ELEMENTS OF CLASSICAL RECURSION THEORY: DEGREE-THEORETIC PROPERTIES AND COMBINATORIAL PROPERTIES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Mingzhong Cai

August 2011

ELEMENTS OF CLASSICAL RECURSION THEORY: DEGREE-THEORETIC PROPERTIES AND COMBINATORIAL PROPERTIES

Mingzhong Cai, Ph.D.
Cornell University 2011

In Recursion Theory (Computability Theory), we study Turing degrees in terms of their *degree-theoretic properties* and *combinatorial properties*. In this dissertation we present several results in terms of connections either between these two categories of properties or within each category.

Our first main result is to build a strong connection between *array nonrecursive* degrees and *relatively recursively enumerable* degrees. The former is a combinatorial property and the latter is a degree-theoretic one. We prove that a degree is array nonrecursive if and only if every degree above it is relatively recursively enumerable. This result has a corollary which generalizes Ishmukhametov's classification of r.e. degrees with strong minimal covers to the class of n-REA degrees.

Then we produce new connections between minimality and jump classes, both are degree-theoretic. By using more and more complicated structures, we can finally build a minimal cover over a minimal degree (which we call a 2-minimal degree) which is $\mathbf{GH}_1$, and this is the highest jump class we can reach by finite iterations of minimality. This result answers a question by Lewis and Montalbán, and it also answers an old question by Lerman raised in the 80's.

One interesting group of combinatorial properties is the class of *tracing notions*. They are important in classical recursion theory as well as in algorithmic randomness. We study several different notions of traceability and show that within the n-REA degrees these tracing notions are equivalent to other combinatorial properties. We also introduce a new notion of *tree traceability* and study some of its properties.

We end with a new approach in studying proof-theoretic strength of the totality of recursive functions, and prove several interesting theorems about a degree structure induced by a natural provability reducibility relation on recursive functions.

BIOGRAPHICAL SKETCH

Mingzhong started learning English while in his middle school, Wenzhou Foreign Language School, which was founded only one year before he entered. The teachers there were great, and the students were even better. The only things that he didn't like were the big birthday cakes at the parties held in the classroom – as the cream always ended on everyone's face instead of inside everyone's stomach.

He spent three years of high school in Ningbo Xiaoshi Middle School. There he was a member of the Student Academy of Sciences. The members received some funding from the school to buy books, and more importantly, they also got a clubroom – which is very uncommon in Chinese high schools, and which became the Joint Bridge-and-Mahjong Club's perfect hideout.

After four years in Fudan University watching wild cats, Mingzhong came to Cornell to study logic and mathematics. His favorite part of Ithaca life is that he can sit on the dock by Cayuga Lake, enjoying the spectacular lakeside view and enumerating the truths of the world ($0'$) – and in some sense, this is the best thing he can do.

IN MEMORIAM AVORUM

ACKNOWLEDGEMENTS

I want to first give my keenest thanks to my advisor, Professor Richard A. Shore: for his enjoyable lectures on recursion theory, which enlightened me in research; for his meticulous academical guidance, which has helped me throughout my graduate life and will continue to benefit me in the future; and for his great patience in editing my papers, for which I want to say thanks many times.

I would like to thank Professor Anil Nerode and Professor Justin Moore for being my special committee members. Professor Nerode was my initial advisor when I came to Cornell and he guided me a lot in my first year. Professor Moore taught me set theory and model theory, and also introduced to me a number of interesting topics in set theory, one of which motivated some part of my research which finally became the last chapter of this dissertation.

I also owe my thanks to the professors here at Cornell who gave interesting lectures which I enjoyed and from which I learned a lot: Professor Cao, Professor Connelly, Professor Hatcher, Professor Kjos-Hanssen, Professor Khoussainov, Professor Sjamaar and others. Special thanks to Professor James West for handling my Ph.D. Language Test. I also would like to thank the staff members in the math department, especially Donna Smith, who found my application package left somewhere hidden in the mailroom, otherwise I would have had no chance to come to Cornell.

