a traced computation $\Psi_\beta^B(x)$, but it might have received initialization after the value of $\Psi_\beta^B(x)$ was put into T_x^β. Namely, it is possible for a module to have a current traced computation, but it is not *$successful$*.

Next, for an α-module M_k^α, we define the *injury set of* M_k^α to be the set of modules M_i^β where $\beta \supset \alpha$, $i > k$, and such that for some $x \in \mathit{zone}_i^\beta$, we have $\Psi_\beta^B(x)$ is a current traced computation (in this case, we also say that M_i^β has a current traced computation). Note that if M_i^β is *$successful$*, we will always consider it to be in the injury set as well. That is, M_i^β is a module in which some $\Psi_\beta^B(x)$-computation which has already been traced, would be injured if M_k^α decides to act and enumerate $\varphi_B(k)$ into B. Note that the modules of a lower priority depend only on the layout of the tree and do not change with time, while the injury set varies with time.

12.4.8 The construction

At stage $s = 0$, initialize all nodes and do nothing. At stage $s > 0$ we define the accessible string δ_s, of length s inductively on its length. A node α is visited at stage s, if $\delta_s \supset \alpha$. Suppose $\alpha \subset \delta_s$ has been defined. We state the actions to be taken by α at stage s, and its successor along the accessible string. Pick the smallest k (if it exists) such that $s^- < k < s$ where $s^- < s$ is the previous stage where $\delta_{s^-} <_L \alpha$, and M_k^α is currently active and requires attention. We say that M_k^α *requires attention*, if one of the following holds:

(A1) M_k^α is *unstarted.*

Otherwise, we have $\Phi^{A\oplus B}(k)\downarrow$, $\Gamma^{A\oplus C}(ag_k^\alpha)\downarrow = \emptyset'(ag_k^\alpha)$, and as well as one of the following:

(A2) M_k^α is *ready*, and let s^- be the previous visit to α. There exists some largest α-stage $t \leq s^-$ such that after α has acted at t, we have M_k^α not *ready*. We require that for some $x \in zone_k^\alpha$, $\Psi_\alpha^B(x)$ has persisted between t and s^-, but not between s^- and s. In short, some relevant computation has recently been destroyed (and possibly by α itself at s^-).

(A3) M_k^α is *ready*, but is not set correctly.

(A4) M_k^α is *ready*, and for every $x \in \cup_{j\leq k} zone_j^\alpha$, $\Psi_\alpha^B(x)[s]\downarrow$ and is M_k^α-believable.

(A5) M_k^α is *waiting*, and $\gamma(ag_k^\alpha) > \delta_\alpha(p_k^\alpha)[s]$.

(A6) M_k^α is *waiting*, and $\Delta_\alpha^C(p_k^\alpha)[s] < f(p_k^\alpha)[s]$.

Step 1: We will act for the α-module M_k^α. At each visit to α, at most one α-module receives attention. Choose the first item in the list above that applies, and take the corresponding action:

- (A1) applies: Pick a fresh follower p_k^α and a fresh agitator ag_k^α. Declare M_k^α to be in *ready* state.

- (A2) applies: Do nothing. This is included to ensure that infinite outcomes of α get a chance to act.

- (A3) applies: Enumerate $\varphi_B(k)$ (if defined) into B.

- (A4) applies: Let p be largest such that $\varphi_B(p) \downarrow < \psi_\alpha(x)$ for some $x \in \cup_{j \leq k} zone_j^\alpha$ (we always take $p \geq k$), and let $z = \max\{Ag_0, \cdots, Ag_p\} \cup \{ag_q^\beta \mid \beta \subseteq_\infty \alpha \text{ and } q \leq p\}$. Note that all the parameters involved must be defined and set correctly, because of M_k^α-believability. Let $m = \max\{\delta_\alpha(y)[s] \downarrow \mid y \leq p_k^\alpha\}$. Define $\Delta_\alpha^C(y) \downarrow =$ a fresh number, with use $C_s \upharpoonright \gamma(z) + m$, for every $y \leq p_k^\alpha$ where no axioms currently apply for $\Delta_\alpha^C(y)[s]$. Declare M_k^α as *waiting*.

- (A5) applies: Declare M_k^α as $successful$. For every $x \in zone_k^\alpha$, we enumerate the value of $\Psi_\alpha^B(x)[s]$ into the trace T_x^α.

- (A6) applies: Enumerate ag_k^α into $\emptyset'$ and pick a fresh agitator. Wait for either $C\upharpoonright\gamma(ag_k^\alpha)$ or $A\upharpoonright\gamma(ag_k^\alpha)$ to change (one of the two must change). If A changes do nothing, else if C changes then we enumerate $\varphi_B(k)$ (if defined) into B.

If an enumeration was made into B in Step 1, we say that *the module M_k^α was injurious.* We separate this case from the rest because these are the "bad actions" which will injure the other modules on the tree.

Step 2: We now determine the effect that our action in step 1 has (if any) on the other modules in the construction. If M_k^α was not injurious, we do nothing. Otherwise we do the following.

- for every $x \in zone_j^\beta$ such that M_j^β is of lower local priority (than M_k^α) and $j > k$, we lower t_x^β.

- we initialize all modules of a lower local priority.

- for every $\beta \supset \alpha$, do the following. Let $i > k$ be the least (if any) such that M_i^β is in the injury set of M_k^α. Note that we naturally consider the injury set before step 1 is taken, so that M_i^β is the smallest β-module which has a current traced computation being injured by the action in step 1. We then initialize all modules M_j^β for all $j \geq i - 1$, and we lower t_x^β for every $x \in \cup_{j \geq i} zone_j^\beta$.

Step 3: Now decide which outcome of α is accessible. Suppose that M_k^α has just received attention (if no module received attention let $k = s$). If (A5) was the action taken, let $\delta_s(|\alpha|) = jf$ where $s^- < j \leq k$ is the largest such that M_j^α is active and M_{j-1}^α is not *successful*. Otherwise if (A5) was not the action taken, we search for the least j with $s^- < j \leq k$ such that M_j^α is *ready* and active, and for some $x \in zone_j^\alpha$, $\Psi_\alpha^B(x)$ has not persisted for at least two visits to α. If j exists, we let $\delta_s(|\alpha|) = j\infty$, otherwise we let $\delta_s(|\alpha|) = kf$.

Global actions: At the end of stage s, we initialize all nodes $\beta >_L \delta_s$, and take the *global actions* for coding: pick the smallest $e < s$ such that either

(i) Ag_e is not set correctly, or

(ii) $\Phi^{A\oplus B}(e) \downarrow \neq \emptyset'(e)$

holds. If (i) holds and $\varphi(e)\downarrow$, we will enumerate $\varphi_B(e)$ into B. On the other hand if (ii) holds, then we will enumerate Ag_e into $\emptyset'$, and wait for $A\restriction\gamma(Ag_e)$ or $C\restriction\gamma(Ag_e)$ change. If $C\restriction\gamma(Ag_e)$ changes we will enumerate $\varphi_B(e)$ into B. In addition, if we had enumerated $\varphi_B(e)$ into B in the previous step, we will also perform the following: for every node α and every $k \geq e$, we initialize the module M_k^α. We also lower t_x^α for every $x \in zone_k^\alpha$ where $k > e$.

Next, we extend the domain of Φ. Wait for $\Gamma^{A\oplus C}(z) \downarrow= \emptyset'(z)$ for every $z \leq$ the largest number used so far, and $z < s$. For every $e < s$ for which no axiom is applicable to $\Phi^{A\oplus B}(e)$, we set $\Phi^{A\oplus B}(e) \downarrow= \emptyset'(e)$ with a fresh B-use $\varphi_B(e)$, and A-use $\varphi_A(e)$ equals to $\max\{\gamma(ag_e^\alpha) \mid \text{for any } \alpha \text{ such that } ag_e^\alpha \downarrow\} \cup \{\gamma(Ag_e)\}$. Clearly there is a danger that the A-use $\varphi_A(e)$ might get driven to infinity, but we will later show that this cannot be the case.

12.4.9 Verification

It is easy to verify the following facts.

Fact 12.4.5. *Suppose M_j^α is a module which is active at a stage s. Then for any $k < j$, either M_k^α is also active at s, or else M_k^α will never be active after s.*

Fact 12.4.6. *Observe that we never wait forever at some step of the construction. When a module M_k^α acts, it can only enumerate the current marker value of $\varphi_B(k)$*

into B. If M_x^α is initialized, then all larger α-modules are also initialized by the same action. Furthermore if M_x^α is initialized then at the same time either the node α is initialized, or $\varphi(x)$ becomes undefined (due to changes below the φ-use on either the A or B side). For any α and k, $zone_k^\alpha$ is never empty. If M_j^α and M_i^α are both active at some stage s, and $j < k < i$, then M_k^α is also active at s. If $p_j^\alpha \downarrow$ and $p_k^\alpha \downarrow$ for $j < k$, then we have $p_j^\alpha < p_k^\alpha$.

It is also not too difficult to verify the following fact. The nontrivial case to consider is when M_{k-1}^α takes an injurious action, and initializes M_k^α. In this situation, M_{k-1}^α itself is not initialized, however its state will be *ready*:

Fact 12.4.7. *If an action is taken to lower t_x^α for some $x \in zone_k^\alpha$, then M_k^α will also be initialized and M_{k-1}^α will become either unstarted or ready.*

Lemma 12.4.8. *Suppose M_k^α receives attention at s, and $\varphi(k)[s] \downarrow$ and $s^- < s$ is the previous visit to the left of α. Then, every module M_i^α which is active at s, for $s^- < i < k$ must either be $successful$ or be set correctly at s.*

Proof. M_i^α cannot be *unstarted* (otherwise M_i^α would have received attention at s). If M_i^α is *ready* then it has to be set correctly (otherwise again, M_i^α would have received attention). This leaves the case M_i^α is *waiting*. Since $\varphi(i)$ is defined at s, by considering $\Delta_\alpha^C(p_i^\alpha)$, it is not difficult to see that at s we must have M_i^α set correctly as well. □

Lemma 12.4.9. *Suppose M_k^α is waiting at stage s. Then for every $x \in \cup_{j\leq k} zone_j^\alpha$, we have $\Psi_\alpha^B(x)[s] \downarrow$. Furthermore $B\restriction\psi_\alpha(x)[s]$ cannot change before M_k^α has a change of state.*

Proof. Let $s_0 \leq s$ be the largest stage when (A4) holds to change M_k^α from *ready* to *waiting*. By Fact 12.4.7, $zone_j^\alpha[s_0] = zone_j^\alpha[s]$ for every $j \leq k$, it suffices to prove the lemma with s_0 in place of s. Fix an $x \in zone_k^\alpha[s_0]$, clearly $\Psi_\alpha^B(x)[s_0] \downarrow$. Suppose that $\varphi_B(z)[s_0] < \psi_\alpha(x)[s_0]$ is enumerated into B at some least stage $t > s_0$, and for some z. Suppose to the contrary that β has had no state change between s_0 and t. Suppose firstly that the enumeration was made by the global actions. We have Ag_z is set correctly at s_0; this follows from the fact that M_k^α is active at s_0 (if $z < k$), and M_k^α-believability if $z \geq k$. In either case we have $\delta_\alpha(p_k^\alpha)$ set to be larger than $\gamma(Ag_z)[s_0]$ at s_0, and consequently neither $\gamma(Ag_z)$ nor $\varphi(z)$ could have changed between s_0 and t. Hence, the global actions at t would destroy the $\Delta^C_\alpha(p^\alpha_k)$-computation set at s_0, resulting in a change of state at t.

Next, we want to show that at stage s_0 the outcome kf is played when α is visited: if not then there is some $j < k$ with $j > s_0^-$ such that M^α_j is *ready* and active. For $x \in \cup_{y \leq j} zone^\alpha_y$, we claim that the $\Psi^B_\alpha(x)[s_0]$-computation (besides being M^α_k-believable) is also M^α_j-believable at s_0. This follows by Lemma 12.4.8 because every module M^α_i for $j \leq i < k$ must be active (since M^α_j and M^α_k are). Hence M^α_j would have received attention instead of M^α_k at s_0, a contradiction.

Suppose now that the enumeration of $\varphi_B(z)[s_0]$ was made by the module M^β_z for some node β, at stage t. Clearly the cases $\beta <_L \alpha$, or $\beta >_L \alpha$ are trivial. Suppose $\beta \supseteq \alpha^\frown o$ for some outcome o. Again $o >_L kf$ is trivial, and from the fact that M^α_k is *waiting* between s and t, it follows that $o = kf$ is impossible. Suppose that $o = k'f$ for some $k' < k$. Since M^β_z is active at t it follows that at t we have $z < l^\alpha_{k'} \leq k$, which means that after M^β_z acts at t, we would initialize M^α_k (M^α_k being of lower local priority). If on the other hand we have $o = k'\infty$ for some $k' \leq k$, then stage t is strictly after s_0. We may assume $k' < k$ otherwise M^α_k is *ready* at t. At t there is some $x' \in zone^\alpha_{k'}[t]$ such that $\Psi^B_\alpha(x')$ has not persisted for two visits to α. Since M^α_k had no change in state between s_0 and t, it follows that $x' \in zone^\alpha_{k'}[s_0] = zone^\alpha_{k'}[t]$, which means that $x' < x$. Since $\Psi^B_\alpha(x')$ had not persisted between s_0 and t, this contradicts the minimality of t.

We are now left with the case $\beta \subseteq \alpha$. If $\beta^\frown nf \subseteq \alpha$ then at s_0 we have $z < l^\beta_n$ (again due to either M^α_k being active or M^α_k-believability). If $z < n$ then at t some outcome to the left of nf will be played when β acts, resulting in an initialization to α. Suppose therefore, that $n \leq z < l^\beta_n$, and hence M^β_z is *successful* at s_0. The only way for M^β_z to leave state *successful*, is for M^β_z to be initialized before t, and so by Fact 12.4.6, either β itself is initialized or $\varphi_B(z)$ is lifted between s_0 and t, another contradiction.

Finally we have the case $\beta^\frown n\infty \subseteq \alpha$ or $\beta = \alpha$. We can conclude (again due to either M^α_k being active or M^α_k-believability) that M^β_z is either *successful* or is set correctly at s_0 (for $\beta = \alpha$ and $z < k$, use Fact 12.4.5 and Lemma 12.4.8). A similar argument as the one used above can be applied to show that M^β_z cannot be *successful*. Hence M_z^β has to be set correctly at s_0, and furthermore α at s_0 will set the use $\delta_\alpha(p_k^\alpha)[s_0]$ at least as big as $\gamma(ag_z^\beta)[s_0]$. Between s_0 and t there can be no change in A below $\gamma(ag_z^\beta)[s_0]$, because M_z^β was observed to be set correctly at s_0. There is also no change in C below $\gamma(ag_z^\beta)[s_0]$, since M_k^α was assumed to have no state change. Hence at t when β gets to act, it must still be that M_z^β is set correctly, and that (A6) applies, causing us to enumerate ag_z^β and get a $C \upharpoonright \gamma(ag_z^\beta)$-change, which would in turn cause $\Delta_\alpha^C(p_k^\alpha)$ to become undefined after the action at stage t. □

Lemma 12.4.10. *At all times, if the module M_k^α is active, and $k >$ stage number of the previous visit to the left of α, the following are true:*

(i) M_k^α is not unstarted $\Leftrightarrow$ p_k^α and ag_k^α are both defined.

(ii) M_k^α has state ready $\Rightarrow$ $\Delta_\alpha^C(p_k^\alpha)$ is undefined.

(iii) M_k^α has state waiting $\Rightarrow$ $\Delta_\alpha^C(p_k^\alpha)$ is defined.

Proof. (i) and (iii) are obvious.

(ii): Suppose to the contrary that s is a stage such that M_k^α is *ready* and is active, but $\Delta_\alpha^C(p_k^\alpha)$ is has an axiom that applies. Let $s^- < s$ be the latest action that caused M_k^α to become *ready*. Note that this must be either M_k^α receiving attention under (A1), or due to some C-change occurring. In any case it must be that $\Delta_\alpha^C(p_k^\alpha)[s^-]$ is undefined. Hence at some time t where $s^- < t < s$ we have some module M_j^α where $j > k$ taking action under (A4) and enumerating the axiom for $\Delta_\alpha^C(p_k^\alpha)[s]$. Since M_j^α is active at t, it follows by Fact 12.4.5 that M_k^α is also active at t, and in fact by Lemma 12.4.8, every M_i^α has to be active, and has to be either *successful* or be set correctly at t for all $k \leq i < j$. Hence at stage t when α was visited, it is not hard to see that we have $t^- < k < t$, and also that M_k^α requires attention under (A4), because every relevant computation converges and is M_k^α-believable at t. Hence it is impossible for M_j^α to act at t, a contradiction. □

Lemma 12.4.11. *Suppose that M_k^α has just received attention under (A5) at s. Then for every $x \in \cup_{j \leq k} zone_j^\alpha[s]$, we have $\Psi_\alpha^B(x)[s]\downarrow$. Furthermore if $\varphi_B(k)[s]\downarrow$, then $\varphi_B(k)[s] > \psi_\alpha(x)[s]$.*

Proof. Let $s_0 < s$ be the stage where (A4) applies to M_k^α and causes the state to change from *ready* to *waiting*. It follows again by Fact 12.4.7 that $zone_j^\alpha[s_0] = zone_j^\alpha[s]$ for all $j \leq k$, and by Lemma 12.4.9, we have $\Psi_\alpha^B(x)[s_0]\downarrow$, and this computation is not injured between s_0 and s. Furthermore at s_0 we had set $\Delta_\alpha^C(p_k^\alpha)\downarrow$ with use $u = \delta_\alpha(p_k^\alpha)$, where we clearly have M_k^α set correctly with $\gamma(ag_k^\alpha) < u$. It also follows that the values of $C \restriction u$ and p_k^α did not change between s_0 and s, so consequently $A\restriction\gamma(ag_k^\alpha)[s_0]$ has to change between s_0 and s in order for (A5) to hold at s. Because M_k^α was set correctly at s_0, it follows that $\varphi_B(k)[s]$ is picked fresh and so is larger than $\psi_\alpha(x)[s_0]$ (this is why it is important that $B\restriction\psi_\alpha(x)[s_0]$ did not change between s_0 and s, so that we can benefit from the A-change in between). □

Lemma 12.4.12. *Suppose $\varphi_B(x)[s]$ is enumerated into B at stage s (by any action), and for some α and $k > x$, M_k^α is either $successful$ or has a current traced computation at s. Then the same action will either*

- *initialize the node α, or*
- *initialize M_k^α, and lower t_x^α for every $x \in zone_k^\alpha$*

Proof. We proceed by induction on the sequence of actions (or substages) in the construction. If $\varphi_B(x)$ was enumerated by the global actions, then it is clear. So, we assume some M_x^β took an injurious action. We must have β and α comparable (otherwise it is trivial), and if $\beta \supseteq_f \alpha$ then it is easy. If $\beta \subset \alpha$ then one can verify that M_k^α would be in the injury set of M_x^β when β acted. The only case left is $\alpha^\frown n\infty \subseteq \beta$ for some n. We show this case is not possible.

At stage s we have M_k^α is either $successful$ or has a current traced computation. Clearly we have $k > x > n$. When α was visited earlier in the stage s, we had some $y \in zone_n^\alpha[s]$ where $\Psi_\alpha^B(y)$ has not persisted for two visits to α. If M_k^α has a current traced computation at s, then M_k^α would have received attention under (A5) and enumerated $\Psi_\alpha^B(z)[s_0]$ into T_z^α at some stage $s_0 \leq s$. Furthermore between s_0 and s, t_z^α is not lowered. On the other hand if M_k^α is successful at s, then it also receives attention under (A5) at some stage $s_0 \leq s$. We claim that in either case, we have $t_y^\alpha[s_0] \leq k$.

In the latter case, this follows because of Fact 12.4.7 and the fact that $k > n$. In the former case, we cannot apply Fact 12.4.7 directly because M_k^α might be initialized between s_0 and s. However if $z < y$ then $k = t^\alpha_z[s] \leq t^\alpha_y[s] = n$ is a contradiction, and on the other hand if $z \geq y$ then $k = t^\alpha_z[s_0] \geq t^\alpha_y[s_0] > k$.

In that case it follows by Fact 12.4.7 and by Lemma 12.4.9 that $\Psi^B_\alpha(y)[s_0]\downarrow$ and at s_0 it had already persisted for two visits to α. Therefore the visit to α at s_0 and the visit to β at s cannot take place in the same stage. Therefore $B\upharpoonright\psi_\alpha(y)[s_0]$ has to change at some action strictly between s_0 and s. By Lemma 12.4.11 this has to be $\varphi_B(x')$ for some $x' < k$. Apply the induction hypothesis to get a final contradiction, and hence we conclude that the case $\alpha \subset_\infty \beta$ is also not possible. □

Lemma 12.4.13. *For any β and e, if M^β_e is initialized finitely often, then it only picks finitely many agitators ag^β_e.*

Proof. Suppose t_0 is a stage after which M^β_e is never initialized. Then $p = \lim p^\beta_e$ must exist. Since after t_0, we only pick a new agitator if (A6) applies, it follows that we may assume that there are infinitely many stages $t_1 < t_2 < \cdots$ larger than t_0 such that M^β_e receives attention under (A6) at all of these stages. When M^β_e next receives attention after each t_n, we must have M^β_e *ready* (otherwise it is not hard to see by Lemma 12.4.10 that (A5) will apply and break the cycle), which means that $\Delta^C_\beta(p)$ receives a fresh axiom between each t_n and t_{n+1}. This is contrary to the fact that $f(p)$ reaches a limit. □

Lemma 12.4.14. *For each e, the following are true:*

(i) $\varphi(e)$ is moved finitely often,

(ii) there is a stage after which no module M^α_k for any α and $k \leq e$ is injurious.

Proof. We prove (i) and (ii) simultaneously by induction on e. Let $s_0 > e$ be large enough such that (i) and (ii) holds for all $k < e$, and also the global actions for coding has stopped acting for $\varphi(0), \cdots, \varphi(e)$. That is, the global requirements no longer enumerate $\varphi(0), \cdots, \varphi(e)$. Note that we are not assuming that (i) and (ii) of the global actions hold for e finitely often; instead we only require that the global actions enumerate $\varphi(e)$ into B finitely often. Let α be the leftmost node such that $|\alpha| = e-1$ and α is visited on or after stage e (assume s_0 large enough such that this happens before s_0). Thus the only modules M^β_e which can receive attention after stage s_0, will belong to the nodes $\beta \subseteq \alpha$, because modules to the right of α are never active after stage s_0. We proceed in a series of steps:

Claim 1: If $\beta_0 \subset_f \beta_1 \subseteq \alpha$, then no β_0-module can take an action which initializes $M_e^{\beta_1}$ after stage s_0. The only β_0-module which can do that after s_0 is $M_e^{\beta_0}$, and so if $\beta_0{}^\frown nf \subseteq \beta_1$, then we must have $n \leq e$ (otherwise it is trivial). Let $t_1 > s_0$ be a stage where $M_e^{\beta_0}$ acts and initializes $M_e^{\beta_1}$. Note that we only need to show that there are finitely many such stages t_1 (since we can choose s_0 large enough for the rest of the lemma). Clearly $M_{e+1}^{\beta_1}$ is in the injury set of $M_e^{\beta_0}$ at t_1, and consequently we have $M_{e+1}^{\beta_1}$ is either $successful$ at t_1, or else it has a current traced computation at t_1. In any case we can let $t_0 < t_1$ be the stage where $M_{e+1}^{\beta_1}$ receives attention under (A5), and sets things up for t_1. Note that at t_1, we will also have $M_{e+1}^{\beta_1}$ initialized and every $x \in zone_{e+1}^{\beta_1}$ gets lowered, but the latter does not happen between t_0 and t_1. Therefore, if there are infinitely many such stages t_1, we may assume that t_0 is large (enough for our purpose) as well. At t_0, we have $M_e^{\beta_0}$ is $successful$ (since $M_{e+1}^{\beta_1}$ is active), and hence between t_0 and t_1, the module $M_e^{\beta_0}$ has to be initialized. Since t_0 is large, the module $M_e^{\beta_0}$ has to be initialized by the actions of a third module M_e^{σ} for some σ, which took an injurious action. By Lemma 12.4.12 this is impossible, by the choice of t_0. This contradiction shows that $M_e^{\beta_0}$ has to be $successful$ at t_1 still, and so cannot be injurious towards $M_e^{\beta_1}$.

Note that a β-module can receive attention infinitely often, in which case Ψ_β^C is not total. However, a module cannot be infinitely injurious; this is important because we want each module to have a finite effect on the rest of the construction:

Claim 2: For any β, if M_e^β is initialized finitely often, then it can be injurious only finitely often. This follows directly from Lemma 12.4.13.

Claim 3: For any β such that $\beta \subseteq_\infty \alpha$, we have M_e^β is initialized finitely often. Start with the minimal such node $\beta \subseteq \alpha$, and work downwards inductively by applying Claims 1 and 2.

Claim 4: If $\beta \subset_f \alpha$, then M_e^β is also initialized finitely often. This time round, start with the maximal such node $\beta \subseteq \alpha$, and work upwards inductively by applying Claims 1, 2 and 3.

The claims above prove (ii) for e. We now show (i). Assume $s_1 > s_0$ is such that no module M_k^σ for any σ and $k \leq e$ is injurious anymore. Note that if $\sigma >_L \alpha$, then ag_e^σ is undefined after s_0, and never receives a definition again. If $\sigma <_L \alpha$ then ag_e^σ never receives a new value after s_0. Let $\sigma \subseteq \alpha$, and by Lemma 12.4.13, we have $\lim ag_e^\sigma$ exists. Finally, it is not hard to put the various facts together and conclude that (i) holds for e as well. $\square$

An immediate corollary to Lemma 12.4.14 is that $\Phi^{A \oplus B}$ is total and correctly computes $\emptyset'$. We define the true path of the construction to be the leftmost path visited infinitely often. Before we can show that the true path exists, we start with a preparatory lemma:

Lemma 12.4.15. *Suppose α is visited at s and has just finished acting and has decided to visit the outcome $k_0\infty$. Then, it is impossible for $M_{k_1}^\alpha$ to be waiting if $k_1 > k_0$.*

Proof. Let $s^- \leq s$ be the stage where $M_{k_1}^\alpha$ received attention under (A4) to give its current *waiting* state. Since outcome $k_0\infty$ was played at s, it follows by Fact 12.4.5 that $M_{k_0}^\alpha$ would be able to receive attention at s^- if it required to do so. Hence at s^-, $M_{k_0}^\alpha$ cannot be *unstarted* or *successful*, and by Lemma 12.4.10(ii) we have $\Delta_\alpha^C(p_{k_0}^\alpha)[s^-] \downarrow$. Hence when $\delta_\alpha(p_{k_1}^\alpha)$ was set at s^-, it must be larger than $\delta_\alpha(p_{k_0}^\alpha)[s^-]$. Since $M_{k_0}^\alpha$ is *ready* at s it follows that C must change below $\delta_\alpha(p_{k_0}^\alpha) < \delta_\alpha(p_{k_1}^\alpha)$ between s^- and s, a contradiction. $\square$

Note that in the above lemma, we do not actually need α to have outcome $k_0\infty$ at s. All we really require is that $k_0 > s^-$ (the previous visit to α) and $M_{k_0}^\alpha$ is active and *ready* at s.

Lemma 12.4.16. *The true path of construction exists.*

Proof. Suppose we have defined the true path up to α. Hence α is visited infinitely often, and $\delta_t <_L \alpha$ for finitely many t. We want to show that some outcome of α is visited infinitely often. Suppose each outcome is visited finitely often; we want to derive a contradiction to the fact that C is low$_2$. Since α is initialized only finitely often, it follows by Lemma 12.4.14 that each α-module is initialized finitely often. We first show the following:

Claim 1: For almost all k, M_k^α eventually becomes active and is active at every α-stage after that. Fix k large enough. We may assume by induction hypothesis, that the statement of the claim holds for any $\beta \subset \alpha$. Let s_0 be large enough such that all parameters for M_j^β for $\beta \subset \alpha$ and $j \leq k$ have settled. We can do this because each β-module is initialized finitely often, and by Lemma 12.4.13, we know that ag_j^β is never refreshed again. We want to show that (1) to (4) of Definition 12.4.3 holds for M_k^α forever after s_0. (1) and (4) are obvious, so we consider the other two. First consider some $\beta^\frown nf \subseteq \alpha$.

If M_n^β receives attention infinitely often, then it is not hard to see that eventually we must have M_n^β only receiving attention under (A4). Note that no β-module smaller than M_n^β can receive attention, so only M_n^β can be responsible for enumerating $\Delta_\beta^C(y)$-axioms for $y \leq p_n^\beta$. Therefore there must be some $x \in zone_n^\beta$ (which would have settled) such that $\Psi_\beta^B(x)$ is divergent (because otherwise the variables z and m in (A4) would eventually settle). Consequently we must visit β at some stage t in which $\Psi_\beta^B(x)[t]$ has not persisted for two β-stages, and by Lemma 12.4.9 M_n^β would have to be *ready* at t. It follows that (A2) will apply to M_n^β at t, a contradiction (to the fact that βf is the true outcome of β) follows by applying the induction hypothesis to β. Therefore, M_n^β only receives attention finitely often, and it follows that eventually, at each time $\beta^\frown nf$ is visited, some module M_j^β for $j \geq n$ would have to receive a state change from *waiting* to *successful*, and subsequently stays *successful* forever (assuming of course, that $j \leq k$). If we wait long enough then $l_n^\beta > k$ at every visit to α.

Now consider some $\beta^\frown n\infty \subseteq \alpha$. We want to show that eventually, every module $M_n^\beta, \cdots, M_k^\beta$ is either *successful* or is set correctly at every α-stage. Any β-module M_j^β for $j \leq k$ which receives attention at an α-stage, can only have done so if (A2) applies (use Lemma 12.4.15 for this, and the fact that ag_j^β has settled). If every module M_j^β for $n \leq j \leq k$ receives attention at finitely many α-stages, then by Lemma 12.4.8, we clearly have what we want. Therefore, we may assume that j is the least such that M_j^β receives attention at infinitely many α-stages for some $n \leq j \leq k$. Take a large enough α-stage t where M_j^β receives attention.

Let $t^- < t$ be the previous visit to β, and $t' \leq t^-$ be the largest β-stage where we finished β's action with M_j^β in state *waiting*. Furthermore, t' and t^- satisfies the conditions described in (A2) of the construction. Hence at stage $t^* \leq t'$ when we bestowed the state *waiting* upon M_j^β, we had set $\delta_\beta(p_j^\beta) = \gamma(z)+m$ where z, m are as defined in the construction. We may assume $t^* > s_0$. Since no module smaller than M^β_j can be responsible for setting Δ_β-axioms anymore, it follows by an argument similar as above, that m will reach a limit. Hence at stage t^*, there must be some $x \in zone^\beta_j[t^*]$ such that $\varphi_B(k+1)[t^*] \downarrow < \psi_\beta(x)[t^*]$. Since the computation $\Psi^B_\beta(x)[t^*]$ must be M^β_j-believable at t^*, we can also deduce that every module $M^\beta_n, \cdots, M^\beta_k$ is either *successful* or is set correctly. This concludes the proof of Claim 1.