At Cornell, I have some good graduate student friends: Paul Shafer is my academic sibling and he helped a lot organizing a number of our academic travels; I really enjoyed talking with Paul, especially on the fun topic of magic and mathematics; James Worthington is a very warm-hearted senior student; Ho Hon Leung's optimistic attitude towards life and math influenced me quite a lot; and also to Adam, Diana, Henry, Jiamou, Sam, Shisen, thanks!

I attended the Logic Summer School in Singapore in 2010 and learned a lot there. So I want to thank the organizers Professor Chi Tat Chong, Professor Qi Feng, Professor Frank Stephan and Professor Yue Yang for hosting a wonderful summer school, Professor Moti Gitik, Professor Denis Hirschfeldt and Professor Menachem Magidor for wonderful talks, and Professor Theodore Slaman and Professor William Woodin for participating in discussions with students. In addition, Dr. Adam Day was a nice roommate there in Singapore and we are still doing some joint research. Professor Liang Yu and Dr. Wei Wang are very friendly and I really enjoyed talking with them in the lunch time.

The research in this dissertation was partially supported by NSF grants DMS-0554855 and DMS-0852811.

Moreover, I also would like to give sincere thanks to all logicians in history (starting from Aristotle) for their brilliant ideas and their everlasting contributions to this subject.

As I have grown old enough to be independent, my parents are still watching and encouraging me with their greatest support. Our languages are never powerful enough to express my love and respect to them.

Table of Contents

List of Figures

CHAPTER I
INTRODUCTION TO CLASSICAL RECURSION THEORY

I.1 Turing Degrees: Introduction to Degree-theoretic Properties

In classical recursion theory, we study the *Turing degrees*. They are induced by a natural preorder called *Turing reducibility*. A set A is Turing reducible to (or recursive in, or computable from) B ($A \leq_T B$) if there is an effective algorithm deciding whether $x \in A$ based on the membership information from B. This notion also applies to functions f and g by identifying them with their graphs. Two sets are Turing equivalent if they are Turing reducible to each other. A *Turing degree* is an equivalence class of Turing equivalent sets (and functions). A set is *recursive* if it is reducible to the empty set, i.e., computable without extra information. We use $\mathbf{0}$ to denote the degree of recursive sets.

The notion of Turing degrees formalizes the intuitive idea of how hard it is to compute a set, and research in this area has been applied to effective mathematics answering whether there are effective algorithms for certain problems, for example the famous Hilbert's Tenth Problem receives a negative answer by establishing connections between r.e. sets and solutions to Diophantine equations. Recursion theory also has strong connections with other branches of logic and they together produce interdisciplinary subjects such as computable model theory, descriptive set theory and reverse mathematics. We will, in Chapter XII, initiate an interesting subject connecting recursion theory and proof theory, investigating proofs and relative provability strength by recursion-theoretic techniques such as diagonalization and the Recursion Theorem.

I.1.1 $\mathcal{L}_0$: Order-theoretic properties

We start our topics with properties definable in $\mathcal{L}_0 = \{\leq\}$, i.e., definable with only the Turing order. Note that the preorder of Turing reducibility on sets naturally induces a partial order on Turing degrees. This partial order has been the center of the study of classical recursion theory for many years. A lot of facts are known about this partial order, for example, it is not a total order ([KlPo54]) and it does not have a maximal element (via a jump operator, see §I.1.2).

minimality

One of the most intuitive notions in degree theory is minimality, which asserts that certain intervals are empty in the degrees. A degree $\mathbf{b}$ above a degree $\mathbf{a}$ is a *minimal cover* of $\mathbf{a}$ if the interval $(\mathbf{a}, \mathbf{b})$ is empty, i.e., there is no degree strictly between them. A minimal cover of $\mathbf{0}$ is usually called a *minimal degree*. In addition, if we require that the degrees strictly below $\mathbf{b}$ are all below $\mathbf{a}$, then $\mathbf{b}$ is called a *strong minimal cover* of $\mathbf{a}$.

Spector first showed the existence of minimal degrees and minimal covers ([Sp56]) and he asked for a characterization of degrees with strong minimal covers. This became one of the most difficult problems in classical recursion theory and is still beyond our current knowledge. Even more specific questions are also hard to approach. For example, Yates asked whether every minimal degree has a strong minimal cover and we still do not know the answer (see a survey paper by Lewis ([Lew06]) and see also discussions in Chapters V and XI).

The study of minimality in the Turing degrees revealed an important idea, namely *tree constructions*. They turned out to be very useful in, for example, constructing initial segments in the Turing degrees ([Ler83]). We will show a new utilization of trees in Turing degree constructions in Chapter III.

A natural generalization of minimality is the iterations of minimality: We call a minimal degree 1-*minimal* and a minimal cover of an n-minimal degree $n+1$-*minimal*. For example, a degree is 2-minimal if it is a minimal cover of a minimal degree, i.e., two steps away from $\mathbf{0}$. One can think of it as a measurement of how close a degree is to the recursive degree $\mathbf{0}$.

There are classical results which indicates that 1-minimal degrees are "computably unpowerful" in various ways. For example, they are not **PA** (classical result, see definition in §I.2.2), not **ANR** (see definition in §I.2.4 and see also [DJS96]), not r.e. (by, for example, Sacks Splitting Theorem [Sac63]) and not 1-generic (see [Ler83, Theorem IV.2.9]). However, we are able to construct 2-minimal degrees that are "relatively higher" than 1-minimal degrees, for example, a 2-minimal degree which is **ANR** (see Chapter V). In contrast, in terms of r.e. degrees, we know that no n-minimal degree can be r.e. or even above any non-recursive r.e. degree. We still do not know, for example, whether we can find a 2-minimal or n-minimal degree which is **PA**.

To construct 2-minimal degrees in general, we introduced a notion of *tree systems*. Each tree system is, intuitively, a "tree of trees". We will provide several examples of tree system constructions in Chapters V, VI and VII.

cupping property and join property

In a partial order, it is natural to ask whether the join (least upper bound) and the meet (greatest lower bound) exist for pairs of degrees. Note that they are easily definable in $\mathcal{L}_0$. In the Turing degrees, we can define a join $A \oplus B$ of two sets A and B by $[A \oplus B](2n) = A(n)$ and $[A \oplus B](2n+1) = B(n)$, i.e., producing copies of A and B at even and odd positions respectively. It is easy to check that it is well-defined in the degrees and gives the least upper bound for degrees $\mathbf{a}$ and $\mathbf{b}$, which we will denote as $\mathbf{a} \vee \mathbf{b}$. Interestingly, one can show that the meet does not necessarily exist in the Turing degrees (using Spector's Exact Pair Theorem [Sp56]). Such a partial order is called an *upper-semilattice* (usl), i.e., every pair of elements has a join but not necessarily a meet.

Some of the usl-properties are strongly related to the notion of strong minimal covers. For example, a degree $\mathbf{a}$ has the *cupping property* if for every $\mathbf{b} > \mathbf{a}$, there is a $\mathbf{c} < \mathbf{b}$ such that $\mathbf{b} = \mathbf{a} \vee \mathbf{c}$. Any degree with the cupping property naturally fails to have strong minimal covers. A degree $\mathbf{a}$ has the *join property* if for every $\mathbf{b} < \mathbf{a}$, there is a $\mathbf{c} < \mathbf{a}$ such that $\mathbf{a} = \mathbf{b} \vee \mathbf{c}$. It is easy to see that such a degree $\mathbf{a}$ cannot be a strong minimal cover.

Various interesting results about these properties have been established, for example, see [JP78], [DJS96], [GMS04] and [Lewb]. One can also study other join-related properties such as the capping property or join-reducible, etc. It is also possible to investigate more complicated properties involving minimality, for example, Ellison and Lewis ([ELxx]) studied the notion of minimal cupping property, and Lewis has several proposals about different properties relating to a possible definition of the Turing jump ([Lewa], see also §I.1.2 below).