By Claim 1 it follows that $p_n = \lim p^\alpha_n$ is defined for almost all n. Note that each α-module is initialized only finitely often, and receives attention finitely often (by assumption). If Ψ^B_α is not total then one can verify using Lemma 12.4.15 that $\alpha^\frown n\infty$ will be visited infinitely often for some n, a contradiction. On the other hand suppose that Ψ^B_α is total. If n is large enough, what is the final state of the module M^α_n? Clearly it cannot be *unstarted*, and cannot be *ready* since Ψ^B_α is total. If it is *waiting* for infinitely many n, then $\Delta^C_\alpha(p_n)$ will be defined forever at infinitely many n. Consequently Δ^C_α is defined on almost all inputs, which implies (by low$_2$-ness) that $f(p_n) > \Delta^C_\alpha(p_n)$ for almost all n. This in turn implies that for all large enough n, M^β_n has to become *successful* and stay in that state forever. Hence some finitary outcome of β will be visited infinitely often (depending on where the consecutive run of *successful* modules start), contrary to assumption. Therefore, there is a leftmost outcome of α visited infinitely often. □

Lemma 12.4.17. *Along the true path of construction, the requirements succeed.*

Proof. Let α be assigned requirement $\mathcal{R}_e$, along the true path of construction, such that Ψ^B_e is total. We show that the version of $\{T^\alpha_x\}$ built after the final initialization of α at s_0 traces $\Psi^B_e(x)$ correctly for almost all x.

Claim 1: For any $s \geq s_0$ and every x, we have $x + 1 - |T^\alpha_x| \geq t^\alpha_x$. Fix two stages $s_2 > s_1 > s_0$ such that two different elements are enumerated into T^α_x at s_1 and s_2. We will be done if we can show that t^α_x is decreased between s_1 and s_2. At s_1 it must be the case that M^α_m receives attention under (A5), enumerating $\Psi^B_\alpha(x)[s_1]$ into T^α_x where $m = t^\alpha_x[s_1]$. By Lemma 12.4.11, the change in $B\restriction\psi_\alpha(x)[s_1]$ has to be due to some $\varphi_B(z)$ entering B for some $z < m$. By Lemma 12.4.12, t^α_x is certainly lowered, which proves the claim.

From Claim 1 it follows that $|T^\alpha_x| \leq x + 1$, because $zone^\alpha_0$ is never lowered. It remains to show that almost every $\Psi_e^B(x)$ is traced. Firstly, we argue that the true outcome of α cannot be infinitary. Suppose not, and the true outcome of α is $n\infty$ for some n. Let s_1 be large enough, so that no module M_j^α is initialized, and $\varphi(j)$ has settled, for every $j \leq n+1$. Thus $zone_j^\alpha$ has settled for all $j \leq n$, and we also assume that $\Psi_e^B(x)[s_1] \downarrow$ on the correct use for every x in the final $\cup_{j\leq n} zone_j^\alpha$. By Claim 1 of Lemma 12.4.16 (applied to $\alpha^\frown n\infty$) we may as well assume that all the $\Psi_e^B(x)$-computations above are M_n^α-believable. Observe that M_n^α cannot remain *ready* forever after s_1, because otherwise (A4) will eventually apply for M_n^α, and M_n^α would have received attention at the next stage where $\alpha^\frown n\infty$ is visited. So, M_n^α eventually becomes *waiting* and enumerates some computation $\Delta_\alpha^C(p_n^\alpha)$, with a use that we can assume does not change anymore. This is a contradiction because M_n^α has to get back to state *ready* in time for the next $\alpha^\frown n\infty$ visit.

Thus we let the true outcome of α be nf for some n. Hence $l_n^\alpha \to \infty$, because $M_j^{\alpha^\frown nf}$ eventually becomes active for all large enough $j > n$, following from Claim 1 of Lemma 12.4.16. Again wait for a stage s_1 large enough so that all relevant activity in $M_0^\alpha, \cdots, M_{n+1}^\alpha$ has ceased. Let $x_0 = \min \cup_{j\geq n+1} zone_j^\alpha[s_1]$. We claim that $q = \Psi_e^B(x)$ is traced in T_x^α for all $x > x_0$. Note that $t_x^\alpha \geq n+1$ after s_1, so let $m \geq n+1$ be the final value attained by t_x^α. Let s be the time where an action was taken to make $t_x^\alpha = m$ (note that s may be smaller than s_1). If the action was an initialization to α, then M_m^α is initialized as well. Otherwise the action was to lower t_x^α from $m+1$ to m. By Fact 12.4.7 we know that M_m^β will have to become $successful$ after stage s, which means that at some point $s' > s$, we have (A5) applies for M_m^β and $\Psi_\alpha^B(x)[s']$ enumerated into T_x^α. By Lemmas 12.4.11 and 12.4.12, we have $\Psi_\alpha^B(x)[s'] = p$. $\square$

Chapter 13

Maximal contiguous degrees

13.1 Introduction

Various authors have demonstrated how strong reducibilities may be used to obtain certain results about the structure of the c.e. Turing degrees. Namely, they showed that certain results about wtt degrees can be "transferred" to results about Turing degrees, using the key notion of *contiguous degrees*. A Turing degree $\boldsymbol{a}$ is contiguous if for every c.e. $A, B \in \boldsymbol{a}$, $A \equiv_{wtt} B$. That is, $\boldsymbol{a}$ contains only a single c.e. wtt-degree. Examples of theorems of this type include Ladner and Sasso's [LS75] result that every noncomputable c.e. Turing degree has a c.e. predecessor with the anti-cupping property (here $\boldsymbol{a}$ has the anti-cupping property if there is c.e. degree $\boldsymbol{b} < \boldsymbol{a}$ such that for every c.e. $\boldsymbol{c} < \boldsymbol{a}$, $\boldsymbol{b} \cup \boldsymbol{c} < \boldsymbol{a}$). Downey [Dow87] then applied Ladner and Sasso's analysis to show the existence of a c.e. degree with the strong anticupping property.

Unfortunately contiguous degrees are relatively rare; Cohen, and Jockusch have observed (by an index set calculation) that contiguous degrees are low$_2$, while Ambos-Spies and Fejer [ASF88] showed that the contiguous degrees were nowhere dense in the low degrees. In fact Ladner and Sasso [LS75] showed that for any noncomputable c.e. Turing degree $\boldsymbol{a}$, there are c.e. degrees $\boldsymbol{b}, \boldsymbol{c} \leq \boldsymbol{a}$ such that $\boldsymbol{b}$ is noncontiguous and $\boldsymbol{c} > \boldsymbol{0}$ is contiguous.

While considering the application of Ladner and Sasso's result to the global setting, Downey introduced the idea of a strongly contiguous degree; a c.e. degree is *strongly contiguous* if it contains only a single wtt-degree. The existence of a contiguous degree was first demonstrated by Ladner and Sasso [LS75], and later Ambos-Spies [AS84a] gave a direct construction of a contiguous degree by analyzing the exact requirements which were needed to ensure contiguity. Downey observed that these constructions could easily be modified to give a strongly contiguous degree; indeed all the known results about the contiguous degrees were also true of the strongly contiguous degrees. This led to the interesting question, which is still open:

Question 13.1.1. *Is every contiguous degree strongly contiguous?*

Downey and Lempp [DL97] showed that the contiguous degrees were definable in the c.e. degrees. Cholak, Downey and Walk [CDW02] showed that there were infinitely many maximal contiguous degrees; here a contiguous degree is maximal contiguous if no c.e. degree which strictly computes it is contiguous. These two results gave the first example of a definable infinite antichain in the c.e. degrees.

Our interest in this chapter is along similar lines. We are interested in ultimately finding a natural class of c.e. degrees which are definable (in the c.e. degrees), and which are simultaneously nonhigh and nonlow$_2$. Our immediate aim in this chapter is more modest. We show in Theorem 13.2.1 that the complete c.e. Turing degree is the join of two c.e. degrees which are maximal contiguous. We remark that one can then combine high permitting with the construction given in Theorem 13.2.1, to show that every *high* c.e. degree is the join of two (maximal) contiguous degrees. A similar technique has already been used in Theorem 10.2.1, in the case of the array computable degrees.

To get at our desired property, we can then consider in conjunction, a definable property which excludes being low$_2$, such as bounding a Slaman triple. Our task would be complete if we could show the following

Conjecture 13.1.2. *There a c.e. degree which bounds a Slaman triple, but is not the join of two (maximal) contiguous degrees.*

The general strategy to construct a c.e. degree which avoids being the join of two smaller c.e. degrees of low computational complexity has already been explored in Theorems 10.1.1 and 10.3.1.

13.2 $\mathbf{0}'$ is the join of two maximal contiguous degrees

In this section we show how to code an arbitrary c.e. set into the join of two (maximal) contiguous degrees.

Theorem 13.2.1. *There are c.e. sets A and B, such that the Turing degrees of A and B are maximal contiguous, and $A \oplus B \geq_T \emptyset'$.*

13.2.1 The requirements

We will construct computably enumerable sets A, B, and a Turing functional Γ, satisfying $\Gamma(A \oplus B) = \emptyset'$. The requirements are divided into three categories; firstly we have a global requirement ensuring the correctness and totality of the reduction $\emptyset' = \Gamma(A \oplus B)$. We also have the contiguity requirements $\mathcal{N}_0^A, \mathcal{N}_0^B, \mathcal{N}_1^A, \cdots$. Lastly we have maximality requirements labelled $\mathcal{Q}_0^A$, $\mathcal{Q}_0^B$, $\mathcal{Q}_1^A, \cdots$, which are further divided into subrequirements with labels of the form $\mathcal{Q}_{e,i}^X$. More specifically, the requirements aim to achieve

$$\begin{aligned}
\mathcal{N}_e^A &: \Phi_e(A) \text{ total } \wedge \Psi_e(\Phi_e(A)) = A \Rightarrow A \equiv_{wtt} \Phi_e(A), \\
\mathcal{Q}_e^A &: \text{Build } Q \equiv_T A \oplus W_e, \\
\mathcal{Q}_{e,i}^A &: \hat{\Delta}_i(Q) \neq A \vee W_e \leq_T A.
\end{aligned}$$

The requirements $\mathcal{N}_e^B$, $\mathcal{Q}_e^B$, and $\mathcal{Q}_{e,i}^B$ are as above, with B in place of A. We fix an effective list $\{\hat{\Delta}_i\}$ of all the possible candidates for wtt-functionals. We denote the computable use of $\hat{\Delta}_i$ by $\hat{\delta}_i$. To implement the maximal contiguous strategy, we need build an auxiliary set $Q \equiv_T A \oplus W_e$ via the "kick set procedure", and ensure that $A \not\leq_{wtt} Q$. We also have $\langle \Psi_e, \Phi_e \rangle$ as a list of all pairs of Turing functionals, and W_e is the e^{th} c.e. set.

We fix a computable enumeration $\cup_s \emptyset'_s$ of $\emptyset'$. If P is a parameter, then $P[s]$ or sometimes P_s refers to the value of the parameter P at the instance (within a stage s) that they are mentioned, and when the context is clear, we drop the mention of s. All parameter values will remain in force until it is cancelled or reassigned.

13.2.2 The strategy for ensuring contiguity

The usual strategy for building a contiguous set is known as the confirmation and cancellation strategy; this appears in Ladner and Sasso [LS75], as well as Downey [Dow87]. We will discuss it here for the benefit of the reader. Suppose for the moment that no coding (of the Halting problem) was required, and we are only thinking about building a single set A. The key point behind this strategy is that whenever a number z is enumerated into A at a stage s, we ensure that

(K.1) all numbers $y > z$ that are picked as followers (which we should think of as the coding markers $\gamma(e, s)$ in the following construction) must be immediately cancelled and new followers are to picked from numbers larger than s,

(K.2) at the next $\mathcal{N}_e$-expansionary stage $t > s$, all new followers $y > z$ that were picked in the meantime (between stages s and t) must also be cancelled.

(K.3) Consider now a single contiguity requirement $\mathcal{N}_e$. $\mathcal{N}_e$ will try to control the numbers that enter A by refining them into a stream (for simplicity, we drop the index e). It picks a number x_0 and waits for the length of agreement $l(\Psi(\Phi(A)), A)$ between $\Psi(\Phi(A))$ and A to be larger then x_0. Once that happens at some stage s_0, it will clear the interval $(x_0, s_0]$ by cancelling all numbers in the interval which had been picked as followers in the meantime. This action, together with the fact that all new numbers to be picked as followers are fresh, ensures that $A_{s_0} \restriction (x_0, s_0] = A \restriction (x_0, s_0]$. We say that the number x_0 is *e-confirmed* at stage s_0.

Next, $\mathcal{N}_e$ would pick another $x_1 > x_0$ and wait for an appropriate stage $s_1 > s_0$ to e-confirm x_1. The stream of confirmed numbers is $x_0 < x_1 < x_2 < \cdots$, with each $x_{i+1} > s_i >$ the use of any computation observed at the time where x_i is confirmed. The point of having this stream of e-confirmed numbers, is so that any enumeration into A made by a requirement believing that $A = \Psi(\Phi(A))$, will have to be an e-confirmed number.

To see that the strategy above makes $A \leq_{wtt} \Phi^A$, we wait for a number x_0 to become confirmed (say at the expansionary stage s_0). We ask if $\Phi^A[s_0]\restriction s_0 = \Phi^A \restriction s_0$. If they are not equal, we wait for the first stage $t > s_0$ such that $\Phi^A[t]\restriction s_0 \neq \Phi^A[s_0]\restriction s_0$, and by (K.3) this has to be due to a number $z \leq x_0$ entering A. Hence either $z = x_0$ or else by (K.1) this action cancels x_0 as a follower. If on the other hand, we have $\Phi^A[s_0] \upharpoonright s_0 = \Phi^A \upharpoonright s_0$, then clearly $x \notin A$. Note that we are not requiring that $\Phi^A[s_0]\upharpoonright s_0 = \Phi^A[t]\upharpoonright s_0$ for every $t \geq s_0$, as would be the case if Φ^A was c.e.

Next, to see that $\Phi^A \leq_{wtt} A$, we again let s_0 be the stage where x_0 is confirmed. We claim that $\Phi^A\upharpoonright \varphi(x_0)[s_0]$ can be computed from $A\upharpoonright s_0$. We ask if $A[s_0]\upharpoonright s_0 = A\upharpoonright s_0$. If they are equal, then we are good. If not, then there is some $y < s_0$ that enters A after s_0, and at the next expansionary stage s_1, we have $A_{s_1}\upharpoonright [y, s_1] = A\upharpoonright [y, s_1]$ by (K.2). We repeat by asking if $A[s_1]\upharpoonright y = A\upharpoonright y$, and if not we wait for a change below y again. After finitely many queries, we will get a positive answer. Observe that we can relax the rule (K.2) a little; in particular we only need to cancel all followers larger than y at the next expansionary stage, provided $y < s_0$.

The strategy $\mathcal{N}_e$ would have the two usual outcomes : an infinitary and a finite outcome. The strategy at $\mathcal{N}_e$ will confirm numbers (K.3) and cancel new followers (K.2) at $\mathcal{N}_e$-expansionary stages. The different contiguity requirements $\mathcal{N}_0, \mathcal{N}_1, \cdots$ are arranged according to priority, with $\mathcal{N}_0$ at the highest priority. $\mathcal{N}_1$ would then confirm the least $\mathcal{N}_0$-confirmed number x that has not yet been $\mathcal{N}_1$-confirmed, and cancel all numbers larger than x (including those that have received prior $\mathcal{N}_0$-confirmation). Every number will receive confirmation from at most finitely many requirements.

The above strategy allows us to build a contiguous set A and make the $\mathcal{N}_e$'s compatible with weak positive requirements. For instance, we could clearly make A noncomputable, but we would not be able to make A nonlow$_2$. In the following construction we have contiguity requirements for both A and B, as well as the coding of $\emptyset'$ into $A \oplus B$. The coding is achieved via movable markers $\{\gamma(e)[s]\}$, where we think of $\gamma(e)[s]$ as the current use of the reduction Γ. We have to ensure that for every e, $\lim_s \gamma(e)[s]$ exists, and if we want to move a marker ($\gamma(e)[s+1] > \gamma(e)[s]$), we have to ensure that this is always accompanied by a change in A or B below $\gamma(e)[s]$. This global requirement is positive because each time some number k enters $\emptyset'[s]$, we must enumerate $\gamma(k)[s]$ into either A or B (unless $\gamma(k)$ has not yet been defined). If we were only trying to make A and B contiguous, then it not matter which side we put $\gamma(k)[s]$ into; however as we will see, we have to choose between

A and B carefully if additionally we also wanted to make both A and B maximal contiguous.

If coding was the only action affecting γ, then clearly $\lim_s \gamma(e)[s] < \infty$ for every e. However, $\gamma(e)[s]$ might enter A or B due to the contiguity requirements. Therefore, it is important that the action of the contiguity requirements do not drive $\lim_s \gamma(e)[s] = \infty$. More specifically, at each say, $\mathcal{N}^A$-expansionary stage we would need to cancel all of $\gamma(e)[s], \gamma(e+1)[s], \cdots$ for some e, and move them to values larger than s. The only way we can do this, is to enumerate $\gamma(e)[s]$ into the other side B. This creates a slightly different situation; now the contiguity requirements would have to actively make enumerations (into the other side) in order to disengage marker values from below the use.

Suppose α was working for some A-contiguity requirement, and β believing in $\alpha^\frown\infty$ was working for some B-contiguity requirement. We want to set things up so that the combined actions of α and β do not drive some $\gamma(e)[s]$ to ∞. A typical scenario is the following. We have several α-expansionary stages (which are not β-expansionary), and we reach a stage s such that $\gamma(0)[s], \gamma(1)[s], \cdots, \gamma(e)[s]$ are all α-confirmed. At the next β-expansionary stage $s_1 > s$, we will have $\gamma(0)[s]$ receiving β-confirmation; this same action also causes $\gamma(1)[s], \cdots, \gamma(e)[s]$ to be enumerated into A at stage s_1. At the next α-expansionary stage s_2 after s_1, α will need to disengage $\gamma(1)[s_2], \cdots, \gamma(s_2)[s_2]$ (to ensure (K.2)), by dumping them into B. Fortunately, we have $\gamma(1)[s_2] > s_1$, which means that at the next β-expansionary stage after s_2, (by the observation that we can relax rule (K.2)), β will not need to disengage $\gamma(1)$. For an arbitrary marker $\gamma(e)$, this chain of action along any path $\alpha_0 \subset \alpha_1 \cdots \subset \alpha_e$ where $|\alpha_e| = e$ will inductively come to a halt. Indeed when α_e disengages $\gamma(e)$ at some α_e-expansionary stage, it will never need to do so again.

13.2.3 Maximal contiguous strategy

We now discuss how to incorporate the maximal contiguous strategy into the proof. The strategy for constructing a maximal contiguous degree can be found in [CDW02]. Since this is not well-known, we will describe it here. We will assume, for simplicity, that there are no coding markers present for the time being. Consider the $\mathcal{Q}_e^A$-node τ attempting to make $A \oplus W_e$ noncontiguous. The mother node τ divides its task amongst infinitely many sub-nodes σ assigned requirements of the form $\mathcal{Q}^A_{e,i}$. τ is responsible for building a set $Q \equiv_T A \oplus W_e$, while each $\mathcal{Q}^A_{e,i}$-node will try to get a disagreement between Q and $\hat{\Delta}_i$.

The τ builds Q using the following procedure. Again we maintain a sequence of coding markers $\{\lambda(n)[s]\}$ targeted at Q. We start with $\lambda(n)[0] = n$ for all n. At a stage s, if the number n enters $A \oplus W_e$, we will enumerate $\lambda(n)[s]$ into Q, and move all the markers $\lambda(n)[s+1], \lambda(n+1)[s+1], \cdots$ to values larger than the use of any wtt-reduction $\hat{\Delta}_i(Q)$ observed so far. This ensures that $A \oplus W_e \leq_T Q$ in the usual way. To simplify the presentation of the proof, we do not code Q directly into $A \oplus W_e$; instead we will ensure that these λ-marker values are the only numbers enumerated into Q during the construction. To now compute if p is in Q, see if $p = \lambda(n)[p]$ for some n at stage p. We ask if $n \in A \oplus W_e$. If not, then $p \notin Q$, otherwise wait for n to enter $A \oplus W_e$ and we will have either $p \in Q$ by then, or else it would have been canceled as a marker.

We now describe how to meet a single $\mathcal{Q}^A_{e,i}$. We pick a follower $\nu \notin A_s$ and attempt to get a disagreement $\hat{\Delta}^Q_i(\nu) \neq A(\nu)$, by the usual Friedberg-style strategy. There are two problems with pursuing the plain strategy. Firstly, when we put ν into A to diagonalize, we will also need to leave a trace in Q, which might immediately destroy the diagonalization. Secondly, $Q \upharpoonright \hat{\delta}_i(\nu)$ might also cause changes in W_e, which is entirely out of our control.

To solve the first problem, we could do the following. We pick two followers $n < \nu$, and wait for $l(\hat{\Delta}_i(Q), A) > \nu$. We then disengage the $\lambda(\nu)$ marker from below the $\hat{\delta}_i(\nu)$-use by first enumerating n into A, and upon recovery of $l(\hat{\Delta}_i(Q), A)$ we enumerate ν into A for the actual diagonalization. This is a reasonable strategy, but unfortunately the first enumeration of n into A will also cancel ν as a follower for the sake of the contiguity requirements. What we will do instead, is to get W_e to help us disengage the $\lambda(\nu)$-marker. Namely, we wait for $W_e \upharpoonright n$ change (and hence we can lift $\lambda(\nu)$ above $\hat{\delta}_i(\nu)$). Only then do we diagonalize by enumerating ν into A at the next recovery stage. We could pick infinitely many pairs of $\langle n, \nu \rangle$, and by the fact that $W_e >_T \emptyset$ (in fact we have $W_e \not\leq_T A$), one of the attempts will succeed.

The resulting disagreement (in the case of a successful attack), would be preserved forever if not for the second problem, i.e. $W_e \upharpoonright n$ might change a second time (after a successful diagonalization), forcing us to change Q below $\hat{\delta}_i(\nu)$. To get around this, we could prepare ourselves beforehand by preparing n followers $x_0 < x_1 < \cdots < x_{n-1}$ all of which are larger than n. The strategy now becomes clear:

(Step 1) Wait for $l(\hat{\Delta}_i(Q), A) > \max\{x_0, \cdots, x_{n-1}\}$.

(Step 2) Wait for a change in $W_e \restriction n$. Lift the use $\lambda(n)$ above $\hat{\delta}_i(x_{n-1})$. Restraint A below n.

(Step 3) Let ν be one of the followers $x_0, \cdots, x_{n-1}$ not yet used. At the next stage where $l(\hat{\Delta}_i(Q), A) > \nu$, we can enumerate ν into A.

(Step 4) Monitor $W_e \restriction n$ for changes. Should W_e change again below n we could repeat steps 3-4 using another of the candidates for ν.

The crucial point is that after step 2, $\lambda(n)$ would have been disengaged from below $\hat{\delta}_i(x_{n-1})$. Hence, any failure to preserve a successful diagonalization would have to be due to a $W_e \restriction n$-change, and we will not run out of followers to use for attacks. Since we have to also ensure that the contiguity requirements are met, we will have to enumerate the followers $x_{n-1}, \cdots, x_0$ in that order (of decreasing magnitude).

$\mathcal{Q}^A_{e,i}$ will have infinitely many modules, with the k^{th} module waiting for a $W_e \restriction k+1$ change before proceeding with the attacks. It might seem that $W_e >_T \emptyset$ is sufficient, which we know of course is impossible. After each $\mathcal{Q}^A_{e,i}$-module M is started, and has picked followers $x_0, \cdots, x_{n-1}$ aimed at A, it might be the case that some number $y < \max\{x_0, \cdots, x_{n-1}\}$ might be enumerated into A due to other requirements. This would also make some of the numbers $x_0, \cdots, x_{n-1}$ unusable. In this case, $\mathcal{Q}^A_{e,i}$ has to start the module M all over again from step 1; we call this an initialization to the module M. If every module M gets stuck in step 2 after its final initialization, then we would have to use A to help us find a stage after which M is never initialized, and then compute an appropriate segment of W.

The above describes the strategy of $\mathcal{Q}^A_{e,i}$ when there are no coding markers present. In the following construction, a module M will start in step 1 (at a stage s) by choosing followers to be current marker values $x_0 = \gamma(x, s), \cdots, x_n = \gamma(x + n, s)$ for some x. If a number $x < x_n$ enters A, the module will be initialized, but occasionally a small number y (say $y < x_0$) will enter B and make $\gamma(x + i, t + 1) > \gamma(x + i, s)$

for all $i = 0, \cdots, n$. In this case however, the module M will not (and in fact, cannot) be initialized, since there had been no A-change. Therefore, it is possible for numbers to be enumerated during the construction into A or B, which are not current marker values. This was different from when we only had the contiguity requirement, and some care has to be taken to ensure this synchronizes with the contiguity requirements. However, such enumerations (of numbers which are not current marker values) will have to be made by some noncontiguity requirement β (say β is an A-noncontiguity requirement). Note that β begins a module by picking current marker values as attackers, which all have to be already α-confirmed for every contiguity node $\alpha^\frown\infty \subseteq \beta$. Therefore, the only action which can possibly make a B-enumeration below β-attackers is the global coding of $\emptyset'$ into B. At the next α-expansionary stage, we do not have to cancel the β-attackers, because either α is an A-contiguity node (in which case it does nothing since there was no A-change), or else α is a B-contiguity node (in which case it will only need to cancel attackers targeted for B).

There is yet another technical point to consider. The X-noncontiguity requirement β will impose a restraint r on X when one of its modules is in step 3. If each module only gets initialized finitely often, then this total restraint is finite (by the fact that W has to change). The usual tree mechanism will normally take care of this; however in this construction there is a global coding action performing its task in the background. Thus, before the coding strategy makes an enumeration x, it will look at the active noncontiguity requirement β of the highest priority, whose restraint is larger than x. We then enumerate x into the other side.

13.2.4 The Construction Tree and Notations Used

The construction takes place on a subtree of the binary tree. Nodes α of length $4e$ and $4e+1$ are assigned to the requirements $\mathcal{N}_e^A$ and $\mathcal{N}_e^B$ respectively, equipped with the outcomes ∞ and fin. Nodes of length $4\langle e, 0\rangle+2$ and $4\langle e, 0\rangle+3$ are assigned to $\mathcal{Q}_e^A$ and $\mathcal{Q}_e^B$, with only a single outcome 0. Lastly, nodes of length $4\langle e, i\rangle+2$ and $4\langle e, i\rangle+3$ for $i > 0$ are assigned to the requirements $\mathcal{Q}_{e,i}^A$ and $\mathcal{Q}_{e,i}^B$, guessing the outcomes d (for diagonalizing) to the left of w (for waiting). We will use the expression "α is a $\mathcal{P}$-node" to mean that α is assigned the requirement $\mathcal{P}$. The ordering amongst the outcomes induces an ordering amongst nodes on the construction in the usual lexicographic way; we say $\alpha <_L \beta$ if α is strictly to the left of β, and we use $\subset$ for strict string extension. We say that β is of *higher α-priority* if $\beta \subset \alpha$ or $\beta <_L \alpha$. Further terminology is stated below. If a definition regarding an A-requirement is made, it is understood that the definition extends to the corresponding B-requirements in the obvious way :

If α is a $\mathcal{Q}^A_{e,i}$-node, define $\tau(\alpha)$ to be the least $\beta \subset \alpha$ such that β is a $\mathcal{Q}^A_e$-node. This is known as the *mother node of* α. At each mother $\mathcal{Q}^A_e$-node τ, we will build a c.e. set Q_τ. We will also maintain a set of markers $\{\lambda_\tau(n,s)\}$, used in making $Q_\tau \equiv_T A \oplus W_e$, as described in the previous section. For each $\mathcal{Q}^A_{e,i}$-node α, define the length of agreement

$$l(\alpha, s) := \max\{x < s \mid (\forall y \leq x)\ \hat{\Delta}_i(Q_{\tau(\alpha)}; y)[s] = A(y)[s]\},$$

and let $r_\alpha[s]$ record the restraint function (on both A *and* B) imposed by α at stage s. Each $\mathcal{Q}^A_{e,i}$-node α maintains infinitely many modules. We will refer to the n^{th} module ($n > 0$) of the node α as the $\langle \alpha, n\rangle$-module. In addition we will define, for each $\langle \alpha, n\rangle$-module, the parameters $L_{\alpha,n}, F_{\alpha,n}, \eta_{\alpha,n}$, $\nu_{\alpha,n}$, and $\xi_{\alpha,n}$ as follows. The introduction of these parameters will make later verification clearer.

- We let $L_{\alpha,n}[s]$ denote the set of numbers for which $\{\gamma(k,s) \mid k \in L_{\alpha,n}[s]\}$ forms the set of followers for the diagonalization attempts. At all times we ensure that $|L_{\alpha,n}[s]| = n+1$, every element $x \in L_{\alpha,n}[s]$ is larger than $\max\{|\alpha|, n, s\}$, and if any $\langle \beta, m\rangle$-module is started earlier than this module, then $\max L_{\beta,m}[s] < \min L_{\alpha,n}[s]$.

- We introduce $F_{\alpha,n}[s]$ to represent the current state of the $\langle \alpha, n\rangle$-module. It may take values $0, 1, 2, 3$, or 4, with 0 representing the fact that the module is unstarted, and $1 - 4$ corresponding to the steps $1 - 4$ in the basic strategy.

- The next marker in $L_{\alpha,n}$, whose value is to be used in the (next) diagonalization attempt is denoted by $\eta_{\alpha,n}[s]$.

- $\nu_{\alpha,n}[s]$ denotes the current number that is used in the diagonalization.

- $\xi_{\alpha,n}[s]$ records the stage $t \leq s$ when the module first fixed the followers $\{\gamma(k,t) \mid k \in L_{\alpha,n}[t]\}$. This is useful because an attacker can initially be picked as a current marker value, but later the marker position is moved.

- A number x is said to be a current follower of the $\langle\alpha, n\rangle$-module (at stage s), if it is chosen by the module as one of the numbers to be used in the diagonalization attempts, i.e. $F_{\alpha,n}[s] > 1$ and $x = \gamma(y, \xi_{\alpha,n}[s])$ for some $y \in L_{\alpha,n}[s]$ and $y \leq \eta_{\alpha,n}[s]$.

In order to give a clearer picture on how the notations are to be used, we describe here the actions a typical module might take. It starts by picking $L_{\alpha,n} = \{3,4,5\}$, $\nu_{\alpha,n} = \xi_{\alpha,n} = 0$, $\eta_{\alpha,n} = 5$, and starts in state $F_{\alpha,n} = 1$. It then waits for a stage s_0 such that $l(\alpha, s_0) > \max\{\gamma(3,s_0), \gamma(4,s_0), \gamma(5,s_0)\}$. At stage s_0 it will decide to pick the numbers $\gamma(3,s_0), \gamma(4,s_0), \gamma(5,s_0)$ as (current) followers and sets $\xi_{\alpha,n} = s_0$ as record of this fact. It then moves to step 2 by setting $F_{\alpha,n} = 2$, and waits for a $W_e\lceil n+1$ change. Once that happens it moves to step 3 and waits for a stage s_1 such that $l(\alpha, s_1) > \gamma(\eta, \xi) = \gamma(5, s_0)$. It then proceeds to put $\gamma(5, s_0)$ into A and sets $\nu_{\alpha,n} = \gamma(5, s_0)$ to keep a record. Then, it sets $\eta_{\alpha,n}$ to point at the next coding marker to be used, in this case 4. It then moves to step 4 and waits for $W_e\lceil n+1$ change to produce a recovery. When that happens at stage s_2 (i.e. $l(\alpha, s_2) > \nu = \gamma(5, s_0)$), we will return to step 3 and wait for a stage s_3 such that $l(\alpha, s_3) > \gamma(\eta, \xi) = \gamma(4, s_0)$. The process runs as described in the basic strategy, except when a number $x \leq \gamma(\eta, \xi)$ enters A, which would initialize the module (see the next paragraph). In this case we will set the state $F_{\alpha,n}$ back to 1 and $\eta_{\alpha,n}$ back to point at 5.

In addition, a $\mathcal{Q}^X_{e,i}$-node α (for some e, i, and $X \in \{A, B\}$) is called an X-*noncontiguity node*. When we initialize the $\langle\alpha, n\rangle$-module at stage s, we mean to say that we set $\nu_{\alpha,n} = \xi_{\alpha,n} = 0$, and if $F_{\alpha,n} > 0$ we will also set $F_{\alpha,n} = 1$, and $\eta_{\alpha,n} = \max L_{\alpha,n}$. The meaning of initializing a node α is the following. If α is a noncontiguity node, we set $F_{\alpha,n} = 0$ for every n. For all other types of nodes, we simply keep a record, and do nothing else.

$\mathcal{N}^A_e$-nodes are called A-*contiguity* nodes (similarly for B-contiguity nodes). For each contiguity node α, we say that a number x is *ready to be α-confirmed* at stage s, if $x = \gamma(i, s)\downarrow$ for some $i \geq |\alpha|$, x is not yet α-confirmed, and $x > \max\{r_\beta[s] \mid \beta$

is of higher α-priority}. For an $\mathcal{N}_e^A$-node α and a stage s (the same goes for the B-contiguity nodes), we say that s is α-expansionary, if

(i) α is visited at stage s,

(ii) $l_A(e,s) > \max\{l_A(e,t) \mid t < s \ \wedge \ \delta_t \supseteq \alpha^\frown\infty\}$, and

(iii) there is some (least) x which is ready to be α-confirmed, and $l_A(e,s) > x$.

Here, $l_A(e,s)$ denotes the nested length of agreement between $\Psi_e(\Phi_e^A)$ and A:

$$l_A(e,s) := \max\{x < s \mid (\forall y < x) \ \ \Psi_e(\Phi_e^A; y)[s] \downarrow = A_s(y)\},$$

with $l_B(e,s)$ defined similarly.

Lastly, we say that a number x is a *current marker value* at stage s, if $x = \gamma(i,s)$ for some i. To *dump a number* x into A (or B), means to enumerate x into A (or B), and to cancel all $\{\gamma(j,s) \mid \gamma(j,s) \geq x\}$. A number x is said to be *reset by the node* α at stage s, if x is a current marker value at stage s, α is visited at stage s, and takes some action which results in y being dumped for some $y \leq x$. If α is visited during the construction at a stage s, then s^- refers to the previous α-expansionary stage if α is a contiguity node, and for all other types of nodes this will refer to the previous α-stage. Whenever the previous stage does not exist, we take s^- to be 0.

13.2.5 Construction

At stage $s = 0$, we set $\lambda_\alpha(n,0) = n$ for all α and n, and we initialize every node. At each stage $s > 0$, we will define the approximation to the true path, δ_s by induction on the substage u. Assume that $\alpha = \delta_s \restriction u$ has been defined. There are four possibilities:

1. *α is an $\mathcal{N}_e^A$ or $\mathcal{N}_e^B$-node*: If s is not α-expansionary, we take no action and let $\delta_s(u) = fin$. Otherwise, let $\delta_s(u) = \infty$ and run the "cancellation and confirmation" strategy:

(1.1) *cancellation*: Suppose that the following conditions are met. Then, take the action described, otherwise do nothing.

(i) $s^- > 0$,

(ii) there is some number $y < s^-$ that entered A between the two visits to α at s^- and s. Let y be the least such.

(iii) α is not initialized between s^- and s.

(iv) let $i \geq |\alpha|$ be the least such that $\gamma(i, s)\downarrow$, $\gamma(i, s) > y$, and $\gamma(i, s) > \max\{r_\beta[s] \mid \beta$ is of higher α-priority$\}$.

Action: Dump $\gamma(i, s)$ into B.

(1.2) *confirmation*: Let x be the smallest number ready to be α-confirmed, where $x = \gamma(i, s)$ for some $i \geq |\alpha|$.
Action: Dump $\gamma(i+1, s)$ (if it is defined) into B, and declare the number x as α-confirmed.

Finally, initialize all nodes $\beta >_L \alpha^\frown\infty$. Similar actions follow for $\mathcal{N}_e^B$.

2. *α is a $\mathcal{Q}_e^A$ or $\mathcal{Q}_e^B$-node*: We describe the action for $\mathcal{Q}_e^A$; again similar actions follow for $\mathcal{Q}_e^B$. Let $\delta_s(u) = 0$. Suppose that there is some least $y \in (A \oplus W_e)[s] \setminus (A \oplus W_e)[s^-]$.
Action: Enumerate $\lambda_\alpha(y, s)$ into Q_α, and set, for each $n > 0$, $\lambda_\alpha(y + n) = n + s + \max Q_{\alpha,s} + \max\{\hat{\delta}_i(z) \mid (\exists t' < s)(\exists \mathcal{Q}_{e,i}^A\text{-node } \beta)\ \tau(\beta) = \alpha\ \wedge\ l(\beta, t') > z\ \wedge\ \beta \text{ is visited at stage } t'\}$.

3. *α is a $\mathcal{Q}_{e,i}^A$ or $\mathcal{Q}_{e,i}^B$-node* : We look at the case $\alpha = \mathcal{Q}_{e,i}^A$, a similar action follows for $\mathcal{Q}_{e,i}^B$. If no α-module is in state $F_{\alpha,n} = 3$ or 4, then we go to (3.1), otherwise go to (3.2).

(3.1) for each of the modules below, let $t < s$ be the previous stage where the module was last run, and $t = 0$ if that does not exist.
Action: For each n such that $0 < n < s$, run the $\langle \alpha, n\rangle$-module by following the steps $(n.1)$ and $(n.2)$ as described below:

(step $n.1$): Suppose $F_{\alpha,n} > 1$, and $\exists x \leq \gamma(\eta_{\alpha,n}, \xi_{\alpha,n})$ such that

∗ either $x \in A_s \setminus A_t$,

∗ or $x \in B_s \setminus B_t$ and x is enumerated under the action of some node β of higher α-priority,

∗ or else $x \in B_s \setminus B_t$ and $x \leq \gamma(y, s)$ for some $y \leq |\alpha|$.

If one of the above applies, we initialize all $\langle \alpha, m\rangle$-modules for $m \geq n$.

(step $n.2$): Choose one of the appropriate actions from the list below:

- $F_{\alpha,n} = 0$: Set $L_{\alpha,n} = \{x, \cdots, x+n\}$, where x is the least number larger than $|\alpha|$, n, s, and $\max L_{\beta,m}[t]$ for any module $\langle \beta, m\rangle$ and $t \leq s$. Set $F_{\alpha,n} = 1$, initialize the $\langle \alpha, n\rangle$-module, and end the step $n.2$.
- $F_{\alpha,n} = 1$: Suppose that for every $x \in L_{\alpha,n}[s]$, $\gamma(x, s) \downarrow$ and is β-confirmed for every contiguity node β such that $\beta^\frown\infty \subseteq \alpha$. Also, we check if $l(\alpha, s) > \gamma(\eta_{\alpha,n}, s)$. If the preceding conditions are all met, set $\xi_{\alpha,n} = s$, $F_{\alpha,n} = 2$, and end the step $n.2$.
- $F_{\alpha,n} = 2$: If $W_{e,t}\restriction n+1 \neq W_{e,s}\restriction n+1$ we set $F_{\alpha,n} = 3$, and end the step $n.2$.

Once all the modules have been run, we go to (3.3).

(3.2) let n be the least such that $F_{\alpha,n} \in \{3, 4\}$. We run the single $\langle \alpha, n\rangle$-module as described below, since it is now ready to begin diagonalization. No other module will be run in this step.

Action: Check if $\exists x \leq \gamma(\eta_{\alpha,n}, \xi_{\alpha,n})$ such that

- either $x \in A_s \setminus A_{s^-}$,
- or $x \in B_s \setminus B_{s^-}$ and x is enumerated under the action of some node β of higher α-priority,
- or else $x \in B_s \setminus B_{s^-}$ and $x \leq \gamma(y, s)$ for some $y \leq |\alpha|$.

If one of the above applies, initialize all the $\langle \alpha, m\rangle$-modules for $m \geq n$. Otherwise, choose one of the appropriate actions below:

- $F_{\alpha,n}[s] = 3$: If $l(\alpha, s) > \gamma(\eta_{\alpha,n}, \xi_{\alpha,n})$ and $\max\{\gamma(j, s) \mid \gamma(j, s)$ is defined$\} > \gamma(\eta_{\alpha,n}, \xi_{\alpha,n})$, we will make an attempt at diagonalization: Set $\nu_{\alpha,n} = \gamma(\eta_{\alpha,n}, \xi_{\alpha,n})$, and dump $\gamma(\eta_{\alpha,n}, \xi_{\alpha,n})$ into A. Also set $\eta_{\alpha,n}[s+1] = \max L_{\alpha,n}\restriction\eta_{\alpha,n}[s]$, set $F_{\alpha,n} = 4$.
- $F_{\alpha,n}[s] = 4$: We will wait for recovery. That is, wait for $l(\alpha, s) > \nu_{\alpha,n}$. If so we set $F_{\alpha,n} = 3$.

Once the actions are finished, go to (3.3).

(3.3) in this step, we decide the outcome. Let $\delta_s(u) = d$ if some $\langle \alpha, n\rangle$-module has now got a state $F_{\alpha,n} \in \{3, 4\}$. If not, then we let $\delta_s(u) = w$. Furthermore, if $\delta_s(u) = d$ and $\delta_{s^-}(u) = w$, we increase the restraint by setting $r_\alpha[s] = 1 + \max\{\gamma(\eta_{\alpha,n}, \xi_{\alpha,n})\} \cup \{r_\alpha[t] \mid t < s\}$ where n is the least $\langle \alpha, n\rangle$-module such that $F_{\alpha,n} = 3$, and initialize all nodes of lower α-priority.

4. *we have reached substage* $u = s$: The final actions to conclude stage s are the following actions for the sake of coding $\emptyset'$.

(4.1) *correction of* Γ: Pick the least e such that $\gamma(e, s) \downarrow$, and $\emptyset'(e)[s] \neq \emptyset'(e)[t]$ where $t < s$ is the stage where $\gamma(e, s)$ received its definition. Let β be the $<_L$-least node such that $\gamma(e, s) \leq r_\beta[s]$.
Action: If β is an A-noncontiguity node, dump $\gamma(e, s)$ into B, otherwise dump $\gamma(e, s)$ into A.

(4.2) *extension of* Γ: Pick the least e such that $\gamma(e, s) \uparrow$. Extend the domain of $\Gamma(A \oplus B)$.
Action: Define $\gamma(e, s)$ to be the smallest number n, such that $n > s$, and $n > \max\{\gamma(i, t) \mid i \in \mathbb{N} \wedge t \leq s\}$. Set $\Gamma^{A\oplus B}(e)[s] \downarrow = \emptyset'(e)[s]$ with use $2\gamma(e, s) + 1$.

Note that if some $\gamma(e, s)$ gets dumped or reset, then $\gamma(e)$ by convention becomes undefined (meaning that it has no value currently assigned to it) until it next receives a definition under (4.2).

13.2.6 Verification

Let TP denote the true path defined as usual, to be the left-most path visited infinitely often. To improve clarity in the verification lemmas, and to clear up any doubts the reader may have about the construction, we give a brief summary.

- Note that the only numbers which can enter A or B at a stage s, are either the current marker values, or the previous values of markers $\gamma(i, t)$ for some i, and $t < s$.

- An A-contiguity node α only enumerates current marker values $\gamma(i, s)$ for some $i \geq |\alpha|$ into B, and vice-versa for B-contiguity nodes.

- Let α be a noncontiguity node. Any number x which is a follower of some $\langle \alpha, n\rangle$-module at stage s, must already be β-confirmed for every contiguity node β such that $\beta^\frown\infty \subseteq \alpha$. Also, no number $\leq x$ can ever be β-confirmed at any time after s (see Lemma 13.2.2). Furthermore, if $\gamma(j, s) < x < \gamma(j+1, s)$ for some j, then $x = \gamma(k, t)$ for some $k > j$ and $t < s$.

- Let α be an X-noncontiguity node. α only makes enumerations into X, and only does so at stages where $\alpha^\frown d$ is played. Furthermore, the only enumerations it makes are current followers of one of its modules.

- The end of stage action (4.1) only enumerates current marker values, for the sake of the correctness of $\Gamma(A \oplus B)$, and puts them into either A or B. The decision as to which side, is made in favour of the highest priority node working on a noncontiguity requirement.

- If α is visited at some stage s, then α will be initialized the next time $\delta_t <_L \alpha$, i.e. we visit left of α.

- Suppose α is an A-noncontiguity node which enumerates x at stage s. There is a stage $t < s$ when α *picked x as a follower*; by this we mean a stage t where α is visited and the relevant α-module sets $\xi_{\alpha,n} = t$ (while in state 1), and also that x is a current marker value at t. Now observe that between t and s, no number $y < x$ can enter B due to a higher α-priority node, nor can $y < x$ enter A.

Lemma 13.2.2. *Suppose a number x has already been α-confirmed, for some contiguity node α at stage s, then no number $y \leq x$ can get α-confirmed after stage s.*

Proof. Suppose there is some $y < x$ which gets α-confirmed at stage $t > s$. Let $u \leq s$ be a stage when α declares x as α-confirmed. We may write $x = \gamma(p, u)$ for some p. Note y must be a current marker value at stage u, so we can also write y as $\gamma(q, u)$ for some $q < p$. Since y is to become α-confirmed at stage $t > u$, we must have $y = \gamma(q, u) = \gamma(q, t)$, and consequently y is larger than the appropriate restraints at stage u, contradicting the fact that x is the least number ready to be confirmed at u. □

Lemma 13.2.3. *For every noncontiguity node α on TP, we have $\tilde{r}_\alpha := \lim_s r_\alpha[s] < \infty$.*

Proof. By induction hypothesis, let $r = 1 + |\alpha| + \max\{\tilde{r}_\beta \mid \beta$ has higher α-priority$\}$, and suppose α is an A-noncontiguity node. Let s be large enough so that the following are stable.

(i) $(\forall t > s)\delta_t \not<_L \alpha$,

(ii) $\emptyset' \restriction r[s] = \emptyset' \restriction r$,

(iii) no noncontiguity node β of higher α-priority enumerates any more elements after stage s,

(iv) r_β has settled for every β of higher α-priority.

It is possible for (iii) to be satisfied; consider some noncontiguity node β with $\beta^\frown d \subseteq \alpha$. After r_β is stable, it is clear that β will never play outcome w anymore. Hence, some least $\langle \beta, n\rangle$-module will remain in state 3 or 4, and so can only enumerate into A or B finitely often.

Now let $t > s$ be a stage such that $\delta_t \supseteq \alpha^\frown d$ (if there is no such stage, then the result holds trivially). We wish to show that r_α is never increased after t, so suppose the contrary. By (i) and (iv), α is never initialized after stage t, and hence the only way that r_α can increase is for there to be some numbers x, n and stage $u > t$ such that $x \leq \gamma(\eta_{\alpha,n}[t], \xi_{\alpha,n}[t]) \leq r_\alpha[t]$ and α is visited again at u, when we have one of the following holds :

(13.2.3.a) $x \in A_u \setminus A_t$,

(13.2.3.b) $x \in B_u \setminus B_t$ and is enumerated by some node β of higher α-priority,

(13.2.3.c) $x \in B_u \setminus B_t$ and $x \leq \gamma(y, u)$ for some $y \leq |\alpha|$.

We assume u is the least such stage, and let x be the least number satisfying (13.2.3.a), (13.2.3.b) or (13.2.3.c). We now analyze which action can be responsible for the enumeration of x. After stage t, no number $\leq r_\alpha[t]$ may be enumerated into A under the end of stage action (4.1), because of (ii) and the fact that α now has the highest priority. Also, (4.1) does not enumerate x into B under the scenario

(13.2.3.c) above. To see this, suppose that it enumerates $\gamma(z, u')$ at the end of stage $u' \in [t, u)$. We must have $z > y$, because otherwise we have $z \leq y \leq |\alpha|$ and $\emptyset'$ is already stable up till $|\alpha|$. Hence, $z > y$ which implies $\gamma(y, u') < \gamma(z, u') = x \leq \gamma(y, u)$, and so some number $p \leq \gamma(y, u')$ must enter either A or B between stages u' and u, contradicting the minimality of x. Hence, the end of stage action (4.1) cannot be responsible for the entry of x.

It is clear also, that no node of a lower α-priority may enumerate an element $\leq r_\alpha[t]$ after stage t. It is also obvious that α itself cannot be responsible since followers are enumerated in decreasing order. Together with (iii), it is evident now that the only possible node responsible for the entry of x, must be some $\mathcal{N}_e^X$-node δ, where $\delta^\frown\infty \subseteq \alpha$ and $X \in \{A, B\}$. Let δ be the first node (wrt time) after stage t that enumerates x, say at stage $t' > t$. Note that x is in fact smaller than some follower of an α-module at stage t, which must already have been δ-confirmed before stage t. By Lemma 13.2.2, no number less than x can receive a δ-confirmation after stage t, and thus δ cannot enumerate x under the action of "Confirmation". Thus, δ has to enumerate x at stage t', under the action of "Cancellation".

If $X = A$, there must be some number $y < x$ such that y enters A between the two visits to δ at stages t and t'. We have $y < x$, and y is enumerated into A after the visit to α at stage t (y could not have entered A before the visit to α at t, otherwise $\alpha^\frown d$ cannot have been visited at stage t). This would contradict the minimality of t' and δ. Hence it must be that $X = B$ and there must be some $y < x \leq r_\alpha[t]$ that is enumerated into B after stage t. By the minimality of t' and δ, we can see that y must be enumerated into B under the action of (4.1). Since δ only enumerates current marker values, it follows that x is picked as a marker after the enumeration of y, and so we must have $x > \gamma(\eta_{\alpha,n}[t], \xi_{\alpha,n}[t])$, a contradiction. □

It now follows that every noncontiguity node on the true path makes only finitely many enumerations into A or B.

Lemma 13.2.4. *At every stage s and for every x, suppose that there is some $\langle\alpha, n\rangle$-module such that $F_{\alpha,n}[s] > 1$, and $\gamma(x-1, s) < \gamma(y, \xi_{\alpha,n}[s]) < \gamma(x, s)$ for some $y \in L_{\alpha,n}[s]$, and $\gamma(y, \xi_{\alpha,n}[s])$ is enumerated at stage s. Then, we must have $|\alpha| < x$.*

Proof. Without loss of generality, assume that α is an A-noncontiguity node. Let

$t < s$ be the largest stage such that $F_{\alpha,n}$ changes state from 1 to 2. Then $L_{\alpha,n}[t] = L_{\alpha,n}[s]$ and $\xi_{\alpha,n}[s] = t$. As in the previous remarks we must have $y \geq x$, and that $\gamma(x,t) \leq \gamma(y,t) < \gamma(x,s)$. Therefore some number $z \leq \gamma(x,t)$ must have entered B between t and s (nothing small can enter A between t and s, otherwise the $\langle \alpha, n\rangle$-module will get initialized). Since z enters B between stages t and s, we cannot have $|\alpha| \geq x$, lest the $\langle \alpha, n\rangle$-module is initialized. □

We now come to an important lemma. We will show that the γ-markers eventually settle, so that the reduction Γ is total. We proceed by induction on e. $\gamma(e)$ may be moved due to one of two reasons: either some number $z < \gamma(e)$ is enumerated, or else $\gamma(e)$ itself is enumerated. We thus split the proof of the lemma in two parts, which argue the two cases separately.

Lemma 13.2.5. *For every e, $\tilde{\gamma}(e) := \lim_s \gamma(e,s) < \infty$.*

Proof. Fix a stage $s > e$ such that the following are stable

(i) $\emptyset' \restriction e + 1[s] = \emptyset' \restriction e + 1$,

(ii) $(\forall t > s)(\forall i < e)(\gamma(i,t) = \tilde{\gamma}(i))$,

(iii) the number $\tilde{\gamma}(e-1)$ is never confirmed (by any node) after stage s,

(iv) let α be the $<_L$-least node in the finite set $\Xi := \{\beta \mid \beta$ is a noncontiguity node, $|\beta| < e$, and β enumerates infinitely many elements$\}$. It is possible for Ξ to be nonempty; in fact by Lemma 13.2.3, Ξ can only contain nodes strictly to the right of the true path. Assume that after stage s, all nodes of length less than e which enumerate only finitely many elements, never do so again.

Claim 13.2.5.1: *We shall show that there are only finitely many stages $t > s$ such that $\gamma(e,t)\downarrow$ and some number z strictly between $\tilde{\gamma}(e-1) < z < \gamma(e,t)$ is enumerated (into either A or B) at t.*

Proof of claim: Since $\alpha >_L TP$, there is a stage $s_1 > s$ such that α (and hence all nodes in Ξ) gets initialized at stage s_1. We claim that after stage s_1, no number z such that $\tilde{\gamma}(e-1) < z < \gamma(e,t)$ is enumerated at any stage $t > s_1$. By Lemma 13.2.4, any such enumeration must be due to the action of some $\langle \beta, m\rangle$-module for some $\beta \in \Xi$. Let β be the first (wrt time) node in Ξ that enumerates such an element at stage $t > s_1$, and let $s_2 \geq s_1$ be the largest stage $< t$ where β is initialized. Let also u be the smallest stage such that $s_2 \leq u < t$, and β is visited at u which $\gamma(e, u) \downarrow$. Such a stage u must exist, because the counterexample z enumerated by β at t is of the form $z = \gamma(\eta, \xi)$ for some η, ξ which are picked after stage s_2, and by the construction when these parameters are picked, $\gamma(e)$ must be defined.

Now $\eta > s_2 > e$; therefore in order for us to get a contradiction, we will need to show that between the two events at u and t, $\gamma(e, u)$ is never moved (i.e. $\gamma(e, u) = \gamma(e, t)$). The sequence of stages is as follows:

$$\begin{array}{ccccc} s_2 & \leq & u & < & t \\ \beta \text{ is initialized} & & \beta \text{ is visited,} & & \beta \text{ enumerates} \\ & & \text{and } \gamma(e,u) \downarrow & & \text{counter-example} \end{array}$$

Note that u and t are chosen to be minimal. Suppose that $\gamma(e)$ is moved between the two events at stages u and t. By minimality of t, it must be the case that the number $\gamma(e, u)$ itself is enumerated (as a current marker value) by the action of some node δ. Note that either $\delta \subseteq \beta$, or else δ is of lower β-priority. There are two cases :

(i) δ is a noncontiguity node: if δ is of lower β-priority then δ is also initialized at stage s_2, and if $\delta \subseteq \beta$, we have $\delta \in \Xi$ and is initialized at stage s_1. In either case, δ cannot start a module which enumerates $\gamma(e, u)$ since $s_1 > e$.

(ii) δ is an $\mathcal{N}_i$-node for some i: by the choice of s, δ has to enumerate $\gamma(e, u)$ under the action of "Cancellation". Suppose that $\delta \subset \beta$. Then $\delta^\frown\infty$ is visited at stage u. If δ is to enumerate $\gamma(e, u)$ at some stage $u' > u$ then some number $z' < \gamma(e, u)$ must enter between the two visits to δ at stages u and u', which is a contradiction to the minimality of t.

Suppose now that $\delta \supset \beta$, and that δ enumerates $\gamma(e, u)$ at stage u' where $u \leq u' < t$. Let $u'' < u'$ be the previous δ-expansionary stage. Since δ is initialized at stage s_2, we must have $u'' \geq s_2$. By the minimality of t (as above), we must have $u'' < u$. We must have some number $q < u''$, and $q < \gamma(e, u)$ which is enumerated between the two δ-expansionary stages at u'' and u'. By the minimality of u, and the fact that $\delta \supset \beta$, it must be that at the δ-expansionary stage u'', we have $\gamma(e)[u''] \uparrow$. When q is enumerated, it cannot be a current marker, because the next definition that $\gamma(e)$ receives after the δ-expansionary stage at u'' has to be larger than u''. Therefore q has to be enumerated by a noncontiguity node (and furthermore cannot be a current marker at the point of enumeration). A noncontiguity node making the enumeration of q has to do so under step (3.2) of the construction (and when the module is at state 3), and it has to also ensure that $\gamma(e)$ is defined at the point of enumerating q. Since $\gamma(e) > u'' > q$ at the point of enumerating q, this contradicts the minimality of t.

Lastly, suppose that $\delta >_L \beta$. Suppose once again that δ enumerates $\gamma(e, u)$ at some stage $u < u' < t$. Let $u'' < u'$ be the previous δ-expansionary stage. Clearly we must also have $u'' < u$ in this case (by minimality of t), but this would mean that we would visit left of δ (and hence there will be a δ-initialization) between the two δ-expansionary stages.

Hence we conclude that δ does not exist, and $\gamma(e)$ is not moved between u and t, giving us our contradiction. This ends the proof of Claim 13.2.5.1. We may now suppose that s is large enough so that no element in the interval strictly between $\tilde{\gamma}(e-1)$ and $\gamma(e, t)$ is ever enumerated at any stage $t > s$ (whenever $\gamma(e, t)$ is defined). Hence any node which resets $\gamma(e, t)$ at any stage $t > s$ has to be a contiguity node β such that $|\beta| \leq e$. We will now prove the next claim:

Claim 13.2.5.2: *Let α be a contiguity node, and $t > s$ be α-expansionary such that no node $\beta \supseteq \alpha^\frown\infty$ ever resets $\gamma(e, u)$ at any stage $u \geq t$. Then, α itself will not reset $\gamma(e, u)$ at any stage $u > t$.*

Proof of claim: Let $u > t$ be the next α-expansionary stage. It suffices to show that α does not reset $\gamma(e, u)$ at stage u (the claim then follows because t is arbitrary). Suppose the contrary, hence there is some number $q < t$ and $q < \gamma(e, u)$ which is enumerated between the two α-expansionary stages t and u. We consider two cases. Firstly, suppose $\gamma(e)$ is undefined when α finishes its actions at t. As in the previous claim, when q is enumerated, $\gamma(e)$ must be undefined because the next definition $\gamma(e)$ receives is larger than t. The alternative is also clearly bad. Therefore we can dispense with the first case.

Suppose the second case holds, i.e. after α finishes its actions at t, we have $\gamma(e,t)\downarrow$. Therefore it must be that some contiguity node δ enumerates $\gamma(e,t)$ under "Cancellation", some time between t and u. If $\delta^\frown\infty \subseteq \alpha$ then t would also be δ-expansionary and so this is impossible. The other possibilities for δ are easy. This completes the argument for Claim 13.2.5.2.

Finally, it follows by a simple inductive argument and Claim 13.2.5.2, that $\gamma(e,s)$ eventually settles. □

Next, we verify that the strategies succeed along the true path.

Lemma 13.2.6. *Along the true path, the contiguity requirements succeed.*

Proof. Let α be an $\mathcal{N}_e^A$-node on the true path, and suppose that $\Phi_e(A)$ is total and $\Psi_e(\Phi_e(A)) = A$. Let s be a stage large enough so that

(i) $(\forall t > s)\delta_t \not<_L \alpha$,

(ii) for all β of higher α-priority, r_β has settled (by Lemma 13.2.3),

(iii) noncontiguity nodes $\beta \subset \alpha$ make no more enumerations,

(iv) for every $i < |\alpha|$, $\gamma(i)$ has settled,

(v) $A_s\restriction r = A\restriction r$, where $r = 1 + \max\{\tilde{r}_\beta \mid \beta \text{ is of higher } \alpha\text{-priority}\}$.

Claim 13.2.6.1: *There are infinitely many α-expansionary stages.*

Proof of claim: To see this, we first let i be the least number such that $\tilde{\gamma}(i) > r$, and $i \geq |\alpha|$. The nested length of agreement $l_A(e,s)$ has infinite lim inf. Hence it is not hard to see that $\tilde{\gamma}(j)$ for every $j \geq i$ must eventually be α-confirmed. Each of these confirmations corresponds to an α-expansionary stage.

Claim 13.2.6.2: *At any stage $t > s$, if a number x receives α-confirmation, then no number y where $x < y \leq t$ can be enumerated into A after that.*

Proof of claim: Since the confirmation action cancels all current marker values $\{\gamma(i,t) \mid \gamma(i,t) > x\}$, any such y has to be enumerated into A by an A-noncontiguity node β. The only nontrivial case is when $\beta \supseteq \alpha^\frown\infty$. By Lemma 13.2.2, y cannot be a follower of any β-module at the instance when x receives α-confirmation. Hence y

is chosen by β after that, so that $y \leq x$ or else $y > t$.

Claim 13.2.6.3: *At any stage $t > s$, if conditions (i)-(iii) for cancellation are met by α, then no number p such that $y < p < t$ can be enumerated into A after that (where y is the number in condition (ii)).*

Proof of claim: Suppose such a p exists. If p is a current marker at t, i.e. $p = \gamma(j, t)$ for some j, then we must have either $j < |\alpha|$ or $\gamma(j, t) \leq r$ or else $B \upharpoonright y + 1$ changes during cancellation at t. The first two cases are not possible by choice of s. In any case p is never a current marker value after α-cancellation at t. Thus p has to be enumerated by an A-noncontiguity node β. When did β pick p as a follower? By the preceding analysis it follows that p has to be picked before α-cancellation at t. Therefore we can only have $\beta \supseteq \alpha^\frown\infty$. On the other hand, p has to be picked after the visit to α at t^-, because y has to enter A between t^- and t (in fact, p can only be picked after y enters A). Since $\beta \supseteq \alpha^\frown\infty$, it follows that β has to in fact pick p at stage t^-, after the construction visits α. If y only enters strictly after stage t^-, then p will be cancelled, so y has to enter A at stage t^-. However the entry of y at stage t^- will cancel all current markers larger than it until the end of the stage. This gives us a contradiction to the fact that p exists.

Finally, we will show that $A \equiv_{wtt} \Phi_e(A)$. To wtt-compute A from $\Phi_e(A)$, we fix an arbitrary $x > r$, $x > \tilde{\gamma}(|\alpha|)$ and x is not α-confirmed at s. If x is never picked as a marker value, then $x \notin A$, so we wait until x is picked at stage $t < x$. Let $x = \gamma(i, t)$ for some $i > |\alpha|$. Since there are infinitely many α-expansionary stages, if $\gamma(i, t)$ is never reset after stage t, then it must eventually be α-confirmed. There are two possibilities :

(i) $\gamma(i, t)$ is reset before it is α-confirmed. In this case, only a noncontiguity node β can enumerate $\gamma(i, t)$ after it is reset. Since it is never α-confirmed, $\beta \not\supseteq \alpha^\frown\infty$. If $\beta >_L \alpha^\frown\infty$ then at the next α-expansionary stage, β will be initialized (so wait till then and see what happens).