I.1.2 $\mathcal{L}_1$: Turing jump and jump classes

In the Turing degrees, one can define a natural unary operator called the *Turing jump*. For each set X one can define X' as the halting problem for algorithms (Turing machines) with access to (oracle) X. Note that this is also a typical example of *relativizing*, i.e., we can change the oracle to an arbitrary set X. It is not difficult to prove that this operator is well-defined on the degrees, and we use $\mathbf{d}'$ to denote the jump of a degree $\mathbf{d}$. By a standard diagonalization argument one can show that $\mathbf{d}'$ is strictly above $\mathbf{d}$ (therefore there is no maximal Turing degree).

We use $\mathcal{L}_1 = \{\leq, '\}$ to denote the language of order with a jump operator. By the results of Shore and Slaman ([SS99] and [Sh07]), the Turing jump operator is definable in $\mathcal{L}_0$, therefore $\mathcal{L}_1$ and $\mathcal{L}_0$ have the same definable relations. However these two languages are still different in terms of the quantifier complexity for the definitions ([Sh07]). Some properties easily definable in $\mathcal{L}_1$ may have very difficult definitions in $\mathcal{L}_0$. The current easiest definition of the jump (also the double jump) involves coding and is very complicated in quantifier complexity, and it is still an open problem to find natural definitions of the jump.

high/low hierarchy

With the jump operator one can give another measurement of how close a degree is to $\mathbf{0}$, or to $\mathbf{0}'$. Note that the jump operator preserves $\leq$ (but it does not necessarily preserve the strict order $<$), so the jump of a degree $\mathbf{d}$ is at least $\mathbf{0}'$, and if $\mathbf{d}' = \mathbf{0}'$, we say $\mathbf{d}$ is *low*$_1$ ($\mathbf{L}_1$), or simply *low*. Extending this notion, we call a degree $\mathbf{d}$ *low*$_n$ ($\mathbf{L}_n$) if $\mathbf{d}^{(n)} = \mathbf{0}^{(n)}$, where $\mathbf{a}^{(n)}$ denotes the n-th iterated jump of $\mathbf{a}$.

On the other hand, for a degree $\mathbf{d}$ below $\mathbf{0}'$, it is always the case that $\mathbf{d}' \leq \mathbf{0}''$, and if equality holds, we call $\mathbf{d}$ *high*$_1$ ($\mathbf{H}_1$) or simply *high*. Similarly if $\mathbf{d}^{(n)} = \mathbf{0}^{(n+1)}$, then we say $\mathbf{d}$ is *high*$_n$ ($\mathbf{H}_n$).

High/low hierarchy gives a nice layering of the degrees below $\mathbf{0}'$: each of the low classes is downward closed and each of the high classes is upward closed in the degrees below $\mathbf{0}'$. There is also the class of *intermediate* degrees which are strictly between the low$_n$ ones and high$_n$ ones, i.e., they do not collapse with either $\mathbf{0}$ or $\mathbf{0}'$ by any finite iterations of the jump. See [Ler83, Figure IV.1.1] for a diagram.

generalized high/low hierarchy

For degrees that are not necessarily below $\mathbf{0}'$, one can define a *generalized high/low hierarchy*: A degree $\mathbf{d}$ is *generalized low*$_n$ ($\mathbf{GL}_n$) if $\mathbf{d}^{(n)} = (\mathbf{d} \vee \mathbf{0}')^{(n-1)}$, and $\mathbf{d}$ is *generalized high*$_n$ ($\mathbf{GH}_n$) if $\mathbf{d}^{(n)} = (\mathbf{d} \vee \mathbf{0}')^{(n)}$. Together with the high/low hierarchy, these classes are also called *jump classes*.

It is known that the generalized high/low hierarchy is not as nice as the high/low hierarchy in the way of layering the degrees (see [Mo06]). However, many results for the high/low hierarchy generalize to the generalized high/low hierarchy, for example, results about $\mathbf{GH}_1$ degrees and $\overline{\mathbf{GL}_2}$ degrees resemble those of $\mathbf{H}_1$

degrees and $\overline{\mathbf{L}_2}$ degrees (see for example [Ler83] and [ASDWY09]). In particular, it is known that every $\mathbf{GH}_1$ degree bounds a minimal degree ([Joc77]), whereas there is a $\mathbf{GH}_2$ degree which does not ([Ler86]). So the generalized high/low hierarchy has some feature similar as the high/low hierarchy in giving a measurement of the highness and lowness of the degrees.