(ii) $\gamma(i, t)$ is α-confirmed before it is reset (if ever). Note that confirmation happens after stage s. Let $v > t$ be the stage where $\gamma(i, t)$ is α-confirmed, and $u =$

$\psi_e(\Phi_e(A);x)[v]$. If $\Phi_e(A)\restriction u = \Phi_e(A)\restriction u[v]$, then $A_v(x) = A(x)$. If $\Phi_e(A)\restriction u \neq \Phi_e(A)\restriction u[v]$, then some number $y \leq x$ has to enter A after stage v (applying Claim 13.2.6.2). Wait for the first y to enter A at a stage $v' \geq v$. If $y = x$ then we are done, and if $y < x$ then the only reason for x to enter A after stage v' would have to be due to an A-noncontiguity node β. This is not possible because the relevant β-module will have to be initialized before it has a chance to enumerate x. Note that it is possible that x enters B later, but this is of no relevance to α. Finally, observe that u can be computed effectively from x.

Next, to wtt-compute $\Phi_e(A)$ from A, we pick an arbitrary z, and try to decide $\Phi_e^A(z)$. Let $t > s$ be the least α-expansionary stage where $l_A(e,t) > z$, and let $u \geq t$ be the least stage where u is α-expansionary and $A\restriction t = A_u\restriction t$. We claim that $\Phi_e^A(z)[u] = \Phi_e^A(z)$, in particular that $A\restriction u[u] = A\restriction u$. If $t = u$ then the result holds, so we let t' be the largest α-expansionary stage such that $u > t' \geq t$. Since $A\restriction t \neq A\restriction t[t']$, the conditions (i)-(iii) under cancellation are met when α is again visited at stage u. By Claim 13.2.6.3, we have $A\restriction u$ is stable after u. □

Lemma 13.2.7. *Along the true path, the noncontiguity requirements succeed.*

Proof. Fix e, i such that $W_e \not\leq_T A$, and let α be the $\mathcal{Q}_{e,i}^A$-node on the true path. It is obvious that $Q_{\tau(\alpha)} \equiv_T A \oplus W_e$. We shall show that $\hat{\Delta}_i(Q_{\tau(\alpha)}) \neq A$. Suppose that the contrary holds, then $\lim_t l(\alpha,t) = \infty$. Let s be large enough, such that

(i) $(\forall t > s)\delta_t \not<_L \alpha$,

(ii) for all β not of lower α-priority, r_β has settled,

(iii) noncontiguity nodes $\beta \subset \alpha$ make no more enumerations,

(iv) for every $i < |\alpha|$, $\gamma(i)$ has settled,

(v) α is visited at stage s.

Suppose that w is the true outcome of α; we shall compute W_e from A for a contradiction. Fix an arbitrary x.

Claim 13.2.7.1: For all but finitely many stages $t > s$, the $\langle\alpha, x\rangle$-module is in state 2.

Proof of claim: To see this, note that for every $t > s$, we must have $F_{\alpha,x}[t] \in \{0,1,2\}$ (since w is the true outcome). Since α never gets initialized after s, it follows that $F_{\alpha,x}[t] = 0$ at only finitely many $t > s$. Hence $\lim_t L_{\alpha,x}[t]$ exists, and so let u be a stage such that $L_{\alpha,x}[u]$ has settled, and also $\gamma(p)$ has settled for every $p \in L_{\alpha,x}[u]$. Since $l(\alpha,t)$ tends to ∞ it follows that $F_{\alpha,x}$ eventually reaches state 2 after u. After it reaches state 2 (after u), it stays in state 2 forever (because $\gamma(p)$ has settled for every $p \in L_{\alpha,x}[u]$). The crucial point is that if the module is initialized by any entry of some small number, then we will go back to state 1 and we do not re-pick the numbers in $L_{\alpha,x}$.

Now, we need to compute $W_e(x)$ from A. We wait for the first stage $t > s$ such that $F_{\alpha,x}[t] = 2$, A up till $1 + \gamma(\eta_{\alpha,x}[t], \xi_{\alpha,x}[t])$ is correct, and α is visited. We let t' be the first such stage, $s < t' < t$. Stages t and t' exist, by Claim 13.2.7.1.

Claim 13.2.7.2: $W_{e,t}(x) = W_e(x)$.

Proof of claim: It is enough to show that the $\langle\alpha, x\rangle$-module is never initialized after t. Suppose $u > t$ is the first stage where the $\langle\alpha, x\rangle$-module is initialized. Note that $\eta_{\alpha,x}[u] \leq \eta_{\alpha,x}[t]$ and $\xi_{\alpha,x}[u] = \xi_{\alpha,x}[t]$. Consequently there is some number $y \leq \gamma(\eta_{\alpha,x}[t], \xi_{\alpha,x}[t])$, such that $y \in B_u \setminus B_t$ is enumerated under the action of some β of higher α-priority. Now, β has to be an A-contiguity node and stages t and u are both β-expansionary. By Lemma 13.2.2, β has to enumerate y under "Cancellation" and therefore $A\restriction y[t] \neq A\restriction y$, which is not possible.

Therefore, it must be the case that d is the true outcome of α. Since r_α has also settled by s, we may assume that α always plays outcome d whenever it is visited on or after s. Let n be the least such that $F_{\alpha,n}[s] \in \{3,4\}$, and $t \leq s$ be the largest stage such that $F_{\alpha,n}$ changes state from 2 to 3. We can see that the $\langle\alpha, n\rangle$-module cannot be initialized after stage t, and after running all the α-modules at stage t, n must be least such that $F_{\alpha,n}[t] \in \{3,4\}$. Hence after stage t, $\alpha^\frown w$ cannot be visited again, and only the $\langle\alpha, n\rangle$-module will be run at every visit to α.

When α is visited at stage t, $\tau(\alpha)$ must have already taken the action to lift the values $\lambda_{\tau(\alpha)}(m,t)$ for all $m > 2n+1$ above the value $\hat{\delta}_i(\gamma(\eta_{\alpha,n}[t], \xi_{\alpha,n}[t]))$. Note that $\eta_{\alpha,n}[t] = \max L_{\alpha,n}$. Since the use of any convergent computation $\hat{\Delta}_i(X;p)$ for any oracle X, and $p \leq \gamma(\eta_{\alpha,n}[t], \xi_{\alpha,n}[t])$ is bounded by $\hat{\delta}_i(\gamma(\eta_{\alpha,n}[t], \xi_{\alpha,n}[t]))$, it follows that at any stage $u > t$, if a convergent computation $\hat{\Delta}_i(Q_{\tau(\alpha)}; \gamma(q, \xi_{\alpha,n}))[u]$ for some $q \in L_{\alpha,n}[u]$ observed at stage u is not preserved, it is necessary that some number $p < \hat{\delta}_i(\gamma(\eta_{\alpha,n}[t], \xi_{\alpha,n}[t]))$ must enter $Q_{\tau(\alpha)}$ after stage u. Since the only numbers to enter $Q_{\tau(\alpha)}$ are the $\lambda_{\tau(\alpha)}$-marker values, this means that m has to enter W_e after stage t for some $m \leq n$. Note that no $m \leq n$ can enter A after stage t, because if that was so, then $m \leq n < k < \gamma(k, \xi_{\alpha,n}[t])$ for every $k \in L_{\alpha,n}[t]$, which would cause the module to be initialized.

Since $\lim_u l(\alpha, u) = \infty$, it follows that there are infinitely many stages $u > t$ such that $F_{\alpha,n}$ changes state at u (between 3 and 4). Every time the state cycles from 3 to 4 and then back to 3, we must have a corresponding change in $W_e \upharpoonright n+1$. Let $u > t$ be the least stage such that α is visited, and $W_{e,u} \upharpoonright n+1 = W_e \upharpoonright n+1$, and $F_{\alpha,n}[u] = 3$. Since $|L_{\alpha,n}[u]| = n+1$, we will never run out of choices for $\eta_{\alpha,n}$. Once $W_e \upharpoonright n+1$ is stable, we cannot have recovery of $l(\alpha)$ anymore, giving a contradiction. Hence $\hat{\Delta}_i(Q_{\tau(\alpha)}) \neq A$. $\square$

Finally, it follows from Lemma 13.2.5 that the usual marker rules for γ are met. Hence $\Gamma(A \oplus B)$ is total and equals $\emptyset'$. This ends the proof of Theorem 13.2.1.

Chapter 14

Limits on jump inversion for strong reducibilities

The work in this chapter is joint with Barbara Csima and Rod Downey. It has been submitted for publication.

14.1 Introduction

The concern of this chapter is the interaction of two basic notions from computability theory. These are the jump operator and reducibilities stronger than Turing reducibility which are of the tabular type. We answer a question of Anderson [And08] by showing that there are no analogues of the Sacks Jump Inversion Theorem [Sac63c] and Shoenfield's Jump Inversion Theorem [Sho59] for these strong reducibilities.

The study of strong reducibilities has been part of computability since the dawn of the subject, as witnessed by Post's paper [Pos44]. A is Turing reducible to B, $A \leq_T B$, means that A can be computed by B via any oracle access mechanism. It is clearly natural to ask what happens when we restrict the access mechanism in the reduction from A to B. Tabular reducibilities such as weak truth table (wtt-) and truth table (tt-) reducibilities do not allow the reducibility to be adaptive. Thus, as is well known, $A \leq_{tt} B$, is defined as $x \in A$ iff $B \models \sigma_{f(x)}$ where f is a computable function and $\sigma_{f(x)}$ is the $f(x)^{th}$ truth table. As is also well known a truth table reduction is simply a Turing reduction Φ which is total for all oracles. Weak truth table reducibility simply has the truth table being partial, or $\Phi^B = A$ where the use of the computation $\varphi(x)$ is a computable function. Thus in either case we are not allowed to *adapt* the size of the reduction as the oracle B varies. $A \leq_{tt} B$ implies $A \leq_{wtt} B$ but it is easy to construct examples where the converse fails.

These reducibilities also arise very naturally when we consider reducibilities coming from reductions in mathematical structures. For example, the reduction of the word problem to the conjugacy problem in combinatorial group theory is a tt-reduction and the degrees of bases of c.e. vector spaces are naturally represented by weak truth table degrees (Downey and Remmel [DR89]).

In recent times, truth table reducibility has become a central area of interest as it has been shown to be a natural reducibility to study in *algorithmic randomness*, a fact first realized by Demuth [Dem88]. The point here is that if $A \leq_{tt} B$, via $\Phi^B = A$, with Φ total on all oracles, then we can use Φ to translate between measures effectively. For instance if B is random with respect to uniform measure, and A is noncomputable, A will be random with respect to the measure generated by the inverse of Φ. Thus, for instance, truth table degrees are absolutely central to the deep investigations of Reimann and Slaman [RS08a, RS08b] on sets never continuously random. They are also deeply connected with things like the Cantor-Bendixson rank of sets for a similar reason.

All of this recent work has highlighted our lack of understanding as to how the finer structure of the (w)tt-degrees relates to the jump operator. The halting problem is a fundamental object of computability theory, and the jump $A' = \{e : \Phi_e^A(e) \downarrow\}$ is the relativized form of the halting problem.

For Turing reducibility, we know a lot about how the jump operator behaves. The most basic theorem is Friedberg's Jump Inversion Theorem [Fri58a], that if $X \geq_T \emptyset'$ then there is a set A with $A' \equiv_T X \equiv_T A \oplus \emptyset'$. Early on, Mohrherr [Moh84] proved that if $X \geq_{tt} \emptyset'$, then there is a set A with $A' \equiv_{tt} X$. Mohrherr's proof came from an analysis of Friedberg's Theorem, and resulted in a 1-generic set A. It was only much later that Anderson [And08] proved that indeed the full analogue of Friedberg's Theorem held; if $X \geq_{tt} \emptyset'$ then there is a set A with $A' \equiv_{tt} X \equiv_{tt} A \oplus \emptyset'$. Anderson's theorem was more difficult than Mohrherr's, and the method employed by Friedberg (which will give generic sets) provably *fails*, so that arguments akin to those from information theory were necessary.

The most important sets in computability theory are the c.e. sets as well as those computable from the halting problem, the Δ^0_2 sets. Shoenfield [Sho59] had proven a jump theorem for such sets. Namely for any Σ^0_2 set $X \geq_T \emptyset'$ there is a Δ^0_2 set A with $A' \equiv_T X$. A well-known fact is that Sacks used the infinite injury method to show that the same result held with A a computably enumerable set. After that many other intricate jump theorems were found culminating in Robinson's Jump Interpolation Theorem [Rob71b]. (See Soare [Soa87] for more details.)

Anderson asked: Do the analogues of any of these basic theorems hold for tt- or perhaps wtt-reducibilities? We prove that the analogues fail to hold and in fact that they fail in more or less the strongest way that they can. Our first result shows that the Sacks Jump Inversion Theorem fails for both the tt- and wtt-reducibilities, by constructing a Δ^0_2 counter-example. We will in fact prove something stronger:

Theorem 14.3.3. *For any computable sequence of Δ^0_2 sets $\{V_e\}_{e\in\mathbb{N}}$ (given by their Δ^0_2 indices), there exists a Δ^0_2 set $S \geq_{tt} \emptyset'$ such that for every e, $V_e' \not\equiv_{wtt} S$.*

From Theorem 14.3.3 we deduce the failure of the Sacks Jump Inversion for both tt- and wtt-reducibilities:

Theorem 14.3.4. *There exists an $(\omega+1)$-c.e. set $S >_{tt} \emptyset'$ such that there is no c.e. set A with $A' \equiv_{wtt} S$.*

Hence S is in the first place of the Ershov Hierarchy where a counter-example can be. The result also gives an interesting fact about the Δ^0_2 wtt-degrees which are realized as the jumps of low c.e. sets. Clearly there are such wtt-degrees $\mathbf{a} > \mathbf{0'}_{wtt}$, namely the wtt-degrees of the jumps of low but not superlow c.e. sets. Our result shows that *not every* wtt-degree $\mathbf{a} > \mathbf{0'}_{wtt}$ can be realized as the jump of a (low) c.e. set.

Our second result shows that the analogue of the Shoenfield Jump Inversion Theorem fails for both the tt- and wtt-reducibilities. By Mohrherr's result, the counter-example S has to be strictly Σ^0_2:

Theorem 14.3.5. *There exists a Σ^0_2 set $S >_{tt} \emptyset'$ such that there is no Δ^0_2 set A with $A' \equiv_{wtt} S$.*

Our notation is fairly standard, as found in Soare [Soa87].

14.2 The basic module

14.2.1 The plan for the c.e. case

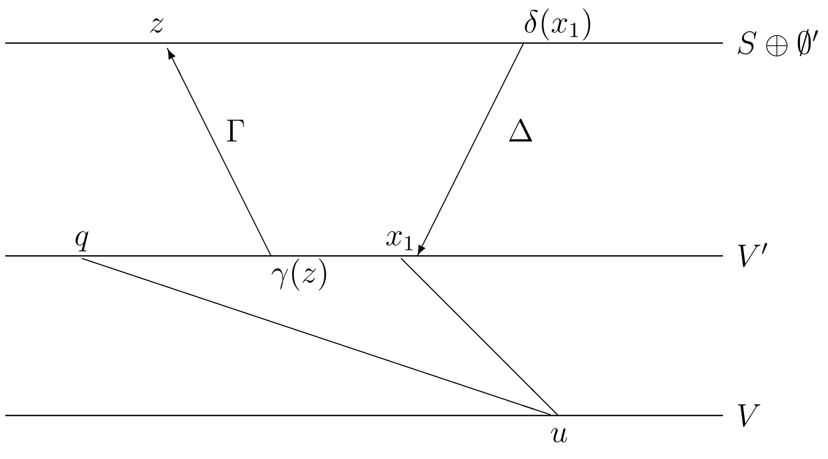
Let $(\Gamma_e, \Delta_e, \gamma_e, \delta_e)_{e\in\omega}$ run through all possible 4-tuples where Γ_e and Δ_e are Turing functionals, and γ_e and δ_e are partial computable functions. Let us suppose we wanted to prove Theorem 14.3.4 directly by constructing S. We must then meet for all $e \in \omega$ the requirements:

$$R_e : \Gamma_e^{V_e'} \neq S \text{ or } \Delta_e^{S\oplus\emptyset'} \neq V_e',$$

where V_e is the e^{th} c.e. set, and γ_e and δ_e bound the uses of the computations of Γ_e and Δ_e, respectively. Then $S \oplus \emptyset'$ will be the desired set. Note that the requirements automatically ensure that $S \oplus \emptyset' \not\equiv_{wtt} \emptyset'$.

Suppose we wanted to satisfy R_e. We can first try making $\Delta^{S\oplus\emptyset'} \neq V'$ (for the purpose of the discussion we drop the subscript e). In particular we assume that the recursion theorem gives us infinitely many indices $x_1, x_2, \cdots$ for which we can control $V'(x_i)$. The obvious plan is to keep $V'(x_1) = 0$ until $\Delta^{S\oplus\emptyset'}(x_1) \downarrow= 0$. We then make $V'(x_1) = 1$ by enumerating an axiom with some use $V\restriction u$. The only way in which $\Delta^{S\oplus\emptyset'}(x_1)$ can later change to be 1, is for some number $< \delta(x_1)$ to enter $\emptyset'$. Our next step would be then to extract x_1 out from V'; if we could always do this then we would know what to do. We would alternate the value of $V'(x_1)$, and we will eventually succeed because $\emptyset'$ is c.e. and the use $\delta(x_1)$ is fixed. Unfortunately we only have partial control over V' and extraction can only be achieved by forcing a change in $V\restriction u$.

We can start another line of attack by trying to make $\Gamma^{V'} \neq S$ true. We pick an attacker z for S, and for simplicity let us first consider the case where Γ is an m-reduction; i.e. $z \in S$ iff $q \in V'$ for some q. We begin by making $S(z) = 1$, and wait for $V'(q) = 1$, i.e. $\Phi_q^V(q)\downarrow$ with some use u. Note that while the uses on Δ and Γ are bounded, this use u may be unbounded. We then begin the attack above by enumerating an axiom $\Phi_{x_1}^V(x_1)$ with the same use u, and wait for a $\emptyset'$-change.



If no $\emptyset'$-change occurs then it is clear that we will succeed at $\Delta^{S\oplus\emptyset'}(x_1) \neq V'(x_1)$. If on the other hand a $V\restriction u$ change occurs before a $\emptyset'$-change, then we will wait for $\Phi_q^V(q)\downarrow$ again with a new use u', and then make $\Phi_{x_1}^V(x_1)$ converge with the same use u'. The point is that if V changes infinitely often this way with no $\emptyset'$-change, then $V'(q) = 0$ and we will succeed via $\Gamma^{V'}(z_e) \neq S(z)$. Lastly if $\emptyset'$ changes then we will remove z from S, and wait for $\Phi_q^V(q)$ to become undefined again. This has to happen (unless already $\Gamma^{V'} \neq S$), and so at some point we will also get a clear on $\Phi_{x_1}^V(x_1)\uparrow$. We can then repeat by making $S(z) = 1$ again. Note that we only toggle z in S whenever $\emptyset'$ changes below $\delta(x_1)$, so requirement R_e can be satisfied with only finite action on S (although we might possibly enumerate infinitely many axioms for x_1).

The strategy for a general wtt-reduction Γ is as above, but we will run a separate copy of the strategy above for each possible configuration of the use $V_e' \restriction \gamma_e(z_e)$. We have $2^{\gamma_e(z_e)}$ many different x's corresponding to each different configuration of $V_e'\restriction\gamma_e(z_e)$. At each stage we look at the current approximation for $V_e'\restriction\gamma_e(z_e)$ (see section 14.2.2) and apply the above plan. As in the basic case we will only toggle z_e in S if $\emptyset'$ changes below some δ_e-use. Each time we toggle z_e we will force the configuration $V_e'\restriction\gamma_e(z_e)$ to change. This can only move lexicographically right finitely many times (consecutively), hence after finitely much toggling of z_e, the configuration for $V_e'\restriction\gamma_e(z_e)$ will return to an earlier one. This makes all the x_τ (for all the τ to the right of the current V_e'-configuration) undefined, so that if $\tau \subset V_e'[s]$ holds again later we can use x_τ to cause further $\emptyset'$-changes.

The above works when diagonalizing against all c.e. sets. However if V is Δ_2^0

then whenever the configuration of $V' \upharpoonright \gamma_e(z_e)$ returns to an earlier one, there is no guarantee that all x_τ (for τ to the right of the current V'-configuration) become undefined. However we can show that some amount of progress has been made because in this case, V has to *return to a previous x_τ axiom*, and thus we will threaten V to be not Δ^0_2.

From the above discussion, the reader will notice that the different requirements act almost independently of one another. In fact all that a single requirement needs to know is the correct initial segment of S. When diagonalizing against all Δ^0_2 sets, it may be possible for a requirement to flip S infinitely often. However we do not need to re-pick the followers of lower priority requirements due to this reason. The only reason why R_2 needs to pick a new z_2 is because R_1 has seen δ_1, γ_1 converge, and wants to protect now a certain segment of S. This initialization happens only finitely often (despite R_1 flipping $S(z_1)$ infinitely often). Therefore it will be straightforward to combine the requirements, and will not require a tree argument as one usually expects in full approximation arguments.

14.2.2 The modular approach

We proceed in a general setting, and then obtain the main theorems as corollaries. We start by fixing a computable sequence $\{V_e\}_{e\in\mathbb{N}}$ of possible Δ^0_2-approximations. That is, $V_{e,s}(x)$ is a computable function of e, s, x. We say that V_e is Δ^0_2 if $\lim_s V_{e,s}(x)$ exists for all x and $V_e(x)$ is this limit.

Let the natural approximation of the jump of V_e be $V'_{e,s}(n) = 1$ iff $\Phi^{V_{e,s}}_{n,s}(n)\downarrow$ (as is customary we assume the hat trick, that there must be a divergence between consecutive convergence with different uses, see Soare [Soa87]). If V_e is Δ^0_2 then this serves as a natural Σ^0_2-approximation to the characteristic function of V'_e in the sense that $V'_e(n) = \liminf_s V'_{e,s}(n)$ for every n. However when approximating $V'_e \upharpoonright x$ as a finite string, $V'_{e,s} \upharpoonright x$ is obviously not ideal because $V'_e \upharpoonright x$ might not be the lexicographically leftmost string specified by $V'_{e,s} \upharpoonright x$ at infinitely many s. It is easy to fix this by delaying any entry of n into the (approximation for the) jump by the following.

We define another approximation $Q_e \upharpoonright x[s]$ for $V'_e \upharpoonright x$ this time by induction as follows: $0 \in Q_e[s]$ iff $\Phi^{V_{e,s}}_{0,s}(0)\downarrow$. For $n > 0$, let $t < s$ be maximal such that $Q_e \upharpoonright n[t] =$

$Q_e\!\upharpoonright\! n[s]$. If $\Phi_{n,r}^{V_{e,r}}(n)\downarrow$ for all $t < r \leq s$, and V_e has been stable up to the use during this period, declare $n \in Q_e[s]$, and declare $n \notin Q_e[s]$ otherwise. Hence if V_e is Δ_2^0 then the lexicographically leftmost segment $Q_e\!\upharpoonright\! x[s]$ specified infinitely often is the segment of the true jump $V_e'\!\upharpoonright\! x$. The "delayed" approximation $\{Q_e[s]\}_s$ will be used when deciding whether or not to act for a module, since it is correct infinitely often. Furthermore the delayed approximation Q_e for V_e' is obtained effectively in e.

We have infinitely many modules $M_{\theta,e}$ indexed by a finite binary string θ and $e \in \mathbb{N}$. Here $M_{\theta,e}$ works in a similar way as requirement R_e above, and guesses that $\theta \subset S$. It outputs (effectively) an infinite binary sequence $m_{\theta,e}$ listing the stage by stage guesses as to whether our toggle point $z_{\theta,e}$ is in S, as well as a number $d_{\theta,e}$ such that if V_e is Δ_2^0, then

(P1) $m = \lim_s m_{\theta,e}(s)$ exists,

(P2) if γ_e and δ_e are total, then $d_{\theta,e}\downarrow$, and we have that either $\Gamma_e^{V_e'}(lz_{\theta,e}) \neq m$ or $V_e' \neq \Delta_e^{(\theta^\frown m^\frown 0^\omega \oplus \emptyset')\upharpoonright d_{\theta,e}}$.

Note that undefined counts as being not equal.

14.2.3 The construction for $M_{\theta,e}$

Now we give the actions of the module $M_{\theta,e}$.

Step 1: Let $z_{\theta,e} = |\theta|$.

Step 2: Wait for $\gamma_e(z_{\theta,e})\downarrow$. Using a strong form of the relativized recursion theorem, for each $\sigma \in 2^{\gamma_e(z_{\theta,e})}$, let $x_\sigma > \gamma_e(z_{\theta,e})$ be a number that we control for V_e'. That is, we may enumerate axioms for $\Phi_{x_\sigma}^X(x_\sigma)$ and also specify the use of the axioms. In short, we call these *axioms for* x_σ. Note that x_σ for different modules are different.

Wait for $\delta_e(\max_\sigma x_\sigma)\downarrow$. Let $d_{\theta,e} = \delta_e(\max_\sigma x_\sigma)$, and proceed to Step 3.

Step 3: We say that *s is a recovery stage* if $\Gamma_e^Q(z_{\theta,e})[s] \downarrow= m_{\theta,e}[s]$ and $\Delta_e^{(\theta^\frown m_{\theta,e}{}^\frown 0^\omega \oplus \emptyset')\upharpoonright d_{\theta,e}}[s] \downarrow= Q[s]\!\upharpoonright\!(1 + \max_\tau x_\tau)$.

For the clarity of presentation, we assume that the enumeration of Q_e is fixed and independent of our actions. In particular we do not follow the customary practice of using a slowdown lemma in the enumeration of the jump. That is, when we define some $\Phi_{x_\sigma}^{V_e}(x_\sigma)$ to converge, we do not assume that $Q_e(x_\sigma)$ responds instantly. If this computation we defined is indeed correct then this will be reflected eventually in $Q_e(x_\sigma)$ and we can just wait for it; on the other hand if V_e changes before $Q_e(x_\sigma)$ responds, then we would have made some progress since the use on the axiom for x_σ was based on some other "real" computation reflected earlier by Q_e. Consequently we say that $Q(x)$ is *good at stage* s, if x is an index which we control by the recursion theorem, and $Q_e(x)[s] = 1$ iff there is a current axiom at s which applies for x.

For any set $X \subset 2^\omega$ and any number $n \in \omega$, we let $u(n, X)$ denote the (current) use of the computation $\Phi_n^X(n)$. At each future stage of the construction, for each $\sigma \in 2^{\gamma_e(z_{\theta,e})}$, x_σ will have a mode associated to it, which will be either IN or OUT, reflecting our desire to have x_σ in or out of V_e'. Initially begin with all x_σ in mode OUT. Unless a change in mode is explicitly stated in the ensuing construction, the mode will not change from one stage to the next.

At all successive stages, $m_{\theta,e}(s)$ outputs the previous value unless $z_{\theta,e}$ is toggled in which case we flip $m_{\theta,e}(s)$.

Stage s: Let $\sigma = Q_e \upharpoonright \gamma_e(z_{\theta,e})[s]$.

If x_σ has mode IN, and there is no axiom that currently applies for x_σ, we enumerate an axiom for x_σ with use $V_{e,s} \upharpoonright \max\{u(q, V_{e,s}) \mid \sigma(q) = 1\}$.

If s is a recovery stage and $Q_e(x_\sigma)$ is good, we call s a *good recovery stage* and proceed as follows.

Case 1: No axiom for x_σ applies. Declare x_σ to have mode IN for stage $s + 1$.

Case 2: An axiom for x_σ applies. Declare x_σ to have mode OUT for stage $s + 1$, and toggle $z_{\theta,e}$.

This completes the construction.

14.2.4 Verification

If $\sigma \subset Q_e[s]$ then we will refer to s as a σ-stage. We first make the following observation.

Lemma 14.2.1. *At all stages s after Step 2 is completed, if an axiom applies for x_τ, then it has use* $\max\{u(q, V_{e,s}) \mid \tau(q) = 1\}$ *with all the uses defined. Moreover, if x_σ has mode IN at stage s and $\sigma = Q_e \upharpoonright \gamma_e(z_{\theta,e})[s]$ then $u(q, V_{e,s})\downarrow$ for every q such* *that $\sigma(q) = 1$.*

Proof. The first statement follows directly from the second, while the second statement follows from the fact that if $q \in Q_e[s]$ then $\Phi_q^{V_e}(q)[s] \downarrow$. $\square$

Lemma 14.2.2. *If $\sigma = Q_e \restriction \gamma_e(z_{\theta,e})[s]$ is to the left of τ and an axiom for x_τ currently applies (with use u), then $V_e \restriction u$ cannot have been stable since the last τ-stage.*

Proof. Since τ is to the right of σ there is a least $q < \gamma_e(z_{\theta,e})$ such that $\tau(q) = 1$ and $\sigma(q) = 0$. Since $\sigma(q) = 0$, we have $q \notin Q[s]$. We know $V_{e,s}$ extends η for some x_τ-axiom η enumerated earlier (say at t), hence $\Phi_q^\eta(q)[t] \downarrow$. This means that $\Phi_q^{V_e}(q)[s] \downarrow$ which means that V_e cannot extend η at every stage between the last $\sigma \restriction q$-stage and s (otherwise $q \in Q_e[s]$). $\square$

We recall that we only enter Case 1 or 2 at good recovery stages.

Lemma 14.2.3. *If Step 3 is started and V_e is Δ_2^0, then there are only finitely many good recovery stages. Consequently $z_{\theta,e}$ is toggled only finitely often.*

Proof. Assume for a contradiction that there are infinitely many good recovery stages. Let s be the stage by which $\emptyset'$ has settled on $d_{\theta,e}$. There are at most two possible configurations of $(\theta^\frown m_{\theta,e}{}^\frown 0^\omega \oplus \emptyset')\restriction d_{\theta,e}$ after s, which differ on the value of $m_{\theta,e}$. Every good recovery stage after s is either a σ_0-stage or a σ_1-stage, where σ_i corresponds to the configuration with $m_{\theta,e} = i$.

We first claim that there is a stage $s_1 > s$ such that Case 1 applies. Suppose not. Then at every good recovery stage after s, we must have that Case 2 applies, whence $z_{\theta,e}$ is toggled. Thus as we visit the good recovery stages after stage s, we must be alternating between the two configurations σ_0 and σ_1, in order to recover the toggles. Also, after we have our first good recovery stage with configuration σ_i after stage s, we give x_{σ_i} mode OUT. Since we are assuming we never enter Case 1 after stage s, this means that x_{σ_i} will remain in mode OUT for the duration of the construction. In particular, it follows that only finitely many axioms are enumerated for x_{σ_1}. Let $t > s$ be a stage where $V_e \restriction \max\{\text{uses of the } x_{\sigma_1} \text{ axioms}\}$ is stable. Let $t_1 > t$ be a good recovery stage with configuration σ_1. Since we were in case 2, an axiom for x_{σ_1} applied at stage t_1. Let $t_2 > t_1$ be a good recovery stage with configuration σ_0. Since $t_1 > t$, the axiom for x_{σ_1} still applied at stage t_2. Now since $\sigma_0 = Q_e \restriction \gamma_e(z_{\theta,e})[t_2]$ is to the left of σ_1, Lemma 14.2.2 shows that V_e could not have been stable on the x_{σ_1} axiom since the previous σ_1-stage, giving the desired contradiction.

The above contradiction shows that s_1 exists. Suppose s_1 is a τ-stage. Let $s_2 > s_1$ be the next good recovery stage (we want to get a contradiction). Since $z_{\theta,e}$ is not toggled by the actions at s_1, it follows that the configuration of $(\theta^\frown m_{\theta,e}{}^\frown 0^\omega \oplus \emptyset')\upharpoonright d_{\theta,e}$ at the beginning of s_2 is the same as at the beginning of s_1. Hence s_2 is also a τ-stage. Since x_τ receives mode IN at s_1, it follows that x_τ has mode IN at the beginning of s_2, where an axiom for x_τ will be enumerated (if there is not already one). At s_1, $Q_e(x_\tau)$ must be 0 because of its goodness, which means that at s_2, $Q_e(x_\tau)$ must be again 0 since s_2 is a recovery stage and a τ-stage. But an axiom for x_τ applies at stage s_2, and s_2 is a good stage, so $Q_e(x_\tau) = 1$, a contradiction. $\square$

Lemma 14.2.4. *$M_{\theta,e}$ satisfies (P1) and (P2).*

Proof. If V_e is Δ^0_2, then (P1) holds by Lemma 14.2.3. To show (P2) holds as well we assume that δ_e, γ_e are total (hence Step 3 is started). Let $\sigma = V_e'\upharpoonright \gamma_e(z_{\theta,e})$, the true segment of V_e'. Also let $r = V_e'(x_\sigma)$. Hence there are infinitely many σ-stages s where $Q_e(x_\sigma)[s] = r$. We claim that $Q_e(x_\sigma)$ is good at almost every such stage.

For every p such that $\sigma(p) = 1$, p must be in the real V_e' and so it is easy to see that once V_e is stable on these uses, any x_σ-axiom we enumerate applies forever. Hence we only enumerate finitely many axioms for x_σ. If V_e extends one of these axioms then at almost every stage $V_e[s]$ extends the axiom and also $Q_e(x_\sigma) = 1$. If V_e extends none of these axioms then at almost every σ-stage where $Q_e(x_\sigma)[s] = 0$, we have $V_e[s]$ extending none of these axioms. Hence $Q_e(x_\sigma)$ is good at almost every σ-stage s where $Q_e(x_\sigma)[s] = r$.

Assume for a contradiction that the last condition in (P2) fails. By Lemma 14.2.3 let s_0 be a stage by which $(\theta^\frown m_{\theta,e}{}^\frown 0^\omega \oplus \emptyset')\upharpoonright d_{\theta,e}$ has settled. There are infinitely many stages after s_0 where $Q_e[t]\upharpoonright 1 + \max\{x_\tau\}$ is correct, and each of these is a recovery stage. By the above paragraph we will have infinitely many stages with a long length of agreement, contradicting Lemma 14.2.3. $\square$

We make a further comment. If we further assumed that $\{V_e\}$ is a c.e. approximation for every e, then the function $(\theta, e) \mapsto \lim_s m_{\theta,e}[s]$ is $(\omega+1)$-c.e.. To see this, suppose s is a stage where we toggled $z_{\theta,e}$. Follow the proof of Lemma 14.2.3 and see that after s, as long as there is no change to the $\emptyset'$ portion of $(\theta^\frown m_{\theta,e}{}^\frown 0^\omega \oplus \emptyset')\restriction d_{\theta,e}$, we only toggle $z_{\theta,e}$ at most 4 times under Case 2 before Case 1 must apply at a good recovery stage. The second paragraph in the proof of Lemma 14.2.3 shows that $z_{\theta,e}$ is never toggled again, unless there is a change to the $\emptyset'$ portion of $(\theta^\frown m_{\theta,e}{}^\frown 0^\omega \oplus \emptyset')\restriction d_{\theta,e}$. Hence, if V_e is c.e., then $z_{\theta,e}$ will be toggled no more than $2d_{\theta,e}$ many times. Since $d_{\theta,e}$ may not converge on every θ, e, it follows that the function $(\theta, e) \mapsto \lim_s m_{\theta,e}[s]$ is $(\omega + 1)$-c.e..

14.3 The failure of the analogues of jump inversion

Towards proving our main theorems, the module $M_{\theta,e}$ will meet requirement R_e, provided that θ is indeed the initial segment of the characteristic function of S. We now show how to combine the modules in such a way that for each e, there is a successful $M_{\theta,e}$ module.

Given a sequence $\{V_e\}$, we apply the previous section to get $m_{\theta,e}, d_{\theta,e}$. We now specify an approximation $S[s]$ by the following. First we order the finite binary strings by the following: $\lambda \prec 1 \prec 0 \prec 11 \prec 10 \prec 01 \prec 00 \prec 111 \prec \cdots$. Hence $\prec$ refers to the ordering obtained by first considering increasing length, and then reverse lexicographic ordering. For each η we will have an associated binary string θ_η, and the corresponding $z_{\theta_\eta,|\eta|}$ as defined in module $M_{\theta_\eta,|\eta|}$. That is, $z_{\theta_\eta,|\eta|} = |\theta_\eta|$. For convenience, we will let z_η denote $z_{\theta_\eta,|\eta|}$. We will arrange it so that for $\eta' \prec \eta$, we have $z_{\eta'} < z_\eta$. Basically z_η serves as a pointer, and points to a location of S where $S(z_\eta)[s]$ will be approximated by the digits of $m_{\theta_\eta,|\eta|}$. Each η codes a guess as to the membership of $z_{\eta'}$ in S for $\eta' \prec \eta$. We will have θ_η represent the η-guess as to the correct initial segment $S\restriction|\theta_\eta|$. As we give the stage by stage construction of S, we will move the pointers z_η, but each z_η will only be moved finitely often. Although z_η is defined to be the length of θ_η, in practice we will define z_η first, and later define θ_η. At every stage s, if y is not being pointed at (i.e. $y \neq z_\eta$ for any η), then we will have $S(y)[s] = 0$.

We give a few notations to be used. Define T_s to be a string of finite length, which can be thought of as the current approximation to the "true strategies". Loosely speaking, only those z_η where strategy $\eta \subset T$ will be the important ones; the other z_η with η not on T are just red herrings; they are the artifacts produced by our wrong guesses. T_s is defined inductively by: $T_s(n) = S(z_{T_s \restriction n})[s]$. Proceed this way until we hit the first undefined z_-.

At stage s to *read the next digit of* $m_{\theta,e}$ means to do the obvious thing: If this is the first time we encounter this instruction then we output $m_{\theta,e}(0)$. Otherwise output $m_{\theta,e}(k+1)$ where $m_{\theta,e}(k)$ was the previous digit read by the construction.

Construction of S: At $s = 0$ make every z_η, θ_η undefined. At stage $s > 0$, only finitely many z_η, θ_η have been defined at the end of stage $s - 1$. Go through all such η in increasing order, and for each we (inductively) update θ_η and specify $S(z_\eta)[s]$. For z_λ we let $\theta_\lambda = \lambda$ and set $S(z_\lambda)[s] =$ the next digit of $m_{\theta_\lambda,0}$.

Now assume that $S(z_{\eta'})[s]$ has been defined for all $\eta' \prec \eta$. We define θ_η as follows. For $y < z_\eta$ such that $y \neq z_{\eta'}$ for any η', set $\theta_\eta(y) = 0$. If $y < z_\eta$ is such that $y = \eta'$, then necessarily $\eta' \prec \eta$. If η' is lexicographically to the right of $\eta \restriction |\eta'|$ then set $\theta_\eta(z_{\eta'}) = 0$. If η' is left of $\eta \restriction |\eta'|$ then set $\theta_\eta(z_{\eta'}) = S(z_{\eta'})[s]$. Otherwise $\eta' = \eta \restriction |\eta'|$ and we let $\theta_\eta(z_{\eta'}) = \eta(|\eta'|)$. Next we define $S(z_\eta)[s]$ by the following. Note first of all that $T_s \restriction |\eta|$ can be evaluated at this point. If $T_s \restriction |\eta| = \eta$ then we let $S(z_\eta)[s]$ be the next digit of $m_{\theta_\eta,|\eta|}$. If $T_s \restriction |\eta|$ is left of η then let $S(z_\eta)[s] = 0$. Otherwise if $T_s \restriction |\eta|$ is right of η we let $S(z_\eta)[s] = S(z_\eta)[s-1]$.

If some $d_{\theta_\eta,|\eta|}$ has converged at stage s, we make all $z_{\eta'}, \theta_{\eta'}$ undefined for all $\eta' > \eta$ and go to the next stage. Otherwise the above stops naturally when we find some least η with z_η not defined at stage $s - 1$. We then pick a fresh value for z_η and set $S(z_\eta)[s] = 0$.

Finally let $S(x) = \liminf_s S(x)[s]$. It is clear that z_η eventually settles on a final value for each η, and also that $|T_s| \to \infty$. Let T be the leftmost path specified infinitely often by T_s. We first show that T actually reflects the correct η's:

Lemma 14.3.1. *For every $\eta \subset T$, we have that θ_η eventually settles, $\theta_\eta \subset S$ and $S(z_\eta) = T(|\eta|) = \liminf m_{\theta_\eta,|\eta|}$.*

Proof. We proceed inductively on $|\eta|$. The statement clearly holds if $|\eta| = 0$ so take $|\eta| > 0$. After z_η settles, the value of θ_η and also $S \restriction z_\eta$ will be decided on the places $\{z_{\eta'} \mid \eta' \prec \eta\}$. There are three cases. If η' is right of $\eta \restriction |\eta'|$ then $\theta_\eta(z_{\eta'})$ is always

0, while at infinitely many stages s, $T_s \supset \eta$ which makes $S(z_{\eta'})[s] = 0$ infinitely often. If η' is left of $\eta\restriction|\eta'|$ then T_s is right of η' at every stage after some s_0. Hence $S(z_{\eta'})[s] = S(z_{\eta'})[s_0]$ for all $s > s_0$ and also $\theta_\eta(z_{\eta'})$ will agree with $S(z_{\eta'})[s_0]$. Finally if $\eta' \subset \eta$ then inductively let $\theta_{\eta'}$ be the limit. It is easy to see that the value of $\theta_\eta(z_{\eta'}) = \eta(|\eta'|) = T(|\eta'|) = S(z_{\eta'})$. Hence θ_η eventually settles and $\theta_\eta \subset S$.

Since η is on T, for almost all s we have T_s is right of η (where $S(z_\eta)[s]$ is unchanged from the previous stage) or $T_s \supset \eta$ (in which the next digit of $m_{\theta_\eta,|\eta|}$ is read). Hence $S(z_\eta) = \liminf m_{\theta_\eta,|\eta|}$. To see that this value is the same as $T(|\eta|)$, observe that $T(|\eta|) = \liminf\{T_s(|\eta|) \mid T_s \supset \eta\} = \liminf\{S(z_\eta)[s] \mid T_s \supset \eta\} = \liminf m_{\theta_\eta,|\eta|}$. □

Lemma 14.3.2. *For every e, if V_e is Δ^0_2, then $V'_e \not\equiv_{wtt} S \oplus \emptyset'$.*

Proof. Without loss of generality we assume that $V'_e = \Delta_e^{S\oplus\emptyset'}$ and $S = \Gamma_e^{V'_e}$ with use bounded by δ_e, γ_e (which are total). Let $\eta = T\restriction e$. By Lemmas 14.2.4 and 14.3.1 $S(z_\eta) = \lim m_{\theta_\eta,e}$ where $\theta_\eta \subset S$. Since $d_{\theta_\eta,e}\downarrow$, the initialization in the construction of S ensures that in fact $(S \oplus \emptyset')\restriction d_{\theta_\eta,e} = (\theta_\eta{}^\frown \lim m^\frown 0^\omega \oplus \emptyset')\restriction d_{\theta_\eta,e}$. A contradiction to the last condition of (P2) follows. □

We now obtain as corollaries, the following three statements.

Theorem 14.3.3. *For any computable sequence of Δ^0_2 sets $\{V_e\}_{e\in\mathbb{N}}$ (given by their Δ^0_2 indices), there exists a Δ^0_2 set $S \geq_{tt} \emptyset'$ such that for every e, $V'_e \not\equiv_{wtt} S$.*

Proof. Apply the results of the past two sections, then $S \oplus \emptyset'$ is the desired set. Note that S is Δ^0_2 because of (P2) and the fact that it is easy to prove that $\{T_s\}$ itself is a Δ^0_2 approximation. □

Theorem 14.3.4. *There exists an $\omega + 1$-c.e. set $S >_{tt} \emptyset'$ such that there is no c.e. set A with $A' \equiv_{wtt} S$.*

Proof. Theorem 14.3.3 gives a us a Δ^0_2 set S. To see that S can be made $\omega + 1$-c.e., use the fact that every module will reach a limit and modify the construction of S slightly to ensure that each time $S\restriction z_\eta[s]$ changes we also reset z_η. It is not hard to see that the ensuing approximation for S will be a $\omega + 1$-c.e.. We sketch the reason why, and leave the details to the reader. The value $S(z_\eta)[s]$ depends directly on the value of $T_s \restriction |\eta|$, which in turn depend on $S(z_\nu)[s]$ where $\nu < \eta$. As long as $S \restriction z_\eta[s]$ remains unchanged, we will either output 0 for $S(z_\eta)[s]$, or the digits of $m_{\theta_\eta,|\eta|}$. θ_η will also not change as long as $S \restriction z_\eta[s]$ remains fixed. Hence the number of changes in $S(z_\eta)$ is at most the number of flips in $m_{\theta_\eta,|\eta|}$ (until z_η is cancelled). This number can be computed by the comments after Lemma 14.2.4. □

We point out that in Theorem 14.3.3, each V_e is a Δ^0_2-approximation. To diagonalize every Δ^0_2 set we need to consider all possible Δ^0_2 approximations:

Theorem 14.3.5. *There exists a Σ^0_2 set $S >_{tt} \emptyset'$ such that there is no Δ^0_2 set A with $A' \equiv_{wtt} S$.*

Proof. Use the list of all possible Δ^0_2 indices. □

Chapter 15

On the degrees of diagonal sets and the failure of the analogue of a theorem of Martin

The work in this chapter is to appear in the Notre Dame Journal of Formal Logic.

15.1 Introduction

The study of computably enumerable (c.e.) sets and their Turing degrees has a long and rich history. In this chapter, we are concerned with several classes of c.e. sets which shows up naturally in several contexts, and in their Turing degrees. In particular these classes arose from the study of automorphisms of the lattice of c.e. sets, as well as in the classification of the sets which are the analogues of the Halting problem. We will first discuss the origins of these classes of sets, and then state the motivations behind our interest in these sets.

The c.e. sets we are concerned with in this chapter are the analogues of maximal(max) and hyperhypersimple(hhs) sets, when we replace the congruence $=^*$ (finite symmetric difference) by a more general congruence relation $=^{c.e.}$ were $A =^{c.e.} B$ iff the symmetric difference of A and B is computably enumerable. Formally, we have:

Definition 15.1.1. A weak array $\{V_x\}_{x\in\mathbb{N}}$ is a sequence of uniformly c.e. sets, i.e. there is some computable f such that $V_x = W_{f(x)}$ for all x. A sequence of sets is disjoint if the sets are mutually disjoint. We say that a c.e. set A is *semi-hyperhypersimple* if for every weak array of disjoint sets $\{V_x\}_{x\in\mathbb{N}}$, there is some x such that $V_x - A$ is c.e. A set A is *semi-maximal* if for every pair of disjoint c.e. sets W_i, W_j, we have either $W_i - A$ is c.e. or $W_j - A$ is c.e.

The sets in Definition 15.1.1 were introduced by Kummer in [Kum91]. Kummer showed that the semi-hhs sets arose naturally in degree theory, more specifically, in the study of different versions of the Halting problem. Recall that $F(i,x)$ is a computable numbering of $P := \{\text{all partial computable functions of a single variable}\}$, if F is partial computable (in two variables) and $\{\lambda x F(i,x)\}_{i\in\mathbb{N}} = P$. A Gödel numbering is a computable numbering F where there is an effective way of getting between F and the standard numbering of P. That is, there are computable functions a and b such that for every i, $\lambda x F(i,x) = \varphi_{a(i)}$ and $\varphi_i = \lambda x F(b(i),x)$.

It is not hard to see that not every computable numbering is a Gödel numbering, for instance Friedberg [Fri58b] gave a numbering of P (in fact, of the c.e. sets) without repetition, generally known as a Friedberg numbering. The diagonal (i.e. a version of the Halting problem) induced by Gödel numberings corresponds to the creative sets. Diagonals induced by arbitrary computable numberings are therefore a natural thing to study. Kummer then proved that the diagonal sets which are Halting problems relative to arbitrary computable numberings can be characterized in terms of a class of c.e. sets defined in a seemingly unrelated way:

Theorem 15.1.2. *A c.e. set is a diagonal of some computable numbering iff it is not semi-hyperhypersimple.*

It was also shown in [Kum91] that the semi and plain versions of both maximality and hyperhypersimplicity coincided for simple sets, further demonstrating that these sets are natural extensions of maximality and hyperhypersimplicity.

In [HK94], Hermann and Kummer explored the lattice theoretic properties of semi-max and semi-hhs sets. Let $\mathcal{L}^*(A)$ and $\mathcal{L}^{c.e.}(A)$ denote the lattice of c.e. supersets of A modulo the ideal of supersets B of A such that $B - A$ is finite (and c.e. respectively). They showed that A is semi-hhs iff $\mathcal{L}^{c.e.}(A)$ is a Boolean algebra, generalizing the well-known result of Lachlan [Lac68] that A is hyperhypersimple iff $\mathcal{L}^*(A)$ is a Boolean algebra. They obtained the corollary:

Corollary 15.1.3. *The property of being a diagonal is elementarily lattice theoretic.*

This means that being the Halting problem relative to some arbitrary computable numbering is elementarily lattice theoretic, generalizing a well-known theorem of Harrington that being the Halting problem relative to the standard numbering is elementarily lattice theoretic. They also showed that a coinfinite set A is semi-max iff $|\mathcal{L}^{c.e.}(A)| = 2$; again this result is analogous to the relationship between maximality and the cardinality of $\mathcal{L}^*(A)$. For these reasons, the noncomputable semi-max sets are sometimes called $\mathcal{D}$-maximal, and the noncomputable semi-hhs sets are sometimes referred to as $\mathcal{D}$-hhs in literature. These results further reinforce the idea that the semi-maximal and semi-hyperhypersimple c.e. sets are in many ways analogues of maximal and hyperhypersimple c.e. sets.

Besides being connected to different versions of the halting problem, these c.e. sets also show up in relation to the study of automorphisms of c.e. sets, as well as their connection with the hemi-maximal and Hermann sets. A splitting of a c.e. set A is a pair A_0, A_1 of disjoint c.e. sets such that $A_0 \sqcup A_1 = A$. The splitting is nontrivial if both A_0, A_1 are noncomputable. A *hemi-maximal* (*hemi-hyperhypersimple*) set is half of a nontrivial splitting of a maximal (hyperhypersimple) set. Let **SM**, **SHHS**, **HM** and **HHHM** denote the classes of c.e. Turing degrees which contain respectively a semi-max, semi-hhs, hemi-max and hemi-hhs set. The following shows the relationship between the various degree classes we are interested in; these can be shown easily. The upward arrows follow from the proof that every maximal set is hhs, and the other implications are in Kummer [Kum91].

$$\begin{tikzcd}
\mathbf{HHHS} \arrow[r] & \mathbf{SHHS} \\
\mathbf{HM} \arrow[u] \arrow[r] & \mathbf{SM} \arrow[u]
\end{tikzcd}$$

Harrington proved that the creative sets form a definable orbit which realize only sets of complete m-degrees. He proved that any two creative sets were effectively automorphic. It was realized that simply considering effective automorphisms alone was not enough to discover more orbits. In an influential paper [Soa74], Soare developed what is known today as the automorphism machinery, which gave a general method of constructing Δ^0_3 automorphisms, via the extension lemma. He used it to show that the maximal sets form an orbit, which contains only sets of high degree. Despite Soare's discovery, there are still very few known definable orbits to date, and many known orbits are generated by the splitting of sets which are themselves known to produce orbits. Downey and Stob [DS92] showed that by considering halves of nontrivial splittings of maximal sets (the hemi-maximal sets), one could get a new definable orbit. They also showed in [DS91] that these sets occur in every jump class. These results on the orbit of the hemi-maximal sets lead to an interesting question about the relationship between orbits in $Aut(\mathcal{E})$ and the Turing degrees of sets realizing these orbits. In particular, the hemi-maximal sets give us an example of a definable orbit which is relatively large in the sense that it contains representatives in every possible jump class. Is it possible for there to be an orbit which contains a member of every noncomputable c.e. Turing degree? In [DH96] Downey and Harrington showed that there can be no such fat orbit.

The semi-max and semi-hhs sets show up again in the search for orbits. Interestingly enough, semi-maximality together with strong r-separability gives us a class of sets which forms a new orbit. A *Hermann* set is a noncomputable semi-maximal set which is strongly r-separable, in the sense that every c.e. set disjoint from it can be separated by a computable set in an infinite way. Clearly every maximal, hemi-max and Hermann set is semi-maximal. Cholak, Downey and Hermann [CDH01] showed that the Hermann sets form an orbit, again with representatives from every jump class. These sets play a central role in a recent paper of Cholak, Downey and Harrington [CDH08], where they showed that not every orbit is elementarily definable. In particular, they constructed an orbit which is as complicated as can be.

Various degree theoretic results about the hemi-maximal sets were obtained by Downey and Stob in [DS91, DS92, DS93]. They showed that every high c.e. degree contains a hemi-maximal set, and that **HM** was downward dense. The latter proof is a straightforward modification of the Friedberg maximal set construction. The reason why we cannot combine permitting with a maximal set construction M is due to the following: When we are required to move a marker say m_i with current value $m_i[s] = p$ for the sake of improving its e-state, we might not have permission from the given set. On the other hand if we merely wanted to make M hemi-maximal, we could move the marker m_i by dumping the value p into the other half of the splitting. The only trouble we get by doing this is that we are not able to enumerate p into A anymore. This is not an issue if we only needed to make A noncomputable. This is characteristic of proofs involving sets with hemi (and even semi) properties, and provides a way of exploiting the hemi (or semi) properties. Downey and Stob showed further that not every c.e. degree is in **HM**, that these degrees can be found in every jump class, and that they are nowhere dense in the low c.e. degrees. These results refute a number of conjectures. As a corollary it follows that there is an orbit containing sets of every high degree, and yet does not only contain sets of high degree (unlike the orbit of the maximal sets). It also shows that the degrees of sets in an orbit are not necessarily closed upwards.

Our interest in these sets are generated by Martin's results. Martin [Mar66] in a classic paper showed the coincidence of several classes of degrees. He showed that the c.e. degrees containing a maximal set were exactly the high degrees, that is, those degrees $\boldsymbol{a}$ such that $\boldsymbol{a}' = \mathbf{0}''$. These were also exactly the c.e. degrees containing a hyperhypersimple set, as well as the degrees containing a dense simple set. Our work here is to contribute to the understanding of the Turing degrees of such sets. The combined work of Kummer [Kum91] and Cholak, Downey and Hermann [CDH01] showed that the degrees in **SM** and **SHHS** also satisfy the same degree theoretic facts listed above for **HM**. The similarity in the Turing degree structures led Kummer [Kum91] to ask if his result was a strict improvement, i.e. whether or not **HM** = **SHHS**. On the other hand it is also natural to suggest that an analogue of the classic theorem of Martin holds, also asked by Kummer in [Kum91]. In this chapter we will answer these questions, by demonstrating that the classes **HHHS** and **SM** are incompatible:

Theorem 15.2.1. *There is a c.e. set A which is semi-maximal, and for all c.e. $B \equiv_T A$, B is not hemi-hyperhypersimple.*

Theorem 15.3.1. *There is a c.e. set A which is hemi-hyperhypersimple, and for all c.e. $B \equiv_T A$, B is not semi-maximal.*

As a corollary we have that none of the implications in the previous diagram can be reversed, and that the analogues of Martin's theorem fail in both cases (semi- and hemi-). Despite the fact that all the results known so far about the Turing degrees of these classes suggest that Martin's results can be generalized to these classes, we are able to demonstrate otherwise:

Corollary 15.1.4. *While the degrees of maximal and hyperhypersimple sets are the same, when we prefix hemi- or semi-, they are not the same.*

In Section 15.2 we will construct a c.e. degree in **SM** but not in **HHHS**. The idea of streaming is introduced and explained in Section 15.2, and forms the central ingredient in the proof of the main result in Section 15.3.

Preliminaries. Our notation is standard and follow Soare [Soa87]. The reader is assumed to have familiarity with standard tree arguments. In Section 15.2 the construction is a fairly standard tree argument, with modules replacing the sub-requirements. The construction in Section 15.3 however is a little different in the following sense. In usual tree arguments, once a follower is appointed and later abandoned, it is never reused. In this case we will need to reuse certain followers which have been discarded. This will be coordinated by the top nodes. How and when this is carried out is explained in further detail in Section 15.3.

We will drop the stage number from the notations if the context is clear. Within a stage s of the construction, several actions may take place and will change the value of an expression P when evaluated at different points within stage s. We will use $P[s]$ and also sometimes P_s to denote the value evaluated at the instance within stage s when it is mentioned. We adopt the convention of using uppercase Greek letters to denote functionals, and lowercase Greek letters for their use. That is, $\gamma_e(x)$ refers to the use of $\Gamma_e^{U_e}(x)$, while $\delta_e(x)$ refers to the use of $\Delta_e^A(x)$. The use refers to the largest bit of the oracle accessed during the computation. Since we are only concerned with reductions which are total, we may assume that the use functions are nondecreasing. That is, $\gamma_e(x) \leq \gamma_e(x+1)$ for every x. Also the use of any convergent computation at s is less than s.

15.2 A semi-maximal set whose degree contains no hemi-hyperhypersimple sets

Theorem 15.2.1. *There is a c.e. set $A >_T \emptyset$ which is semi-maximal, and for all c.e. $B \equiv_T A$, B is not hemi-hyperhypersimple.*

15.2.1 Requirements

We build a c.e set A satisfying the requirements :

$$\begin{aligned} \mathcal{R}_e &: \text{ If } X_e \cap Y_e = \emptyset, \text{ then one of } X_e \cap \bar{A} \text{ or } Y_e \cap \bar{A} \text{ is c.e.} \\ \mathcal{Q}_e &: \text{ If } \Gamma_e^{U_e} = A \text{ and } \Delta_e^A = U_e, \text{ and } U_e \cap V_e = \emptyset, \\ &\quad\ \text{then } U_e \sqcup V_e \text{ is not hyperhypersimple.} \end{aligned}$$

Let $\langle \Gamma_e, \Delta_e, X_e, Y_e, U_e, V_e \rangle_{e \in \mathbb{N}}$ be an effective list of all tuples $\langle \Gamma, \Delta, X, Y, U, V \rangle$ such that Γ, Δ are Turing functionals, and X, Y, U, V are c.e. sets. It is clear that A built this way is of neither computable, nor of high degree.

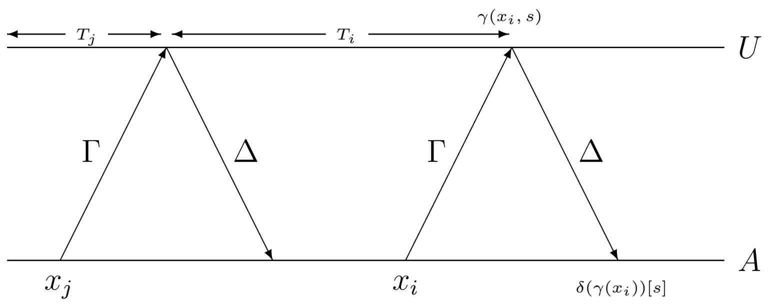
The strategy for requirement $\mathcal{Q}_e$ builds the disjoint weak array $\{T_i\}_{i \in \mathbb{N}}$ witnessing the fact that $U_e \sqcup V_e$ is not hyperhypersimple. The i^{th} *module of* $\mathcal{Q}_e$ ensures that $T_i \cap \overline{U_e \sqcup V_e} \neq \emptyset$, and we will try and ensure that all of the $\mathcal{Q}_e$-modules are successful.

15.2.2 Description of strategy

In this section, we discuss the strategy for each requirement in isolation. Hence, we will drop the subscript and write $\mathcal{R}$ or $\mathcal{Q}$ when discussing the respective strategies.

The basic strategy used to build a set A, which is not of hemi-hyperhypersimple degree can be found in [DS92, DS93]; we describe it here briefly for the benefit of the reader. The strategy for requirement $\mathcal{Q}$ builds $\{T_i\}_{i \in \mathbb{N}}$, and wants to ensure that $T_i \cap \overline{U \sqcup V} \neq \emptyset$ through the i^{th} module, which we will call M_i. The action for each M_i is the following: It starts by picking a follower x_i targeted at A, and waits for $l_\Gamma(s) > x_i$ and $l_\Delta(s) > \gamma(x_i, s)$, where l denotes the respective length of agreement. Once that happens, we would enumerate into T_i all the numbers $y \leq \gamma(x_i, s)$ such that $y \notin U_s \sqcup V_s$, and y is not already in some other T_j. We would now freeze

$A \upharpoonright \delta(\gamma(x_i))[s]$ to preserve the computations in both directions. The picture below summarizes the situation.



Note that M_i would be satisfied temporarily, for it would have made $T_i[s] \cap \overline{U_s \sqcup V_s} \neq \emptyset$. M_i would not need to do anything else until all the numbers we put in T_i had also entered V (they could not have entered U due to the A-restraint we imposed). When that happens, we would then put x_i into A and freeze $A \upharpoonright x_i$. Now some time in future, U would have to respond with a change below $\gamma(x_i, s)$. Since U and V are disjoint, this cannot be a number we had placed in T_i at s. Hence, it has to be a number $< \gamma(x_j, s)$ which would be impossible since we always hold $A \upharpoonright x_i$. To summarize, the action of M_i is to:

1. Pick follower x_i large enough. Wait for $l_\Gamma(s)$ and $l_\Delta(s)$ to grow.

2. Satisfy $T_i[s] \cap \overline{U_s \sqcup V_s} \neq \emptyset$ temporarily, and impose A-restraint.

3. If ever $T_i[t] \cap \overline{U_t \sqcup V_t} = \emptyset$, enumerate x_i into A and impose A-restraint. If we ever enter this state, we would have a global win on the requirement $\mathcal{Q}$.

Since each module only imposes finite A-restraint and enumerates at most once, it is easy to see that all modules of the requirements $\mathcal{Q}_0, \mathcal{Q}_1, \cdots$ can be arranged in the style of a finite injury method, if we only wanted to build such a set A with no additional property. However, the presence of the $\mathcal{R}$-type requirements forces us to have to be careful about the choice of followers x_i for a $\mathcal{Q}$-module, as described below.

We will now discuss the plan to satisfy a single requirement $\mathcal{R}$. Suppose we were trying to construct a maximal set: one maintains a set of markers $a_0[s], a_1[s], \cdots$ which are all pointing at elements in $\bar{A}_s$. We try to maximize the state of each marker a_i, by letting $a_i[s+1]$ occupy the location of some $a_j[s]$ for some $j > i$, if such an action increases the state of the marker a_i.

Things would be very bad for the $\mathcal{Q}$-requirements, if we had to do the above, for each movement of a marker a_i is accompanied by the enumeration of all the values $a_i[s], a_{i+1}[s], \cdots, a_{j-1}[s]$ into A. Fortunately for us, we are allowed to have infinitely many elements in $\bar{A}$ whose states are never maximized. All we need is to ensure that either $X \cap \bar{A}$ or $Y \cap \bar{A}$ is c.e. Note that the set $X \cap \bar{A}$ is 2-c.e. since a number might first enter X, then later enter A. What we want to do, is to prevent the latter from happening infinitely often without our permission, on one of the two sides $X \cap \bar{A}$ or $Y \cap \bar{A}$.

More precisely, it is perfectly all right for us to have an alternating sequence of numbers $n_0 < n_1 < n_2 < \cdots$ such that $n_{2k} \in Y \cap \bar{A}$, and $n_{2k+1} \in X \cap \bar{A}$. What we do is that each time a new n_{2k+2} shows up (i.e. enters Y), we will freeze all numbers strictly between n_{2k} and n_{2k+2} (i.e. keep them out of A). We claim this does the job for $\mathcal{R}$: if we only freeze finitely many intervals, then $Y \cap \bar{A}$ is finite. If we freeze infinitely many intervals, then $X \cap \bar{A}$ is c.e., because if some $(X \cap \bar{A})(n)$ flips from 0 to 1 after n is frozen, then it can never flip back to 0.

As a side note, we remark that the strategy used to build a hemi-maximal (hemi-hyperhypersimple) set is very similar to the strategy used to build a semi-maximal (semi-hyperhypersimple) set, except that we are not allowed any injuries to each separate requirement. For example, in the above, we are allowed to make A-enumerations within a frozen interval, and provided this happens finitely often our semi-maximal requirement is still satisfied. If we were instead trying to make A hemi-maximal by building the other half C of the splitting of a maximal set $A \sqcup C$, then once we freeze an A-interval (x_1, x_2) (which means we dump the entire interval (x_1, x_2) into C to maximize some states), then we cannot allow A to change within the frozen interval (x_1, x_2) anymore.

15.2.3 Interaction between strategies, and the streaming procedure

The construction is to take place on a subtree of $2^{<\omega}$, and we think of the construction tree as growing downwards. To implement the above strategy at an $\mathcal{R}$-node α, we use a process called *streaming*. This term will be used throughout this proof and the next, and is a crucial ingredient in helping $\mathcal{Q}$-requirements in their selection of followers. We maintain a list S^α of numbers which are *streamed by* α where α is an $\mathcal{R}$-node. The way to think about streaming is the following. One pictures $\mathbb{N}$ as a collection of infinite points on a line, and at the beginning $S^\alpha = \emptyset$. We think of S^α as acting as a sort of a gate or barrier situated at the node α. As more numbers get streamed (i.e. enter S^α), these numbers on the line fall through the gate and drop down to the $\mathcal{Q}$-nodes extending $\alpha^\frown\infty$ (which stands for infinitely many α-expansionary stages), who can then pick an appropriate follower from this list. Here $\alpha^\frown\infty$ and $\alpha^\frown f$ are the two immediate successors of α on the tree, and an α-expansionary stage s is a stage where more numbers are streamed by α. At each α-expansionary stage we ensure that there are new, fresh numbers waiting for the gate to open. If there are infinitely many α-expansionary stages, then the gate allows infinitely many numbers through. If a $\mathcal{Q}$-node extends $\alpha_0{}^\frown\infty$ and $\alpha_1{}^\frown\infty$, then it only appoints a follower x if x falls through both gates S^{α_0} and S^{α_1}. More specifically, the rule for streaming is the following. Each time a new large number enters Y and is not yet in A, we put it in S^α. Furthermore,

1. all $\mathcal{Q}$-modules of a lower priority believing that α streams infinitely many numbers (i.e. α-outcome ∞), will pick their x-followers from S^α, while $\mathcal{Q}$-modules believing that α streams finitely many numbers (i.e. α-outcome f) will pick followers larger than $\max S^\alpha[s]$, and are initialized each time S^α grows. This restriction ensures that numbers which are missed out during α-streaming are never later enumerated into A. This corresponds to "freezing the interval" between two streamed numbers, as mentioned previously.

2. if β is an $\mathcal{R}_k$-node below $\alpha^\frown\infty$, then β will only stream numbers which have already been streamed by α. This ensures that $\mathcal{Q}$-nodes below $\beta^\frown\infty$ get a continuous stream of numbers which they may use as followers. It follows

similarly (now taking into account the growth of S^α), that

$$\begin{aligned}|S^\beta| < \infty \quad &\Rightarrow \quad Y_k \cap \bar{A} \text{ is c.e.}\\ |S^\beta| = \infty \quad &\Rightarrow \quad X_k \cap \bar{A} \text{ is c.e.}\end{aligned}$$

The most direct way of arranging the requirements on the construction tree is as follows. There will be levels on the construction tree devoted to the $\mathcal{R}$-requirements which will have two outcomes ∞, f. We also need to put a top node τ for each $\mathcal{Q}$-requirement. Each τ has infinitely many τ-modules, which we denote by $M_0^\tau, \cdots$. Each module is treated as a subrequirement of τ and is assigned to the nodes on a level below $|\tau|$. Note that this layout is slightly different from the actual layout in the formal construction.

We see that having two different modules M_i^τ and M_j^τ of the same $\mathcal{Q}$-node τ act on different $\mathcal{R}$-guesses immediately produces a problem. Take for instance the module M_0^τ assigned to some node σ below $\alpha^\frown\infty$, where α is an $\mathcal{R}$-node. Now at non-α-expansionary stages when S^α does not grow, we might have some τ-daughter node σ' (assigned module M_j^τ) below $\alpha^\frown f$ running its basic strategy. Namely, it will pick follower $x > \max S^\alpha[s]$, and put all the numbers $z \leq \gamma(x)$ into T_j. The danger is that at the next α-expansionary stage, we might have to stream x into S^α. If this happens at each α-expansionary stage for $j > 0$, then σ would not be able to pick any number in S^α as a follower since the array $T_0, T_1, \cdots$ has to be disjoint.

This obstacle is by no means an impossible one, because σ never needs to enumerate anything into A, unless the enumeration also produces a global win for τ. Even though σ is of lower local priority (since σ's position on the tree is lower than the position of α), its global priority is higher than that of α (because σ is assigned a module working for τ, which is above α). Hence it is all right for σ to pick a follower not in S^α and basically ignore the streaming strategy of α. The requirement α will only sustain a finite amount of injury coming in this fashion (at most finitely often for each master $\mathcal{Q}$-node above α).

Finally, we will explain exactly how we intend to arrange the construction. The construction takes place on a subtree of $2^{<\omega}$. Nodes of even length $|\alpha| = 2e$ are assigned the requirement $\mathcal{R}_e$ with two outcomes $\infty <_{left} f$. Nodes of odd length $|\alpha| = 2e+1$ are assigned the requirement $\mathcal{Q}_e$, with only a single outcome 0.

In light of the above discussion, we observe that each $\mathcal{Q}$-node τ only needs to enumerate at most once if it is not injured anymore. Furthermore all τ-modules ignore streaming nodes lying between τ and itself, so it is not necessary for us to spread the τ-modules out on the nodes below τ. Rather, we will run all the τ-modules at the node τ itself. Since each module only requires a finite amount of processing time, we will finish off with one module before moving on to the next module. If at any point in time τ makes an A-enumeration (via one of its modules), it will impose a final A-restraint and be done. We will however, still need to determine local priority amongst the modules of two different $\mathcal{Q}$-nodes $\tau_1 \subset \tau_2$. This is elaborated in Section 15.2.5, and will be used to determine which τ_2-modules are allowed to injure which τ_1-modules.

Notice that the $\mathcal{Q}$-strategies above cannot be modified to construct a semi-maximal set not of semi-hyperhypersimple degree. This is because we will need to modify the $\mathcal{Q}$-strategies to diagonalize all c.e. sets, and the $\mathcal{Q}$-subnodes will now make separate enumerations into A without a global $\mathcal{Q}$-win for us. This is similar to the situation we will face in Theorem 15.3.1. We also cannot modify the $\mathcal{R}$-strategies to construct a hemi-maximal set not of hemi-hyperhypersimple degree, because the $\mathcal{Q}$-strategies now cannot enumerate inside a frozen zone (any interval frozen by $\mathcal{R}$ during streaming is permanently frozen since we have to make A half of a maximal set).

We make a note here about the theorem. The $\mathcal{Q}$-requirements actually show something slightly stronger; they show that $U \sqcup V$ is not even finitely strongly hypersimple. This is because almost every number is enumerated into one of the T_i (unless already in $U \sqcup V$), and each T_i is finite. A characteristic index for T_i can be easily computed.

15.2.4 Notations

Let $\alpha <_{left} \beta$ denote that α is strictly to the left of β, under lexicographic ordering. We write $\alpha \subset \beta$ to mean that α is a strict initial segment of β, and $\alpha \subseteq \beta$ to mean $\alpha \subset \beta$ or $\alpha = \beta$. We say that α is an *$\mathcal{S}$-node*, if α is assigned the requirement $\mathcal{S}$.

For each $\mathcal{Q}_e$-node α, we build a weak array $\{T_i^\alpha\}_{i\in\mathbb{N}}$, and write T^α for $\sqcup_{i\in\mathbb{N}} T_i^\alpha$. The i^{th} module of α, called the *(α, i)-module*, is responsible for making sure that

$T_i^\alpha \cap \overline{U_e \sqcup V_e} \neq \emptyset$ provided the premises in $\mathcal{Q}_e$ hold. We also denote the (α, i)-module by M_i^α. For each $i \in \mathbb{N}$, we let x_i^α denote the follower targeted at A that M_i^α has picked. We also let F_i^α denote the state of M_i^α. This may either be 0 (meaning that the module is pending action), or it can be 1 (meaning that the module has already ensured that $T_i^\alpha \cap \overline{U_e \sqcup V_e} \neq \emptyset$ is at least temporarily satisfied). We introduce a global parameter called $SAT(e)$, for each $e \in \mathbb{N}$. This starts off initially as $SAT(e) = 0$, and when some $\mathcal{Q}_e$-node makes an enumeration into A, we will declare $SAT(e) = 1$ to record the fact that $\mathcal{Q}_e$ has been satisfied (via the falsification of the premise). We will subsequently prevent all other $\mathcal{Q}_e$-nodes at the same level from acting, since these nodes no longer need to act, provided that the appropriate restraint is held.

If α is an $\mathcal{R}_e$-node, we let S^α denote the set of numbers streamed by α, for use by the $\mathcal{Q}$-nodes extending $\alpha^\frown\infty$. A stage s is *α-expansionary*, if either $s = 0$, or else α is visited by the construction at stage s, and there are at least $t + 1$ distinct numbers $y_0, \cdots, y_t$ such that t is the previous α-expansionary stage, and for all i,

1. $y_i > \max S^\alpha[s]$,
2. $y_i \in Y_{e,s} \cap \bar{A}_s$,
3. for all $\mathcal{R}$-nodes β such that $\beta^\frown\infty \subseteq \alpha$, we have $y_i \in S^\beta[s]$.

At each such α-expansionary stage $s > 0$, we will stream the numbers $y_0, \cdots, y_t$ into the set S^α. Each time we stream numbers into S^α, we want to put more numbers into S^α than the previous time. Condition 1 states that we only stream numbers in increasing order. Condition 3 ensures that S^α is a refinement of the set S^β.

When we *initialize an (α, i)-module* for some $\mathcal{Q}$-node α, we mean that we set $x_i^\alpha \uparrow$ and $F_i^\alpha = 0$. To *initialize the $\mathcal{Q}$-node* α, we initialize all α modules, and reset the definition of the weak array $\{T_i^\alpha\}_{i \in \mathbb{N}}$ by setting $T_i^\alpha = \emptyset$ for all i. To *initialize the $\mathcal{R}$-node* α, we set $S^\alpha = \emptyset$.

15.2.5 The local priority ordering

Suppose α is a $\mathcal{Q}_e$-node. The basic strategy of each α-module has both a positive component (in the sense that it might change A), and a negative component (it wants to prevent changes to A). If any α-module changes A, then *every* module of

α will not need to do any more work. Therefore conceptually it makes more sense to think of the α-modules as having only strictly negative action. If any α-module sees a chance to change A, we will let α take over, change A and freeze every α-module. Hence any positive activity is an action of the node α, having priority α (even though the module which was lucky enough to discover this fact had very low local priority). In more complicated $\emptyset'''$-arguments, this would correspond to a subnode σ of some top node τ witnessing the chance of a global win for α.

If α is a $\mathcal{Q}_e$-node, then we define the set of modules with a *lower local α-priority*, to be all the M_i^β for some $\mathcal{Q}$-node $\beta \subset \alpha$ and $i > e$. That is, these modules might have higher "global priority", but we want to arrange for a secondary ordering in which we place almost all of β's modules below α. All the modules with higher α-priority have negative restraint which will not be injured by an A-enumeration made by α. During the construction if α makes an enumeration into A, it will initialize all nodes $\eta \supset \alpha$ since they have lower "global priority", as well as initialize all modules of lower local priority. On the other hand for each M_i^β, there are only finitely many levels in the construction tree which have higher local priority than it (namely, the levels up to $\mathcal{Q}_{i-1}$), so the number of injuries it sustains is finite.

15.2.6 The construction

During the construction, when we say that $\gamma_e(n)[s]\downarrow$ or $\delta_e(n)[s]\downarrow$, we not only mean that the respective computations have converged, but also that $\Gamma_e^{U_e}(m)[s] = A_s(m)$ and respectively, $\Delta_e^A(m)[s] = U_{e,s}(m)$ holds for all $m \leq n$. That is, the respective lengths of agreements are sufficiently long, and if a computation converges without agreement we treat it as being divergent.

At stage $s = 0$, we initialize all nodes, set $SAT(e) = 0$ for all e, and do nothing else. Let $s > 0$. We define the stage s approximation to the true path, TP_s of length s inductively. We say that a node τ is *visited* at stage s, if $TP_s \supset \tau$. Suppose that $\alpha = TP_s \upharpoonright d$ is defined. There are two cases:

1. α is an $\mathcal{R}_e$-node: if s is not α-expansionary, let $TP_s(d) = f$, and do nothing. Otherwise, let $TP_s(d) = \infty$, and enumerate into S^α all the y_i's satisfying conditions 1-3 for an α-expansionary stage above.

2. α is a $\mathcal{Q}_e$-node: let $TP_s(d) = 0$. If $SAT(e) = 1$, do nothing. Otherwise if there is some i such that $F_i^\alpha = 1$, $\delta_e(\gamma_e(x_i^\alpha))[s] \downarrow$, and $T_i^\alpha[s] \cap \overline{U_{e,s} \sqcup V_{e,s}} = \emptyset$, we do the following:

    - enumerate x_i^α into A,
    - set $SAT(e') = 0$ for all $e' > e$,
    - initialize *all nodes* β such that $|\beta| > |\alpha|$,
    - initialize M_i^β for each $\mathcal{Q}$-node $\beta \subset \alpha$ and $i > e$ (i.e. initialize all modules of a lower local priority),
    - set $SAT(e) = 1$, indicating that $\mathcal{Q}_e$ is satisfied.

    Finally, if neither of the above applies, we look for the smallest i such that $F_i^\alpha = 0$, and we take actions for M_i^α - there are two possibilities:

    (a) x_i^α is currently undefined: Check if there is some $x \notin A_s$ satisfying

    i. $x > \delta_e(\gamma_e(x^\star))[s]$ (where we wait for convergence), where $x^\star$ = largest follower picked by any α-module so far,

    ii. $x >$ any number used or mentioned prior to the end of stage s^-, where $s^- \leq s$ is the stage where α was last initialized,

    iii. $x \in S^\beta[s]$ for all $\mathcal{R}$-nodes β such that $\beta^\frown\infty \subseteq \alpha$, and

    iv. $x > \max S^\beta[s]$ for all $\mathcal{R}$-nodes β such that $\beta^\frown f \subseteq \alpha$ or $\beta <_{left} \alpha$.

    If there is such x, we set $x_i^\alpha \downarrow= x$ for the least such x, and initialize all $\mathcal{Q}_{e'}$-nodes $\supset \alpha$, for $e' \geq i$ (i.e. initialize all nodes $\supset \alpha$ which do not have higher local priority. This is important because we do not want the future actions of these nodes to injure M_i^α).

    (b) x_i^α is currently defined: Check if $\delta_e(\gamma_e(x_i^\alpha))[s] \downarrow$. If so, we enumerate all y satisfying $y \leq \gamma_e(x_i^\alpha)[s]$, $y \notin U_{e,s} \sqcup V_{e,s}$, and $y \notin T^\alpha[s]$ into T_i^α. Initialize all $\mathcal{Q}_{e'}$-nodes $\supset \alpha$, for $e' \geq i$. Set $F_i^\alpha = 1$ to indicate that we have temporarily satisfied M_i^α.

This concludes the inductive definition of TP_s. Finally, initialize all nodes $\beta >_{left} TP_s$ and go to the next stage.

15.2.7 Verification

The true path of the construction is defined as usual to be the leftmost path visited infinitely often during the construction.

Lemma 15.2.2. *Each α on the true path is initialized only finitely often. If α is a $\mathcal{Q}$-node, then each of its modules is also initialized finitely often.*

Proof. We restrict our attention to $\mathcal{Q}$-nodes on the true path. Let α_e denote the $\mathcal{Q}_e$-node on the true path. It will be sufficient to prove the following sentences:

φ_e : $\alpha_0, \cdots, \alpha_e$ are initialized only finitely often,

θ_e : only finitely many enumerations can be made into A by a node β of length $|\beta| \leq |\alpha_e|$,

ψ_e : $M_j^{\alpha_i}$ is initialized only finitely often, for $i \leq e$ and $j \leq e+1$,

by induction on e. This follows from the fact that φ_0 is clearly true, and that $\varphi_e \Rightarrow \theta_e \Rightarrow \psi_e \Rightarrow \varphi_{e+1}$. □

Lemma 15.2.3. *All $\mathcal{R}$-nodes on the true path are satisfied.*

Proof. Let α be an $\mathcal{R}_e$-node on the true path, such that $X_e \cap Y_e = \emptyset$. Let o be the true outcome of α, and s_0 be a stage after which $\alpha^\frown o$ is never initialized. Suppose $o = f$ is the true outcome of α, then we claim that $Y_e \cap \bar{A}$ is c.e. by specifying a c.e. set $R =^* Y_e \cap \bar{A}$.

Given $x > x_0 := \max S^\alpha[s_0]$, we enumerate x into R if there is a stage $t > s_0$ such that α is visited and $x \in Y_{e,t} \cap \bar{A}_t$. Furthermore we also require that there is some $\mathcal{R}$-node α^- of maximal length such that $\alpha^{-\frown}\infty \subseteq \alpha$; and for this node α^-, we require that $\max S^{\alpha^-}[t] > x$, and $x \notin S^{\alpha^-}[t]$.

We claim that this enumeration describes $Y_e \cap \bar{A}$. Firstly take $x \in Y_e \cap \bar{A}$ and $x > x_0$. First note that α^- must exist, otherwise $Y_e \cap \bar{A}$ will be finite. So, the only reason why x is never put in R after it shows up in Y_e, must be because $x \in S^{\alpha^-}$. There can only be at most s_0 many such x, since the last α-expansionary stage is before s_0, and consequently $Y_e \cap \bar{A} \subseteq^* R$. Next, we consider an $x \in R$. Such an x is put in R at stage $t > s_0$, and we want to show that x does not enter A after stage t. Since x never enters S^{α^-}, it clearly cannot be enumerated by a $\mathcal{Q}$-node $\beta \supseteq \alpha^-{}^\frown\infty$. Neither can x be enumerated by $\beta >_{left} \alpha^-{}^\frown\infty$ since such an enumeration has to take place after stage t, but at stage t we would have initialized β (since at t we visited α). Thus x cannot enter A after stage t, so that $R \subseteq Y_e \cap \bar{A}$.

Now suppose $o = \infty$ is the true outcome of α. We will build an c.e. set $\tilde{R} = X_e \cap \bar{A}$: we enumerate x into $\tilde{R}$, if there is some stage $t > s_0$ such that $\alpha^\frown\infty$ is visited, $x \in X_{e,t} \cap \bar{A}_t$, and $\max S^\alpha[t] > x$. We claim that $\tilde{R} = X_e \cap \bar{A}$. The direction $\supseteq$ is obvious, and if some x is enumerated in $\tilde{R}$ at stage t, then necessarily x never enters S^α (since $X_e \cap Y_e = \emptyset$), so for reasons similar to the ones above, x cannot enter A after stage t. □

Lemma 15.2.4. *Let M_i^α be a module of a $\mathcal{Q}_e$-node α which is not necessarily on the true path. After F_i^α is set to* 1*, no enumeration can be made into A below $\delta_e(\gamma_e(x_i^\alpha))$ unless either α enumerates, or else M_i^α is initialized in the same stage.*

Proof. Suppose to the contrary that β is a $\mathcal{Q}$-node which does the enumeration. The only possibility is that either $\beta = \alpha$, or we have $\beta \supset \alpha$. In the latter case if β is not of higher local priority, it would be initialized at the same time when F_i^α is set to 1. On the other hand if β is of higher local priority it would initialize M_i^α whenever it makes an enumeration. □

Lemma 15.2.5. *All $\mathcal{Q}$-nodes on the true path are satisfied.*

Proof. Let α be a $\mathcal{Q}_e$-node on the true path, where the premise is true via the sets U_e and V_e. Let s_0 be the first stage after which α is never initialized. $\{T_i^\alpha\}$ is a weak array since it is never reset after s_0, and it is clearly disjoint by step 2b of the construction.

We first claim that $SAT(e)$ never equals 1 after stage s_0. Suppose the contrary. Thus at some stage t_0 we have a module M_i^β for some $\mathcal{Q}_e$-node β making an A-enumeration. We may assume that β is never initialized after t_0. Also we have at some largest stage $t_1 < t_0$, β is visited and flips F_i^β to 1. By Lemma 15.2.4, no enumeration in A below $\delta_e(\gamma_e(x_i^\beta))[t_1]$ can be made until β enumerates x_i^β at stage t_0. After β's action at t_0, no number below x_i^β can ever enter A.

Because of the enumeration made by β, we now have the disagreement $A(x_i^\beta) = 1 \neq A_{t_1}(x_i^\beta) = \Gamma_e^{U_e}(x_i^\beta)[t_1]$. Since we know that $\Gamma_e^{U_e} = A$, there must be a change, say $U_e(p)$, in U_e below $\gamma_e(x_i^\beta)[t_1]$ *after β's action at* t_0. But β's action at stage t_0 was due to the fact that it saw $T_i^\beta[t_0] \cap \overline{U_{e,t_0} \sqcup V_{e,t_0}} = \emptyset$, which means that $p \notin T_i^\beta[t_0]$. Clearly $p \notin U_{e,t_1} \sqcup V_{e,t_1}$, which means that the only reason why it was not put into T_i^β at stage t_1, must be because we already have $p \in T^\beta[t_1]$. Hence $x_i^\beta > \delta_e(p)[t_1]$ by condition 2(a)(i) of the construction, so that $p \notin U_e$, giving a contradiction.

Since $SAT(e)$ is never 1 after s_0, α is not blocked from action each time it is visited after s_0. Fix an i, we need to show that M_i^α eventually succeeds in making $T_i^\alpha \cap \overline{U_e \sqcup V_e} \neq \emptyset$. It will be sufficient to argue that each module state F_i^α eventually settles down to 1. This is because we never put anything into T_i^α unless $F_i^\alpha = 0$, so not every number in $\lim T_i^\alpha$ can enter $U_e \sqcup V_e$, lest $SAT(e)$ is set to 1.

Now we argue inductively that $\lim F_i^\alpha = 1$; assume all $M_{i'}^\alpha$ are eventually in state 1 for $i' < i$. If $F_i^\alpha = 0$ at a sufficiently large stage then M_i^α will receive attention at each visit to α; in fact, all we do at each subsequent visit to α is to give it attention until F_i^α becomes 1. Now a follower will eventually be picked for x_i^α, because conditions 2(a)(i), (ii) and (iv) specify lower bounds for x_i^α which do not increase until x_i^α is picked. Condition (iii) will be satisfied eventually since S^β increases at each visit to α, and since each β only streams numbers into S^β, which are not yet in A. Once x_i^α receives a definition, it will never be cancelled. F_i^α will be set to 1 when $\delta_e(\gamma_e(x_i^\alpha)) \downarrow$ (and subsequently never goes back to 0). $\square$

15.3 A hemi-hyperhypersimple set whose degree contains no semi-maximal sets

Theorem 15.3.1. *There is a c.e. set A which is hemi-hyperhypersimple, and for all c.e. $B \equiv_T A$, B is not semi-maximal.*

15.3.1 Requirements

We build disjoint c.e. sets A and C satisfying the requirements :

$\mathcal{S}_e$: $\{V_x^e\}_{x\in\mathbb{N}}$ is disjoint $\Rightarrow \exists x$ such that $V_x^e - (A \sqcup C)$ is finite.

$\mathcal{Q}_e$: If $\Gamma_e^{U_e} = A$ and $\Delta_e^A = U_e$, build disjoint c.e. sets

X, Y such that both $X - U_e$ and $Y - U_e$ are not c.e.

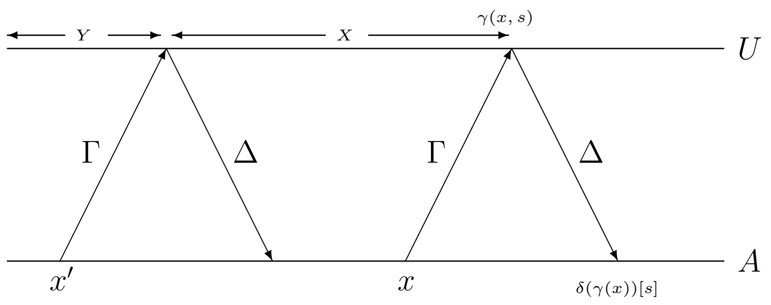
We satisfy requirement $\mathcal{Q}_e$ via the subrequirements

$$\begin{aligned}\mathcal{Q}_{e,2k} &: \text{ If the } \mathcal{Q}_e\text{-premises hold, ensure that } X - U_e \neq W_k.\\ \mathcal{Q}_{e,2k+1} &: \text{ If the } \mathcal{Q}_e\text{-premises hold, ensure that } Y - U_e \neq W_k.\end{aligned}$$

Here, we let $\langle \Gamma_e, \Delta_e, U_e\rangle_{e\in\mathbb{N}}$ be an effective list of all tuples $\langle \Gamma, \Delta, U\rangle$ such that Γ, Δ are Turing functionals, and U is a c.e. set. Also $\{V_x^e\}_{x\in\mathbb{N}}$ stands for the e^{th} uniformly c.e. sequence. We assume a listing where $\{V_x^e\}_{x\in\mathbb{N}}$ is a disjoint sequence for every e. As usual, we use uppercase Greek letters to denote functionals, and lowercase Greek letters for their corresponding use. The $\mathcal{Q}$-requirements ensure that A is noncomputable. Also, C is automatically noncomputable, otherwise A has high degree and thus there will be a maximal set $U \equiv_T A$.

15.3.2 Description of an isolated strategy

We first describe a single strategy used to meet $\mathcal{Q}_e$ and make A not of semi-maximal degree. In this section, we may occasionally drop the subscripts since we are describing strategies in isolation. The first try would be to proceed in roughly the same fashion as the $\mathcal{Q}$-strategies in Theorem 15.2.1. We now build disjoint c.e. sets X, Y and monitor the lengths of agreement at $\mathcal{Q}_e$. The subrequirement $\mathcal{Q}_{e,2k}$ (similarly $\mathcal{Q}_{e,2k+1}$) will ensure that for some p, we have $(X - U_e)(p) \neq W_k(p)$. To do this, we appoint a follower x for the $\mathcal{Q}_{e,2k}$-strategy, and wait for the nested length of agreement to exceed x. We would then enumerate all $p \leq \gamma(x,s)$ into X, provided $p \notin Y$ and freeze $A\restriction\delta(\gamma(x))$ to preserve the computations both ways. We will have a similar diagram as before:



Again, note that $\mathcal{Q}_{e,2k}$ would be satisfied temporarily, until the opponent makes $(X - U_e)\restriction\gamma(x,s) = W_k\restriction\gamma(x,s)$ true. That is, every number we had put in X and not in U_e, would also have entered W_k. When that happens, we put x into A and freeze $A \upharpoonright x$. The resulting U-change will cause a disagreement with W_k that lasts forever. If we followed exactly this then the atomic strategy of each $\mathcal{Q}_{e,2k}$ is the same as before, however the reader will observe a significantly different effect on the rest of the construction: a single $\mathcal{Q}$-module in Theorem 15.2.1 will make an A-enumeration for a $\mathcal{Q}$-win. In this case however, the $\mathcal{Q}$-subrequirements may each enumerate a finite number of times without getting a global $\mathcal{Q}$-win. Hence some modification to the atomic strategy will have to be made.

As the reader might recall, the streaming strategies in Theorem 15.2.1 works only because each streaming node ignores the A-enumerations made by $\mathcal{Q}$-modules which are of a higher global priority than it. This is obviously a problem here, since now there can be infinitely many such enumerations of higher global priority. Fortunately this time we do not need to make A semi-maximal, but hemi-hyperhypersimple. Informally this helps because instead of being allowed only two states (in or out of some c.e. set), we are now allowed three or more different states (being in $V_0, V_1, V_2, \cdots$).

15.3.3 Interaction between two conflicting strategies

The construction will take place on a finite branching tree. For the rest of this section and the next, we consider a $\mathcal{Q}$-node τ of a higher priority than an $\mathcal{S}$-node σ. σ has a rightmost finitary outcome (call it f for the time being, although this has a different label in the formal construction) and two infinitary outcomes to the left (it will be clear later why we have two). The subrequirements of a $\mathcal{Q}$-node τ will be assigned to nodes extending τ. These subrequirement nodes are called τ-daughter nodes. The main difficulty here analogous to the one we outlined in Theorem 15.2.1 is the following: It might be that while σ is waiting for numbers to show up in the array $\{V_x\}$, some τ-daughter node in the region $\supset \sigma^\frown f$ picks a follower x and enumerates all $p \leq \gamma(x)$ into X. We say that the number x is *X-used.* This can happen for different x, i.e. a number of different x might become X-used or Y-used, while σ is waiting. Suppose next, some number shows up in $\{V_x\}$, causing σ to wake up and apply some streaming strategy which takes us to one of the left outcomes, call it i. We cannot control which numbers enter $V_0, V_1, \cdots$, and it may be the case that after streaming, the only surviving numbers left are all X-used (or all Y-used). Then, the

τ-daughter nodes $\supseteq \sigma^\frown i$ will only have numbers which have already been X-used to choose from, and so the τ-daughter nodes working for, say $\mathcal{Q}_{e,2k+1}$ will be unable to appoint a suitable follower. In light of this discussion, it is clear now what we need to incorporate into the construction:

(F1) Firstly, we need to "recycle followers" from right to left, as described in the following scenario: Suppose that a number x has been appointed a follower by some $\alpha \supseteq \sigma^\frown f$. When x undergoes σ-streaming, and assuming it survives the streaming, it will be available for nodes $\supseteq \sigma^\frown i$ to choose from. In this case, α relinquishes control of x (assuming α hasn't enumerated x in yet), and when α is next visited it will appoint another follower larger than x. x will now be available to nodes $\supseteq \sigma^\frown i$ (of the correct type, of course) for the rest of the construction. x might be recycled again a second time if there is another $\mathcal{S}$-node which does the above, but in any case x always migrates from the right to the left.

(F2) Each time σ applies a streaming strategy, it needs to make sure that there are at least two numbers $z_1 \neq z_2$, such that z_1 is not yet X-used and z_2 is not yet Y-used, and both z_1, z_2 survive the streaming. This ensures that both types of τ-daughter nodes $\supseteq \sigma^\frown i$ are able to appoint followers when they are visited.

How should we carry out streaming at σ then, in order to have (F2)? We will have a single finitary outcome, and two infinite outcomes corresponding to two different streaming strategies (B_0 and B_1). We will see that three outcomes is enough. The outer streaming strategy B_0 will look for two numbers $z_1 \neq z_2$, such that z_1 is not yet X-used and z_2 is not yet Y-used, and both z_1, z_2 are in $V_1 \sqcup V_2$. It will then kill all other numbers $\neq z_1, z_2$ by dumping them in C. The inner streaming strategy B_1 will be active (and carries out its own actions) while B_0 is waiting for new numbers to show up in $V_1 \sqcup V_2$, and B_1 is reset each time B_0 finds new numbers for streaming. B_1's actions are the following: it looks for two numbers $z_1 \neq z_2$ such that z_1 is not yet X-used and z_2 is not yet Y-used, and *either* z_1 *or* z_2 is in V_2. It then dumps all other numbers which have not yet been B_0-streamed. Since B_1 is active only when B_0 fails to find the required numbers, it follows that each time B_1 acts with z_1, z_2, and if $z_1 \in V_2$, then either z_2 hasn't appeared in $V_0 \cup V_1 \cup V_2$, or else already we have $z_2 \in V_0$.

It is clear that if B_0 finds infinitely many numbers, then $V_0 - (A \sqcup C)$ is finite, since it kills every number which does not survive streaming. If B_0 gets stuck at some stage but B_1 manages to find infinitely many numbers, then $V_1 - (A \sqcup C)$ is finite because every number which survives B_1-streaming is either already in V_2, or it never enters V_1 (else together with its companion will be streamed by B_0). Lastly if both B_0 and B_1 get stuck, then $V_2 - (A \sqcup C)$ is finite.

15.3.4 Technical considerations

It should be clear that the above works for σ, and also that the τ-daughter nodes below σ on the true path will always have followers of the correct type to choose from. This ensures (F2). Also, it shows us that we can actually prove something slightly stronger. We do not actually need to consider full infinite arrays in the $\mathcal{S}$-requirements, but only that the $\mathcal{S}$-requirements consider triples $\langle V_0, V_1, V_2 \rangle$ of disjoint c.e. sets. This is the best we can do, since if we only consider pairs $\langle V_0, V_1 \rangle$ of disjoint c.e. sets in the $\mathcal{S}$-requirements, then the set $A \sqcup C$ produced would be both r-maximal and hyperhypersimple, and hence maximal. The problem with only having a pair of disjoint c.e. sets is that we have too few states to play with.

There are various technical difficulties in arranging for (F1). For instance, the number x in the diagram in Section 15.3.2 is X-used but not Y-used. However if some number $\leq x'$ enters A in future, then x cannot be used by any τ-daughter node anymore, because upon recovery of the computations, we may now have $\delta(p) > x$ for some $p \in Y$. To make things less messy and improve readability in the formal construction, we propose to organize the construction in the following manner.

Followers appointed by a node assigned the even requirements $\mathcal{Q}_{e,2k}$ will have to be even numbers, while followers appointed by the odd nodes assigned $\mathcal{Q}_{e,2k+1}$ are odd numbers. This eliminates the need to flag a number as being X-used or Y-used. To keep track of whether or not a number x is suitable for use by τ-children, we will flag x as being *τ-confirmed*, when the nested length of agreement is $> x$, and x is currently appointed a follower by some τ-daughter node α. Instead of enumerating numbers into X (or Y if x is odd) only when α is next visited, we will instead enumerate the appropriate numbers into X immediately when x receives τ-confirmation (when τ is visited). This τ-confirmation will tell an $\mathcal{S}$-node $\sigma \supset \tau$ which numbers to consider for streaming; If a number can no longer be used for streaming then the τ-confirmation on it must be removed, to avoid confusing the streaming node σ. To illustrate the interaction between confirmation and streaming, we present the following example:

Suppose $\tau \subset \sigma$ where τ is a $\mathcal{Q}$-node, and σ is an $\mathcal{S}$-node and suppose α is a τ-daughter node such that $\alpha \supseteq \sigma^\frown f$. α starts by appointing a follower x (of the correct parity). At the next τ-expansionary stage s when τ is visited, we will have $\delta(\gamma(x))[s] \downarrow$, and by convention is less than s. We will declare x as τ-confirmed, and perform the following two actions:

1. Enumerate all corresponding $p \leq \gamma(x)$ into X (or Y depending on the parity of x).
2. Cancel all current followers y where $x < y < s$.

This confirmation tells the rest of the nodes below τ, that x can be used as a follower by *any τ-daughter node of the correct parity.* This is true as long as $A \restriction \delta(\gamma(x))$ does not change, and even when σ plays an infinite outcome to the left of f, as long as x survives the streaming, x will be released by α and can continue to be used by other τ-daughter nodes. On the other hand when $A \restriction \delta(\gamma(x))$ changes, then it has to be due to an $A \restriction x$-change (because of (2) above), and so confirmation on x can be removed, and x will no longer be considered.

The reader may also recognize this strategy as being similar to the cancellation and confirmation strategy, used in the construction of a contiguous c.e. degree as presented in [AS84a] and [Dow87]. Note that the requirements actually imply that there cannot be three disjoint elements in $\mathcal{L}^*(A \sqcup C)$, so that the lattice of supersets only consists of four elements. Hence $A \sqcup C$ is quasimaximal. Define a set to be $k+1$-maximal, if it is the intersection of two k-maximal sets, where 1-maximal sets are the maximal sets. Then by nesting more streaming strategies in the $\mathcal{S}$-requirements, one could modify the proof below to show that for any $k \geq 1$, there is a hemi-$(k+1)$-maximal set whose degree does not contain a semi-k-maximal set. There are several possible classes of sets which one may investigate, defined by various finite restrictions on the weak arrays.

15.3.5 Construction tree layout

The construction is organized on a finite branching tree, which grows downwards. For each requirement $\mathcal{R}$, we say that α is an *$\mathcal{R}$-node*, if α is assigned the requirement $\mathcal{R}$. The assignment of nodes is as follows.

Nodes of length $|\alpha| = 2\langle e, 0\rangle$ are assigned the requirement $\mathcal{Q}_e$ with two outcomes $\infty <_{left} f$. The left outcome ∞ stands for infinitely many α-expansionary stages in which the lengths of agreement increase, while outcome f stands for the guess that there are only finitely many α-expansionary stages. In the region below $\alpha^\frown f$, there will be no need for any action to be taken for the subrequirements of $\mathcal{Q}_e$. Nodes of length $|\alpha| = 2\langle e, k+1\rangle$ will be assigned the requirement $\mathcal{Q}_{e,k}$ with a single outcome 2, since the action of each $\mathcal{Q}_{e,k}$ is finitary. The reason why we choose the number 2 is to keep it consistent with the "finitary" outcome of the $\mathcal{S}_e$-nodes, see below.

We say that τ is a *top node*, if τ is a $\mathcal{Q}_e$-node for some e. If $\alpha \supset \tau$ where α is a $\mathcal{Q}_{e,k}$-node and τ is a $\mathcal{Q}_e$-node, we say that α is a *daughter node of* τ. In this case we also refer to τ as the top node of α, denoted by $\tau = \tau(\alpha)$. Furthermore if k is even we call α a (τ, X)-daughter node, otherwise we say it is a (τ, Y)-daughter node. α and β are called *sibling nodes* if they have the same top. Note that we also label τ-daughter nodes $\alpha \supseteq \tau^\frown f$, even though α never needs to act; this is to keep the construction tree layout and labelling of nodes clear and less confusing.

Nodes of odd length $|\alpha| = 2e+1$ are assigned the requirement $\mathcal{S}_e$. The node α has 3 outcomes, labelled $0 <_{left} 1 <_{left} 2$. Outcome n stands for "$V_n^e - (A \sqcup C)$ is finite". Let $\alpha <_{left} \beta$ denote that α is strictly to the left of β, under the usual lexicographic ordering. We write $\alpha \subseteq \beta$ to mean $\alpha \subset \beta$ or $\alpha = \beta$. As mentioned above, $\mathcal{Q}_e$-nodes are referred to as top nodes. A $\mathcal{Q}_{e,i}$-node will be referred to as a $\mathcal{Q}$-node, while an $\mathcal{S}_e$-node is known as an $\mathcal{S}$-node (when we do not want to be specific about the index e).

15.3.6 Notations

At the $\mathcal{Q}_e$-node τ, we build the disjoint c.e. sets X_τ and Y_τ. We will only concern ourselves with the τ-computations which converge correctly both ways, so we monitor the *nested length of agreement* via $l_\tau[s]$, defined as the largest $x < s$ such that

1. for all $y < x$, we have $\Gamma_e^{U_e}(y)[s] \downarrow = A_s(y)$, and
2. for all $z \leq \gamma_e(x-1, s)$, we have $\Delta_e^A(z)[s] \downarrow = U_{e,s}(z)$.

In particular if $x < l_\tau[s]$ at a stage s, then $\delta_e(\gamma_e(x))[s] \downarrow$, and by restraining $A \upharpoonright \delta_e(\gamma_e(x))[s]$, we will be able to preserve the computations below $\Delta_e^A(\gamma_e(x))[s]$. Hence, U_e cannot change below $\gamma_e(x)[s]$ before we remove the restraint on A, otherwise we would get a τ-win by continuing to hold the same restraint on A. We sometimes write $\gamma_\tau, \delta_\tau, \Gamma_\tau, \Delta_\tau$ in place of $\gamma_e, \delta_e, \Gamma_e, \Delta_e$, to avoid cumbersome notations.

If α is a τ-daughter node, we use $x_\alpha[s]$ to denote the stage s follower that α has appointed, which might be put into A some time in the future to force changes in U_e. If α is a (τ, X)-daughter node, then it appoints even followers (i.e. even values for x_α), while if α is a (τ, Y)-daughter node then it appoints odd followers. This restriction is to help in streaming at $\mathcal{S}$-nodes - an $\mathcal{S}$-node σ will make sure that there is a continuous stream of numbers both even and odd, for use by nodes below.

A stage s is τ-*expansionary*, if either $s = 0$, or else τ is visited by the construction at stage s, and

1. $l_\tau[s] > l_\tau[s^-]$ where s^- is the previous τ-expansionary stage, and
2. $l_\tau[s] > x_\alpha[s]$ for every $\alpha \supseteq \tau^\frown\infty$.

Note that in (2) above, we wait for l_τ to exceed x_α for every $\alpha \supseteq \tau^\frown\infty$, including those α which are not daughters of τ. Undefined values count as 0. At τ-expansionary stages, we will take the least $x = x_\alpha$ in (2), enumerate all numbers y such that $x < y < s$ into C, and declare x as τ-*confirmed*. The purpose of τ-confirming a number x, is to ensure that x can be used as a follower by any (τ, X)-daughter node (supposing x is even, similarly for x odd), so long as it is not yet killed. This feature is necessary so that $\mathcal{S}$-nodes below τ can run their streaming strategies compatible with τ.

Suppose that σ is an $\mathcal{S}_e$-node. There are two parameters, B_0^σ and B_1^σ, corresponding to the outcomes 0 and 1 respectively. These contain numbers which have been successfully streamed by σ. By convention, we fix $B_2^\sigma = \mathbb{N}$. If α is any node, then we let $Avail_\alpha[s] := \cap\{B_i^\sigma[s] \mid \sigma \text{ is an } \mathcal{S}\text{-node such that } \sigma^\frown i \subseteq \alpha\}$. This represents all the numbers which are currently available to α. We say that two numbers are of *different parity* if one of them is even, and the other is odd. At any time there is a basic restraint that applies to σ. Denote this by $r_\sigma[s] :=$ the least number larger than

1. $\max\{x_\beta[t] \mid (\beta \subset \sigma \ \vee \ \beta <_{left} \sigma) \text{ and } t \leq s\}$ (all current and past followers),
2. s^- where s^- is the previous stage such that $TP_{s^-} <_{left} \sigma$.

That is, any number handled by σ has to be at least larger than r_σ. We say that a number z is *σ-good* at stage s, if the following holds:

1. $z \in \overline{A_s \sqcup C_s} \cap Avail_\sigma$,
2. z is τ-confirmed for every top node τ such that $\tau^\frown\infty \subseteq \sigma$,
3. $z > r_\sigma[s]$.

The stage s approximation to the true path is denoted by TP_s, and will be defined during the construction. The idea is that σ will only consider numbers which are σ-good, for the purpose of streaming. If a number is larger than r_σ but is not σ-good, then it is useless for streaming, and σ will get rid of these numbers.

We state now what it means to *initialize* a node α at stage s. If α is a top node, then we set $X_\alpha = Y_\alpha = \emptyset$, and remove all τ-confirmations. If α is a daughter node, then we set $x_\alpha =\uparrow$. If α is an $\mathcal{S}$-node, then we set $B_0^\alpha = B_1^\alpha = \emptyset$. To *remove all confirmations* on a number z means to remove any τ-confirmation currently on z for every top node τ. To *dump* a number n at a stage s means to enumerate n into C unless $n \in A_s \sqcup C_s$.

15.3.7 The construction

At stage $s = 0$, we initialize all nodes, and do nothing else. Let $s > 0$. We define the stage s approximation to the true path, TP_s of length $< s$ inductively. During any time (in a stage s), if the construction encounters $HALT$, we will immediately cease all further action, end the current stage s and go to stage $s+1$. As usual, we say that a node α is *visited* at stage s, if $TP_s \supset \alpha$. Suppose that $\alpha = TP_s \restriction d$ is defined. We want to state the action for α and specify the outcome taken by α. There are three cases:

1. *α is a $\mathcal{Q}_e$-node*: If s is not α-expansionary, let $TP_s(d) = f$, and do nothing. Otherwise, let $TP_s(d) = \infty$, and do the following in order.

   (a) For every $\beta >_{left} \alpha^\frown\infty$, we do the following: if β is a $\mathcal{Q}$-node, we remove all confirmations on x_β, if it is defined. If β is an $\mathcal{S}$-node, then we remove all confirmations on z for every $z \in B_0^\beta \sqcup B_1^\beta$.

   (b) Initialize every $\beta >_{left} \alpha^\frown\infty$.

   (c) Pick the smallest number z such that $z = x_\beta$ for some $\beta \supseteq \alpha^\frown\infty$, and z is not α-confirmed. If z exists and is even, we enumerate all y satisfying $y \leq \gamma_e(z)[s]$ and $y \notin X_{\alpha,s} \sqcup Y_{\alpha,s}$ into X_α. If z exists and is odd we enumerate all such y into Y_α instead. Finally, declare z as α-confirmed.

   (d) If $\delta_e(\max X_\alpha \sqcup Y_\alpha)[s] \geq s^-$, then $HALT$ where s^- is the previous α-expansionary stage.

2. *α is a $\mathcal{Q}_{e,2k}$-node*: Let $TP_s(d) = 0$. If α is a $\mathcal{Q}_{e,2k+1}$-node, then exactly the same steps described below are to be taken, except that we replace X by Y, and even with odd. If $\alpha \supseteq \tau(\alpha)^\frown f$, do nothing for α. Otherwise there are three subcases, pick the one that applies and take the actions described:

   (a) *x_α is currently undefined*: We need to pick a new follower for x_α by the following. Check if there is some even number $x \notin A_s \sqcup C_s$ satisfying:

      i. For every top node τ such that $\tau^\frown\infty \subseteq \alpha$, either x is currently τ-confirmed, or else $x > \delta_\tau(\max X_\tau \sqcup Y_\tau)[s]$.

      ii. $x > \max\{x_\beta[t] \mid \beta \subset \alpha \ \vee \ \beta <_{left} \alpha \text{ and } t \leq s\}$.

      iii. $x > s^-$, where s^- is the previous stage such that $TP_{s^-} <_{left} \alpha$.

      iv. $x \in Avail_\alpha$.

If there is such x, we set $x_\alpha \downarrow= x$ for the least such x. In any case, initialize all nodes $\beta \supset \alpha$ and $HALT$.

(b) *x_α is currently defined, but $x_\alpha \notin A_s \sqcup C_s$*: As we will see later in Lemma 15.3.2(8), x_α must be $\tau(\alpha)$-confirmed. This is the waiting phase. See if $(X_{\tau(\alpha)} - U_e) \upharpoonright \gamma_e(x_\alpha) = W_k \upharpoonright \gamma_e(x_\alpha)$. If they are not equal, do nothing. Otherwise we do the following:

   i. Enumerate x_α into A.

   ii. For every $z > x_\alpha$, remove all confirmations on z.

   iii. Initialize all nodes $\beta \supset \alpha$ and $HALT$.

(c) *x_α is currently defined, and $x_\alpha \in A_s \sqcup C_s$*: Do nothing, since α has been successful.

3. *α is an $\mathcal{S}_e$-node*: Firstly, we get rid of all the numbers which are not α-good. More specifically, for every $z_1 < z' < z_2$ such that z_1 and z_2 are α-good, z_1 is larger than $\max B_0^\alpha \sqcup B_1^\alpha$, and z' is not α-good, dump z'. Next, there are three subcases, pick the first in the list that applies. Let $s^\star = \max\{s, 1+$ the largest number mentioned so far$\}$.

   (a) *There are α-good numbers z_1, z_2 of different parity, such that $z_i \in (V_1^e \sqcup V_2^e)$ and* $\max B_0^\alpha < z_i < s$ *for both i*: For every number $z \neq z_1, z_2$[1] such that $\max\{B_0^\alpha, r_\alpha\} < z < s^\star$, we dump z. Add z_1, z_2 to B_0^α, and let $TP_s(d) = 0$. Initialize all nodes $\beta >_{left} \alpha^\frown 0$ and set $B_1^\alpha = \emptyset$.

   (b) *There are α-good numbers z_1, z_2 of different parity, such that* $\max B_0^\alpha \sqcup B_1^\alpha < z_i < s$ *for both i, and $z_i \in V_2^e$ for some i*: For every number $z \neq z_1, z_2$ such that $\max\{B_0^\alpha \sqcup B_1^\alpha, r_\alpha\} < z < s^\star$, we dump z. Add z_1, z_2 to B_1^α. Next, we review the quality of numbers in B_1^α: Namely, for every $z \in B_1^\alpha$, such that z is no longer α-good and $z > r_\alpha$, we will dump z. Let $TP_s(d) = 1$. Initialize all nodes $\beta >_{left} \alpha^\frown 1$.

   (c) *Otherwise*: Do nothing, and let $TP_s(d) = 2$.

This concludes the inductive definition of TP_s. If the construction encounters $HALT$ during stage s, then take TP_s to be whatever that was defined so far. Go

[1]Any choice for z_1 and z_2 is fine.

to the next stage. Note that in step 3(a) we require *both of* z_1, z_2 to be in $V_1^e \sqcup V_2^e$, while in step 3(b) we only require *one of* z_1, z_2 to be in V_2^e.

15.3.8 Verification

The true path TP of the construction is defined as usual to be the leftmost path visited infinitely often during the construction. That is, for every n, $TP \upharpoonright n$ is visited infinitely often and $TP_s <_{left} TP \upharpoonright n$ only finitely often. We say that a number z is *currently in use by* α at some stage s, if $z = x_\alpha[s]$ (if α is a $\mathcal{Q}$-node) or $z \in B_0^\alpha \sqcup B_1^\alpha[s]$ (if α is an $\mathcal{S}$-node). During the verification process we will occasionally refer to a certain step in the construction; to reduce confusion we will prepend "S" to the step number. For instance, we write S2(a)(ii) to refer to step 2(a)(ii) of the construction.

The following lemma lists a few facts about the construction. (4) says that any number which is confirmed must be currently in use. (5) tells us that if x_α is confirmed, then that confirmation cannot be removed until α is initialized. (8) says that if some $\mathcal{Q}$-node picks a follower at a τ-expansionary stage, then at the next τ-expansionary stage this follower will receive τ-confirmation (i.e. there is no delay). (11) describes a key fact about confirmation - once a number x is τ-confirmed, then $A \upharpoonright \delta_\tau(\gamma_\tau(x))$ will be held until the confirmation is removed.

Lemma 15.3.2. *In the following, τ is a top node, and σ is an $\mathcal{S}$-node.*

1. *Enumerations into A are made only by $\mathcal{Q}$-nodes; enumerations into C are made only by $\mathcal{S}$-nodes.*

2. *$A \cap C = \emptyset$.*

3. *If $\alpha <_{left} \beta$ or $\alpha^\frown 2 \subseteq \beta$, and z_1, z_2 are in use by α, β respectively, then $z_1 < z_2$.*

4. *If z is currently τ-confirmed and $z \notin A_s \sqcup C_s$, then z must currently be in use by some $\alpha \supseteq \tau^\frown\infty$.*

5. *If $x_\alpha \downarrow$ and is currently τ-confirmed, then $\tau^\frown\infty \subseteq \alpha$. Furthermore, this τ-confirmation on x_α cannot be removed unless α is initialized (either by the same action or before).*

6. *If a $z \in B_i^\sigma$ for some z and $i < 2$, and z is currently τ-confirmed, then either $\tau^\frown\infty \subseteq \sigma$ or $\tau \supseteq \sigma^\frown i$.*

7. *If τ is a top node, then at any time there can be at most one $\beta \supseteq \tau^\frown\infty$ with a follower x_β that is not τ-confirmed.*

8. *Suppose $\tau^\frown\infty \subseteq \alpha$, and α appoints a follower which is currently not τ-confirmed. Then, the follower will be τ-confirmed at the next τ-expansionary stage, provided that α is not initialized in the meantime.*

9. *The true path of the construction exists.*

10. *Each node on the true path is initialized finitely often.*

11. *Once a number z is τ-confirmed at s, then $A \upharpoonright \delta_\tau(\gamma_\tau(z))[s]$ does not change unless either z is enumerated, or the τ-confirmation on z is removed (either by the same action or before).*

Proof. (1)+(2)+(3): The first two are trivial. For (3), observe that z_2 will be considered by β only after α has started using z_1. If α is a $\mathcal{Q}$-node, then it is clear. If α is an $\mathcal{S}$-node, use the fact that α only enumerates a new z in $B_0^\alpha \sqcup B_1^\alpha$ at stage s, if $z < s$.

(4): We argue by induction on the stage number. More specifically we break down each stage s into separate actions by the different nodes on TP_s (this is assumed to be the case in the rest of the verification).

When z receives its τ-confirmation, it must clearly be in use by some $\mathcal{Q}$-node $\supseteq \tau^\frown\infty$. Assume that z is currently in use by some $\alpha \supseteq \tau^\frown\infty$, and now it is β which gets to act at stage s. If $\beta \supset \alpha$ or $\beta >_{left} \alpha$, then β's action will have no effect on α and its parameters. If $\beta <_{left} \alpha$ then α would have been initialized earlier in the stage and z cannot be in use by α. Hence we may assume $\beta \subseteq \alpha$, in the following three cases:

- β is a top node: To have any effect on α we must have $\beta^\frown f \subseteq \alpha$, and all confirmation on z is removed.

- β is a $\mathcal{Q}$-node: If $\beta = \alpha$ then z remains in use by α. So, $\beta \subset \alpha$ and therefore when β acts, we must have $x_\beta \downarrow$ (use the fact that we always halt in S2(a)).

By (3) we have $x_\beta < z$, and so if β is to initialize α at s, it has to also remove all confirmation on z.

- β is an $\mathcal{S}$-node: If $\beta = \alpha$ then the only consideration is when $z \in B_1^\alpha$ and β plays outcome 0. We must have $\max\{B_0^\alpha, r_\alpha\} < z < s$ holds at s, and hence we will either dump z, or we add z to B_0^α (and so z continues to be in use by α). If $\beta \subset \alpha$ then α has to be initialized when β acts at s (else β's action has no effect on the induction), and so $\tau^\frown\infty \subseteq \beta$ (otherwise τ will get initialized and all τ-confirmations are removed). Furthermore we have $i = \alpha(|\beta|) > 0$, and β plays outcome $< i$ after acting at s. We first claim that $\max\{B_{i-1}^\beta, r_\beta\} < z < s^\star$ holds at s. Note that z is in use by α, so it must be that at the point when α appoints z as a follower (if α is a $\mathcal{Q}$-node) or when α streams z (if α is an $\mathcal{S}$-node), z has to be larger than the stage number of the previous visit to the left of α. A second fact to pay attention to is that if β puts z' into B_j^β at stage t, then $z' < t$. These two facts give $z > \max B_{i-1}^\beta[s]$. The fact that $z > r_\beta[s]$ follows by chasing the definition. Finally since the bounds on z are as such, we must have z is either dumped or continues to be in use by β after β acts at s.

(5): Note that in order for a number z to become τ-confirmed, z has to be first appointed a follower by some $\mathcal{Q}$-node $\beta \supseteq \tau^\frown\infty$. Thus we cannot have $\tau <_{left} \alpha$ nor can we have $\tau^\frown f \subseteq \alpha$. On the other hand it is clear that we cannot have $\tau \supset \alpha$ nor $\tau >_{left} \alpha$. To see this, assume $\tau \supset \alpha$ or $\tau >_{left} \alpha$ and consider the following. If α appoints this number z as a follower after z receives τ-confirmation, then α will initialize τ when appointing z and remove the confirmation on z. Else α has to appoint z before z receives τ-confirmation. However when z later receives τ-confirmation we must also have $z = x_\beta$ for some $\beta \supseteq \tau^\frown\infty$, which is not possible by (3). This shows the first part.

We can only remove this τ-confirmation on $z = x_\alpha$, if τ itself is initialized (in which case α is initialized as well), or directly through the actions of some node η later on. We show that when η acts and removes τ-confirmation on z, the same action also initializes α. Suppose η is a top node. Now η will remove all confirmations on z under S1(a). If $z = x_\beta$ then it must be that $\beta = \alpha$ by (3), and so α will be initialized immediately in S1(b). If $z \in B_0^\beta \sqcup B_1^\beta$ then we must have $\beta \subset \alpha$ once again by (3), and considering the stage where α appointed z relative to when β streams z. Hence α will also be initialized immediately in S1(b). If η is a $\mathcal{Q}$-node, then it is easy. η being an $\mathcal{S}$-node is impossible.

(6): We cannot have $\tau <_{left} \sigma^\frown i$ nor can we have $\tau^\frown f \subseteq \sigma$ due to similar reasons as in (5). Also we cannot have $\tau >_{left} \sigma^\frown i$ because only numbers $< s$ are placed in B_i^σ at stage s.

(7): Suppose the contrary, and fix a stage s, τ, and $\beta_1, \beta_2 \supseteq \tau^\frown\infty$ such that both x_{β_1} and x_{β_2} are not τ-confirmed at s. Assume that s is the least stage where this holds for τ. Since we always halt when a $\mathcal{Q}$-node appoints a follower, it follows that these current incarnations of x_{β_1} and x_{β_2} are appointed at different τ-expansionary stages, say $s_1 < s_2 \leq s$, respectively. By (5), it follows that x_{β_1} is not τ-confirmed when τ acts at s_2. Furthermore x_{β_1} cannot be given τ-confirmation at s_2, which means that some $z < x_{\beta_1}$ has to be given τ-confirmation instead. This contradicts the minimality of s.

(8): Suppose that x_α is appointed at stage s, and s^+ is the next τ-expansionary stage. At s^+ we would τ-confirm x_α unless some $z < x_\alpha$ is given τ-confirmation instead, contradicting (7).

(9)+(10): We argue simultaneously by induction. Suppose $\alpha \in TP$ exists, and is initialized finitely often. If α is an $\mathcal{S}$-node then it is clear that one of its outcomes is visited infinitely often. We consider α to be a top node. It is a problem only if there are infinitely many α-expansionary stages. We need to see that there are infinitely many α-expansionary stages where we do not encounter $HALT$ at α. Suppose s is large enough such that the construction halts at α (if s does not exist then we are done).

Let $s^+ > s$ be the next α-expansionary stage. At stage s^+ we claim that there cannot be a number z receiving τ-confirmation under S1(c): If there were, then we would have $z = x_\eta[s^+]$ for some $\eta \supseteq \alpha^\frown\infty$. Since we encountered a $HALT$ at stage s, it follows that η can only have appointed z as its follower before stage s, which means that by (5) and (8), z would be already τ-confirmed by the stage s^+, a contradiction. Hence z doesn't exist, and one can conclude that the value $\max X_\alpha \sqcup Y_\alpha$ is unchanged between s and s^+. We have $\delta_e(\max X_\alpha \sqcup Y_\alpha)[s] < s$, so we will be done if we can show that $A_s \upharpoonright s = A_{s^+} \upharpoonright s$, because then the construction does not halt at S1(d) at

s^+. Between s and s^+, which $\mathcal{Q}$-node β can possibly enumerate below s? Since α is not initialized (we assume s is large enough), it follows that $\beta >_{left} \alpha^\frown\infty$. This is not possible either, since β is initialized at s, and consequently picks its follower larger than s.

Suppose now α is a $\mathcal{Q}$-node. Since α is initialized finitely often, it follows that eventually if $x_\alpha \downarrow$, then it will be final. Hence we would be done if we can show that α does not halt in S2(a) infinitely often. It will also follow that α initializes $\alpha^\frown 2$ finitely often. To complete the induction, we will need to show that if α is visited at a sufficiently large stage s in which $x_\alpha \uparrow$, it will be able to find a suitable follower for appointment.

Let σ be the maximal $\mathcal{S}$-node such that $\sigma^\frown i \subseteq \alpha$ for some $i < 2$. If σ does not exist, then it is clear that α has no problems appointing a follower at s because $Avail_\alpha = \mathbb{N}$. The main trouble that α faces in choosing followers comes from the restriction in $Avail_\alpha$, because it has to conform to streaming strategies from above. Since σ played outcome $i < 2$ when it was visited at s, it follows that there are some σ-good numbers z_1, z_2 which are newly added to B_i^σ. Our task therefore is to show that both z_1 and z_2 satisfy the conditions in S2(a) to be appointed a follower of α - in that case α could then appoint one of z_1, z_2 of the correct parity.

Certainly $z_1, z_2 \notin C_s$ because there are no $\mathcal{S}$-nodes between σ and α. Also $z_1, z_2 \notin A_s$ since α is not initialized. Clearly we have z_1, z_2 satisfies S2(a)(iv). As for S2(a)(ii) and S2(a)(iii), observe that the bounds reach a limit since α is on the true path, so if s is large enough then z_1, z_2 will be larger than the required lower bounds in S2(a)(ii) and S2(a)(iii). Finally we show that z_1, z_2 satisfy S2(a)(i) for τ. If $\tau \subset \sigma$ then z_1, z_2 are τ-confirmed since they are σ-good, and this confirmation cannot be removed as we travel from σ down to α (because z_1, z_2 are newly streamed and thus cannot be in use by anyone yet). So we may assume that $\sigma^\frown i \subset \tau^\frown\infty \subseteq \alpha$. The fact that we did not halt at τ as we travel from σ down to α, means that we have $\delta_\tau(\max X_\tau \sqcup Y_\tau) < s^-$, where s^- is the previous stage where $\sigma^\frown i$ is visited. We have $z_1, z_2 > \max B_i^\sigma[s^-]$ which implies that $z_1, z_2 \geq s^- > \delta_\tau(\max X_\tau \sqcup Y_\tau)$ because we would have dumped all the useless numbers $< s^-$ when σ acted at s^-. Hence, both z_1 and z_2 are available for α to choose from at stage s.

(11): Any A-change has to be effected by some $\mathcal{Q}$-node β which enumerates

$x_\beta < \delta_\tau(\gamma_\tau(z))[s]$. What are the possible positions of β relative to τ? It is not hard to see that $\beta >_{left} \tau^\frown\infty$ is impossible. If $\beta \subset \tau$ or $\beta <_{left} \tau$ then β's action (when it enumerates x_β) also initializes τ. Therefore we must have $\beta \supseteq \tau^\frown\infty$. By (5) and (8), it follows that at the instant when β enumerates x_β, it must be that $y = x_\beta$ is already τ-confirmed. So, we have that both y and z are τ-confirmed. If $y = z$ then it is trivial so we assume $y \neq z$. If y receives τ-confirmation after z does, and since τ is not initialized between the confirmations of z and y, it follows that $y > \delta_\tau(\gamma_\tau(z))[s]$ because $y = x_\beta$ has to satisfy S2(a)(i). Hence we must have that y receives τ-confirmation before z. But then $y < z$ (for similar reasons), and we are done because β will remove all confirmation on z the same time it enumerates y. $\square$

The $\mathcal{Q}$-strategies succeed. Fix a $\mathcal{Q}_{e,2k}$-node α on the true path, and let $\tau = \tau(\alpha)$. Also assume that $\Gamma_e^{U_e} = A$ and $\Delta_e^A = U_e$. Hence there are obviously infinitely many τ-expansionary stages, and so $\alpha \supseteq \tau^\frown\infty$. Clearly $X_\tau \cap Y_\tau = \emptyset$. A similar argument as the one below will follow for (τ, Y)-daughter nodes. By Lemma 15.3.2(10) it follows that x_α will receive a final definition $x_\alpha = x$, at stage s_0, where x is even.

Lemma 15.3.3. *If α, τ, e, k and x are as above, and suppose further that $x \in A \sqcup C$. Then there is some number p such that $p \in U_e \cap X_\tau \cap W_k$.*

Proof. It is not hard to see that after α appoints x as its follower, x cannot be dumped by any $\mathcal{S}$-node, nor can it be enumerated into A by any $\mathcal{Q}$-node (other than α itself): The only nontrivial case to consider is when we have some $\mathcal{S}$-node σ such that $\sigma^\frown i \subseteq \alpha$ which dumps x (for $i = 1$ or 2). In this case, by Lemma 15.3.2(5) and (8), it follows that when σ is next visited after s_0, we must have that x is σ-good. Furthermore x stays σ-good forever and so σ cannot possibly dump x after s_0.

Since $x \in A \sqcup C$, α will eventually enumerate x at some stage $s_1 > s_0$. Let $t < s_1$ be the stage when x receives τ-confirmation. Before the next τ-expansionary stage $s_2 > s_1$, some number $p < \gamma_e(x)[s_1]$ must enter U_e. By Lemma 15.3.2(11) and the fact that both t and s_1 are τ-expansionary stages, we have $\gamma_e(x)[t] = \gamma_e(x)[s_1]$. Suppose for a contradiction that at stage t when x receives τ-confirmation, we already have $p \in X_\tau \sqcup Y_\tau$. It is easy to see that at stage t, we must have $\delta_e(p)[t] < x$ (by considering when p could have been put in $X_\tau \sqcup Y_\tau$). Since x must remain τ-confirmed until stage s_1, it follows that there is no change in $A \restriction x$ between t and s_1 and also between s_1

and s_2. Thus $\delta_e(p)[s_2] = \delta_e(p)[t] < x$, and since p has entered U_e by s_2, it follows that $U_e(p) \neq \Delta_e(p)[s_2]$, a contradiction to the fact that s_2 is τ-expansionary. Hence at stage t we do not already have $p \in X_\tau \sqcup Y_\tau$, and consequently when x is given τ-confirmation at t, we will place p in X_τ, since x is even. Finally, since α takes S2(b) at stage s_1, it follows that $W_k(p) = (X_\tau - U_e)(p) = 1$ holds at s_1. $\square$

It is clear that if $x \notin A \sqcup C$, then $X_\tau - U_e \neq W_k$ since $\gamma_e(x)$ settles. On the other hand if $x \in A \sqcup C$ then $X_\tau - U_e \neq W_k$ by Lemma 15.3.3.

The $\mathcal{S}$-strategies succeed. Fix an $\mathcal{S}_e$-node σ on the true path, with true outcome i, and assume that V_o^e, V_1^e, V_2^e are pairwise disjoint. Let s_0 be large enough so that $\sigma^\frown i$ is never initialized. We say a number y is *σ-excellent*, if there is a stage $t > s_0$ such that y is σ-good at every visit to $\sigma^\frown i$ after t.

Lemma 15.3.4. *There are infinitely many even numbers in B_i^σ, and infinitely many odd numbers in B_i^σ, which are σ-excellent.*

Proof. We consider the even case, a similar argument follows for the odd case. Fix an arbitrary number $M > \lim r_\sigma$. We can then consider indices e and k large enough such that $W_k = \emptyset$, $U_e = A$ and Γ_e, Δ_e are constant on $\mathcal{P}(\mathbb{N})$, and such that the $\mathcal{Q}_{e,2k}$-node α on the true path appoints a final follower $x > M$ for some even x, and $\alpha \supset \sigma^\frown i$. We show that x is the required even number. By Lemma 15.3.3 it follows that $x \notin A \sqcup C$. It is also obvious that $x \in Avail_\sigma$ forever. Also by Lemma 15.3.2(5) and (8) it follows that x will be τ-confirmed for any $\tau^\frown\infty \subseteq \sigma$, and stays τ-confirmed forever. $\square$

There are three cases, we first consider the case when $i = 2$. We claim that $V_2^e - (A \sqcup C)$ is finite. Suppose for a contradiction that there is some $p > \max\{B_0^\sigma \sqcup B_1^\sigma, \lim r_\sigma\}$ and $p \in V_2^e - (A \sqcup C)$. We may assume that $\max B_0^\sigma \sqcup B_1^\sigma < z_1 < p < z_2$ for two σ-excellent numbers z_1, z_2 of different parity from Lemma 15.3.4. Hence p must also be σ-excellent, otherwise p will be dumped. In that case, when the conditions become right, S3(a) or S3(b) will apply to bring us to the left of the true outcome, a contradiction.

Now suppose that $i = 1$. We claim that $V_1^e - (A \sqcup C)$ is finite. Note that $|B_1^\sigma| = \infty$. Suppose once again for a contradiction, that there are $p_2 > p_1 > \max\{B_0^\sigma, \lim r_\sigma\}$

and $p_1, p_2 \in V_1^e - (A \sqcup C)$ exists. Now p_1 and p_2 must both be put in B_1^σ, lest they be dumped. Furthermore p_1 and p_2 have to be σ-excellent, otherwise they will also be dumped after entering B_1^σ. We may assume that p_2 is put in B_1^σ after p_1 shows up in V_1^e. Hence when p_2 was placed in B_1^σ at stage t, there must be a companion number q of opposite parity placed in B_1^σ together with p_2, such that $q \in V_2^e[t]$. If p_1 and p_2 are of the same parity, then p_1 and q are of different parity and S3(a) would apply instead of S3(b) at stage t. So, it must be that p_1 and p_2 are of different parity. In that case when p_2 eventually shows up in V_1^e, then S3(a) would apply to give another contradiction.

Finally, consider the case $i = 0$. We show that $V_0^e - (A \sqcup C)$ is finite. Again note that $|B_0^\sigma| = \infty$. If $p > \lim r_\sigma$ and $p \in V_0^e$, then p cannot be put in B_0^σ, and hence would be dumped when a large enough number goes into B_0^σ.

Bibliography

[AHLM04] K. Artheya, J. Hitchcock, J. Lutz, and E. Mayordomo. Effective strong dimension, algorithmic information, and computational complexity. *Proceedings of the Twenty-First Symposium on Theoretical Aspects of Computer Science (Montpellier, France, March 2527, 2004)*, pages 632–643, 2004.

[AK00] C. Ash and J. Knight. *Computable Structures and the Hyperarithmetical Hierarchy*, volume 144 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 2000.

[And08] B. Anderson. Automorhisms of the truth-table degrees are fixed on some cone. 2008. To appear.

[AS84a] K. Ambos-Spies. Contiguous r.e. degrees. In *Logic Colloquium '83*, volume 1104 of *Springer-Verlag Lecture Notes*, pages 1–37. 1984.

[AS84b] K. Ambos-Spies. An extension of the nondiamond thoerem in classical and α-recursion theory. *Journal of Symbolic Logic*, 49:586–607, 1984.

[ASF88] K. Ambos-Spies and P. Fejer. Degree theoretical splitting properties of recursively enumerable sets. *Journal of Symbolic Logic*, 53(4):1110–1137, 1988.

[ASJSS84] K. Ambos-Spies, C. Jockusch, R. Shore, and R. Soare. An algebraic decomposition of recursively enumerable degrees and the coincidence of several degree classes with the promptly simple degrees. *Transactions of the American Mathematical Society*, 281:109–128, 1984.

[Bar06] G. Barmpalias. Random non-cupping revisited. *J. Complexity*, 22(6):850–857, 2006.

[BDG] G. Barmpalias, R. Downey, and N. Greenberg. K-trivial degrees and the jump-traceability hierarchy. *Proceedings of the American Mathematical Society.* To appear.

[BLN] G. Barmpalias, A. Lewis, and K.M. Ng. The importance of Π^0_1 classes in effective randomness. *Journal of Symbolic Logic.* To appear.

[BLS08a] G. Barmpalias, A. Lewis, and M. Soskova. Randomness, Lowness and Degrees. *Journal of Symbolic Logic*, 73(2):559–577, 2008.

[BLS08b] G. Barmpalias, A. Lewis, and F. Stephan. Π^1_0 classes, LR degrees and Turing degrees. *Annals of Pure and Applied Logic*, 156(1):21–38, 2008.

[BM] M. Bickford and C. Mills. Lowness properties of r.e. sets. *Theoretical Computer Science.* typewritten unpublished manuscript.

[BM07] G. Barmpalias and A. Montalbán. A cappable almost everywhere dominating computably enumerable degree. *Electronic Notes in Theoretical Computer Science*, 167:17–31, 2007.

[BN03] B. Bedregal and A. Nies. Lowness properties of reals and hyperimmunity. *WoLLIC 2003, Electronic Lecture Notes in Theoretical Computer Science*, 84, 2003.

[CDG08] P. Cholak, R. Downey, and N. Greenberg. Strong jump-traceability 1 : The computably enumerable case. *Advances in Mathematics*, 217:2045–2074, 2008.

[CDH01] P. Cholak, R. Downey, and E. Hermann. Some orbits for $\mathcal{E}$. *Annals of Pure and Applied Logic*, 107:193–226, 2001.

[CDH08] P. Cholak, R. Downey, and L. Harrington. The complexity of orbits of computably enumerable sets. *Bulletin of Symbolic Logic*, 14(1):69–87, 2008.

[CDJL05] R. Coles, R. Downey, C. Jockusch, and G. LaForte. Completing pseudo-jump operators. *Annals of Pure and Applied Logic*, 136:297–333, 2005.

[CDW02] P. Cholak, R. Downey, and S. Walk. Maximal contiguous degrees. *Journal of Symbolic Logic*, 67(1):409–437, 2002.

[Cen99] D. Cenzer. Π^0_1 classes in computability theory. In *Handbook of computability theory*, volume 140 of *Stud. Logic Found. Math.*, pages 37–85. North-Holland, Amsterdam, 1999.

[CFLW] D. Cenzer, J. Franklin, J. Liu, and G. Wu. A superhigh diamond in the tt-degrees. In preparation.

[CGM06] P. Cholak, N. Greenberg, and J. Miller. Uniform almost everywhere domination. *Journal of Symbolic Logic*, 71(3):1057–1072, 2006.

[CGS01] P. Cholak, M. Groszek, and T. Slaman. An almost deep degree. *Journal of Symbolic Logic*, 66(2):881–901, 2001.

[Cha75] G. Chaitin. A theory of program size formally identical to information theory. *Journal of the Association for Computing Machinery*, 22:329–340, 1975.

[Cha76] G. Chaitin. Information-theoretical characterizations of recursive infinite strings. *Theoretical Computer Science*, 2:45–48, 1976.

[CS07] J. Cole and S. Simpson. Mass problems and hyperarithmeticity. *Journal of Mathematical Logic*, 7(2):125–143, 2007.

[Deg79] A. Degtev. Reducibilities of tabular type in the theory of algorithms. *Russian Mathematical Survey*, 34(3):155–192, 1979.

[Dem88] O. Demuth. Remarks on the structure of tt-degrees based on constructive measure theory. *Commentationes Mathematicae Universitatis Carolinae*, 29(2):233–247, 1988.

[DGa] R. Downey and N. Greenberg. Strong jump-traceability II : The general case. In preparation.

[DGb] R. Downey and N. Greenberg. Totally ω computably enumerable degrees II: Left c.e. reals. In preparation.

[DG08] R. Downey and N. Greenberg. Turing degrees of reals of positive packing dimension. *Information Processing Letters*, 108(5), 2008.

[DGMN08] R. Downey, N. Greenberg, N. Mikhailovich, and A. Nies. Lowness for computable machines. *Computational Prospects of Infinity, Lecture Notes Series of the Institute for Mathematical Sciences, NUS*, 15:79–86, 2008.

[DGMW06] R. Downey, N. Greenberg, J. Miller, and R. Weber. Prompt simplicity, array computability and cupping. *Proceedings of "Computational Prospects of Infinity", Singapore, Part II : Presented talks*, pages 59–78, 2006.

[DGW07] R. Downey, N. Greenberg, and R. Weber. Totally ω computably enumerable degrees I: Bounding critical triples. *Journal of Mathematical Logic*, 7(2):145–171, 2007.

[DH96] R. Downey and L. Harrington. There is no fat orbit. *Annals of Pure and Applied Logic*, 80:277–289, 1996.

[DH09] R. Downey and D. Hirschfeldt. *Algorithmic Randomness and Complexity.* Springer-Verlag, in preparation, 2009.

[DHMN05] R. Downey, D. Hirschfeldt, J. Miller, and A. Nies. Relativizing Chaitin's halting probability. *J. Math. Log.*, 5(2):167–192, 2005.

[DHNS03] R. Downey, D. Hirschfeldt, A. Nies, and F. Stephan. Trivial reals. *Proceedings of the 7th and 8th Asian Logic Conferences, World Scientific, Singapore*, pages 103–131, 2003.

[DHNT06] R. Downey, D. Hirschfeldt, A. Nies, and S. Terwijn. Calibrating randomness. *The Bulletin of Symbolic Logic*, 12(3):411–491, 2006.

[Dia] D. Diamondstone. Prompt simplicity does not imply superlow cuppable. *Journal of Symbolic Logic.* To appear.

[DJ94] R. Downey and C. Jockusch. Every low boolean algebra is isomorphic to a recursive one. *Proceedings of the American Mathematical Society*, 122 (3):871–880, 1994.

[DJS90] R. Downey, C. Jockusch, and M. Stob. Array nonrecursive sets and multiple permitting arguments. In *Recursion Theory Week*, volume 1432 of *Lecture Notes in Mathematics*, pages 141–174. 1990.

[DJS96] R. Downey, C. Jockusch, and M. Stob. Array nonrecursive sets and genericity. In *Computability, Enumerability, Unsolvability: Directions in Recursion Theory*, volume 224 of *London Mathematical Society Lecture Note Series*, pages 93–104. Cambridge University Press, 1996.

[DL97] R. Downey and S. Lempp. Contiguity and distributivity in the enumerable degrees. *Journal of Symbolic Logic*, 62:1215–1240, 1997.

[DL02] R. Downey and G. LaForte. Presentations of computably enumerable reals. *Theoretical Computer Science*, 284:539–555, 2002.

[dLMSS56] K. de Leeuw, E. Moore, C. Shannon, and N. Shapiro. Computability by probabilistic machines. *Annals of Mathematics Studies*, 34:183–212, 1956.

[DLS93] R. Downey, S. Lempp, and R. Shore. Highness and bounding minimal pairs. *Mathematical Logic Quarterly*, 39:475–491, 1993.

[DN] D. Diamondstone and K.M. Ng. A strengthening of prompt simplicity and superlow cupping. In preparation.

[Dow] R. Downey. The sixth lecture on algorithmic randomness. *Logic Colloquium '06.* To appear.

[Dow87] R. Downey. Δ^0_2 degrees and transfer theorems. *Illinois Journal of Mathematics*, 31:419–427, 1987.

[DR89] R. Downey and J. Remmel. Classification of degree classes associated with r.e. subspaces. *Annals of Pure and Applied Logic*, 42:105–124, 1989.

[DS91] R. Downey and M. Stob. Jumps of hemimaximal sets. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 37:113–120, 1991.

[DS92] R. Downey and M. Stob. Automorphisms of the lattice of recursively enumerable sets: Orbits. *Advances in Math*, 92:237–265, 1992.

[DS93] R. Downey and M. Stob. Splitting theorems in recursion theory. *Annals of Pure and Applied Logic*, 65:1–106, 1993.

[DS04] N. Dobrinen and S. Simpson. Almost everywhere domination. *Journal of Symbolic Logic*, 69:914–922, 2004.

[FHA$^+$06] L. Fortnow, J. Hitchcock, P. Aduri, V. Vinochandran, and F. Wang. Extracting Kolmogorov complexity with applications to dimension zero-one laws. *Proceedings 33rd ICALP*, LNCS 4051:335–345, 2006.

[FMN09] S. Figueira, J. Miller, and A. Nies. Indifferent sets. *Journal of Logic and Computation*, 19(2):425–443, 2009.

[FNS06] S. Figueira, A. Nies, and F. Stephan. Lowness properties and approximations of the jump. *Proceedings of the Twelfth Workshop of Logic, Language, Information and Computation (WoLLIC 2005), Electronic Lecture Notes in Theoretical Computer Science*, 143:45–57, 2006.

[Fri58a] R. Friedberg. A criterion for completeness of degrees of unsolvability. *Journal of Symbolic Logic*, 22:159–160, 1958.

[Fri58b] R. Friedberg. Three theorems on recursive enumeration. *Journal of Symbolic Logic*, 23:309–316, 1958.

[FS08] J. Franklin and F. Stephan. Schnorr trivial sets and truth-table reducibility. *Technical Report TRA3/08, School of Computing, National University of Singapore*, 2008.

[GHN] N. Greenberg, D. Hirschfeldt, and A. Nies. Characterizing the strongly jump traceable sets via randomness. In preparation.

[GM09] N. Greenberg and J. Miller. Lowness for Kurtz randomness. *Journal of Symbolic Logic*, 74(2), 2009.

[GN] N. Greenberg and A. Nies. Benign cost functions and lowness properties. Submitted.

[Har] K. Harris. A characterization of the low$_n$ degrees by escape functions. Submitted.

[Hau19] F. Hausdorff. Dimension und äßeres maß. *Mathematische Annalen*, 79:157–179, 1919.

[HK94] E. Herrmann and M. Kummer. Diagonals and $\mathcal{D}$-maximal sets. *Journal of Symbolic Logic*, 59(1):60–72, 1994.

[HNS07] D. Hirschfeldt, A. Nies, and F. Stephan. Using random sets as oracles. *Journal of the London Mathematical Society*, 75:610–622, 2007.

[Ish97] S. Ishmukhametov. Weak recursive degrees and a problem of Spector. *Recursion theory and complexity (Kazan, 1997)*, 2:81–87, 1997.

[JLY04] C. Jockusch, A. Li, and Y. Yue. A join theorem for the computably enumerable degrees. *Transactions of the American Mathematical Society*, 356:2557–2568, 2004.

[JM85] C. Jockusch and J. Mohrherr. Embedding the diamond lattice in the recursively enumerable truth-table degrees. *Proceedings of the American Mathematical Society*, 94(1):123–128, 1985.

[JS72a] C. Jockusch and R. Soare. Degrees of members of Π^0_1 classes. *Pacific J. Math.*, 40:605–616, 1972.

[JS72b] C. Jockusch and R. Soare. Π^0_1 classes and degrees of theories. *Transactions of the American Mathematical Society*, 173:33–56, 1972.

[JS83] C. Jockusch and R. Shore. Pseudojump operators. I. The r.e. case. *Transactions of the American Mathematical Society*, 275(2):599–609, 1983.

[KH07] B. Kjos-Hanssen. Low for random reals and positive-measure domination. *Proceedings of the American Mathematical Society*, 135(11):3703–3709 (electronic), 2007.

[KHMS06a] B. Kjos-Hanssen, W. Merkle, and F. Stephan. Kolmogorov complexity and the recursion theorem. *STACS 2006*, 3884:149–161, 2006.

[KHMS06b] B. Kjos-Hanssen, J. Miller, and R. Solomon. Lowness notions, measure and domination. 2006. To appear.

[KHN] B. Kjos-Hanssen and A. Nies. Superhighness. To appear.

[KHNS05] B. Kjos-Hanssen, A. Nies, and F. Stephan. Lowness for the class of Schnorr random sets. *Notre Dame Journal of Formal Logic*, 35(3):647–657, 2005.

[KP54] S.C. Kleene and E. Post. The uppersemilattice of degrees of recursive unsolvability. *Annals of Mathematics*, 59:379–407, 1954.

[KS07] A. Kučera and T. Slaman. Lower upper bounds of ideals. 2007. To appear.

[Kuč85] A. Kučera. Measure, Π^0_1-classes and complete extensions of PA. In *Recursion theory week (Oberwolfach, 1984)*, volume 1141 of *Lecture Notes in Mathematics*, pages 245–259. Springer, Berlin, 1985.

[Kuč86] A. Kučera. An alternative, priority-free, solution to Post's problem. In *Mathematical foundations of computer science, 1986 (Bratislava, 1986)*, volume 233 of *Lecture Notes in Comput. Sci.*, pages 493–500. Springer, Berlin, 1986.

[Kuč93] A. Kučera. On relative randomness. *Ann. Pure Appl. Logic*, 63(1):61–67, 1993. 9th International Congress of Logic, Methodology and Philosophy of Science (Uppsala, 1991).

[Kum91] M. Kummer. Diagonals and semihyperhypersimple sets. *Journal of Symbolic Logic*, 56(3):1068–1074, 1991.

[Kum96] M. Kummer. Kolmogorov complexity and instance complexity of recursively enumerable sets. *SIAM Journal of Computing*, 25:1123–1143, 1996.

[Kur81] S. Kurtz. *Randomness and genericity in the degrees of unsolvability.* Ph.D. Dissertation, University of Illinois, Urbana, 1981.

[Lac68] A. Lachlan. On the lattice of recursively enumerable sets. *Transactions of the American Mathematical Society*, 130:1–37, 1968.

[Lac72] A. Lachlan. Embedding nondistributive lattices in the recursively enumerable degress. *Conference in Mathematical Logic, London, 1970, Lecture Notes in Mathematics*, 255:149–177, 1972.

[Lac75] A. Lachlan. A recursively enumerable degree which will not split over all lesser ones. *Annals of Mathematical Logic*, 9:307–365, 1975.

[Lev73] L. Levin. On the notion of a random sequence. *Soviet Mathematics Doklady*, 14:1413–1416, 1973.

[LS75] R. Ladner and L. Sasso. The weak truth table degrees of r.e. sets. *Annals of Mathematical Logic*, 4:429–448, 1975.

[LS89] S. Lempp and T. Slaman. A limit on relative genericity in the recursively enumerable sets. *Journal of Symbolic Logic*, 54:376–395, 1989.

[Lut90] J. Lutz. Category and measure in complexity classes. *SIAM Journal of Computing*, 19:1100–1131, 1990.

[Lut00] J. Lutz. Dimension in complexity classes. *Procceedings to the 15th Conference on Computational Complexity*, pages 158–169, 2000.

[Lut03] J. Lutz. The dimensions of individual strings and sequences. *Information and Computation*, 187:49–79, 2003.

[LWZ00] A. Li, G. Wu, and Z. Zhang. A hierarchy for cuppable degrees. *Illinois Journal of Mathematics*, 44(3):619–632, 2000.

[Maa83] W. Maass. Characterization of the recursively enumerable sets with supersets effectively isomorphic to all recursively enumerable sets. *Transactions of the American Mathematical Society*, 279:311–336, 1983.

[Mar66] D. Martin. Classes of recursively enumerable sets and degrees of unsolvability. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 12:295–310, 1966.

[May02] E. Mayordomo. A Kolmogorov complexity characterization of constructive hausdorff dimension. *Information Processing Letters*, 84:1–3, 2002.

[Mila] J. Miller. Extracting information is hard. *Advances in Mathematics.* To appear.

[Milb] J. Miller. The K-degrees, low for K degrees, and weakly low for K sets. To appear.

[ML66] P. Martin-Löf. The definition of random sequences. *Information and Control*, 9:602–619, 1966.

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

[Moh84] J. Mohrherr. Density of a final segment of the truth-table degrees. *Pacific J. Math*, 115:409–419, 1984.

[Nga] K.M. Ng. Beyond strong jump traceablility. Submitted.

[Ngb] K.M. Ng. On hyper jump traceable sets. In preparation.

[Nie] A. Nies. Superhighness and strong jump traceability. In preparation.

[Nie97] A. Nies. On a uniformity in degree structures. *Complexity, Logic and Recursion Theory, Lecture Notes in Pure and Applied Mathematics, Feb 1997*, pages 261–276, 1997.

[Nie02] A. Nies. Reals which compute little. *CDMTCS Research Report 202, The University of Auckland*, 2002.

[Nie05a] A. Nies. Eliminating concepts. In *Proceedings of the IMS workshop on the Computational Prospects of Infinity, Singapore*, (in press), 2005.

[Nie05b] A. Nies. Lowness properties and randomness. *Advances in Mathematics*, 197:274–305, 2005.

[Nie07] A. Nies. Non-cupping and randomness. *Proceedings of the American Mathematical Society*, 135(3):837–844 (electronic), 2007.

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

[NSW06] K.M. Ng, F. Stephan, and G. Wu. Degrees of weakly computable reals. *Proceedings of "Second Conference on Computability in Europe", CiE 2006, Swansea*, pages 413–422, 2006.

[Odi89] P. Odifreddi. *Classical Recursion Theory. Vol. I.* North-Holland Publishing Co., Amsterdam, 1989.

[Pos44] E. Post. Recursively enumerable sets of positive integers and their decision problems. *Bulletin of the American Mathematical Society*, 50:284–316, 1944.

[Rei04] J. Reimann. Computability and dimension. *Unpublished notes, University of Heidelberg*, 2004.

[Rob71a] R. Robinson. Interpolation and embedding in the recursively enumerable degrees. *Annals of Mathematics*, 93 (2):285–314, 1971.

[Rob71b] R. Robinson. Jump restricted interpolation in the recursively enumerable degrees. *Annals of Mathematics*, 93 (2):586–596, 1971.

[RS08a] J. Reimann and T. Slaman. Measures and their random reals. 2008. To appear.

[RS08b] J. Reimann and T. Slaman. Probability measures and effective randomness. 2008. To appear.

[Sac63a] G. Sacks. *Degrees of Unsolvability.* Princeton University Press, Princeton, N.J., 1963.

[Sac63b] G. Sacks. On the degrees less than $\mathbf{0}'$. *Annals of Mathematics*, 77 (2):211–231, 1963.

[Sac63c] G. Sacks. Recursive enumerability and the jump operator. *Transactions of the American Mathematical Society*, 108:223–239, 1963.

[Sch73] C. Schnorr. Process complexity and effective random tests. *Journal of Computer and System Sciences*, 7:376–388, 1973.

[Sho59] J. Shoenfield. On degrees of unsolvability. *Annals of Mathematics*, 69:644–653, 1959.

[Sim07] S. Simpson. Almost everywhere domination and superhighness. *MLQ. Mathematical Logic Quarterly*, 53(4-5):462–482, 2007.

[Soa74] R. Soare. Automorphisms of the lattice of recursively enumerable sets, Part 1: Maximal sets. *Annals of Mathematics*, 100:80–120, 1974.

[Soa77] R. Soare. Computational complexity, speedable and levelable sets. *Journal of Symbolic Logic*, 42:545–563, 1977.

[Soa82] R. Soare. Automorphisms of the lattice of recursively enumerable sets, Part 2 : Low sets. *Annals of Mathematical Logic*, 22:69–107, 1982.

[Soa85] R. Soare. Tree arguments in recursion theory and the $\emptyset'''$-priority method. *Proceedings of Symposia in Pure Mathematics*, 42:53–106, 1985.

[Soa87] R. Soare. *Recursively enumerable sets and degrees.* Perspectives in Mathematical Logic. Springer-Verlag, 1987.

[Sol75] R. Solovay. Draft of paper (or series of papers) on Chaitin's work, unpublished notes, 215 pages. 1975.

[SS90] R. Shore and T. Slaman. Working below a low$_2$ recursively enumerable degree. *Archive of Mathematical Logic*, 29:201–211, 1990.

[SS91] T. Slaman and R. Solovay. When oracles do not help. *Fourth Annual Conference on Computational Learning Theory*, pages 379–383, 1991.

[SS93] R. Shore and T. Slaman. Working below a high recursively enumerable degree. *Journal of Symbolic Logic*, 58:824–859, 1993.

[Sta93] L. Staiger. Kolmogorov complexity and Hausdorff dimension. *Information and Computation*, 103:159–194, 1993.

[Ste06] F. Stephan. Martin-Löf random and PA-complete sets. In *Logic Colloquium '02*, volume 27 of *Lecture Notes in Logic*, pages 342–348. Association of Symbolic Logic, La Jolla, CA, 2006.

[Sti72] J. Stillwell. Decidability of the "almost all" theory of degrees. *Journal of Symbolic Logic*, 37:501–506, 1972.

[SY06] F. Stephan and L. Yu. Lowness for weakly 1-generic and Kurtz-random. *Theory and applications of models of computation, Lecture Notes in Computer Science*, 3959:756–764, 2006.

[Ter98] S. Terwijn. Computability and measure. *Ph. D Thesis, University of Amsterdam*, 1998.

[TZ01] S. Terwijn and D. Zambella. Algorithmic randomness and lowness. *Journal of Symbolic Logic*, 66:1199–1205, 2001.

[vM19] R. von Mises. Grundlagen der Wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 5:52–99, 1919.

[Zam90] D. Zambella. On sequences with simple initial segments. *ILLC technical report, ML-1990-05, University of Amsterdam*, 1990.