Jockusch and Posner showed that all minimal degrees are $\mathbf{GL}_2$ ([JP78]), and Lerman ([Ler83, Section IV.3]) observed that by relativizing Jockusch and Posner's theorem it is easy to show that all n-minimal degrees below $\mathbf{0}'$ are $\mathbf{GL}_2$ (therefore $\mathbf{L}_2$). Then it is natural to ask (as Lerman did) whether this is true for all n-minimal degrees in general. We will answer this question negatively in Chapter VI. In fact, we can construct a 2-minimal degree which is in $\mathbf{GH}_1$ (Chapter VII), and it is not difficult to show that $\mathbf{GH}_0$ degrees (i.e., degrees above $\mathbf{0}'$) cannot be n-minimal for any n, so $\mathbf{GH}_1$ is the highest jump class we can reach with finite iterations of minimality.

I.1.3 $\mathcal{L}_2$: Recursive enumeration

r.e. degrees and relatively r.e. degrees

A set A is *recursively enumerable* (r.e.) if there is an effective algorithm listing all the members of A. Relativizing this, a set A is r.e. in B if there is such an effective algorithm listing members of A, given membership information from B. A degree is r.e. if it contains an r.e. set. The structure of the r.e. degrees is the most well-studied subject in classical recursion theory (see survey [Sh99]). It began with Post's problem ([Po44]) asking whether there are incomplete nonrecursive r.e. degrees, and it has provided a number of elegant techniques, namely priority arguments.

Note that most established theorems about the r.e. degrees relativize to an arbitrary oracle. So it is natural to study the *relatively r.e.* (**RRE**) degrees: a degree $\mathbf{a}$ is relatively r.e. if it is r.e. in some $\mathbf{b} < \mathbf{a}$. Joint with Shore, we will, in Chapters II and III, prove a strong connection between **RRE** degrees and **ANR** degrees, which we will define later. In particular, we will show that a degree $\mathbf{a}$ is **ANR** if and only if every degree $\mathbf{b} \geq \mathbf{a}$ is **RRE**. One can view this result as a definition of the **ANR** degrees in the language $\mathcal{L}_2 = \{\leq, ', R\}$ where R stands for a binary r.e. relation: $\mathbf{x}\, R\, \mathbf{y}$ if $\mathbf{x}$ is r.e. in $\mathbf{y}$. It is still a major open question whether this relation R is definable in $\mathcal{L}_0$.

n-REA degrees

By iterating the r.e. relation, one can get the n-**REA** degrees. A degree is 1-**REA** if it is r.e., and a degree is $n+1$-**REA** if it is r.e. in and strictly above an n-**REA** degree. These give natural generalizations of the r.e. degrees (as a degree-theoretic property). In Chapter X we will generalize several theorem about the r.e. degrees to the n-**REA** degrees.

The first one is about the equivalence between *array recursiveness* (which we define later) and *r.e. traceability* (a notion introduced in [Ish99] to study the strong minimal cover problem, see also §I.1.1 and §I.2.5). The second one is about the equivalence between *strong jump traceability* and *strong superlowness* (see definitions in Chapter X), two notions that arose in trying to give a combinatorial characterization of K-*trivials*, an important notion in algorithmic randomness. We also give a new proof of a known result generalizing Arslanov's completeness criterion ([Ars81]). For more information about various tracing properties, see §I.2.5.

I.2 Introduction to Combinatorial Properties

The degree-theoretic properties we reviewed in the previous section are strongly connected to definability. For example, a set A is r.e. if and only if it can be defined by a Σ_1^0 formula in arithmetic (which provides a negative solution to Hilbert's Tenth Problem), and being recursive and being recursive in $\mathbf{0}'$ correspond to respectively Δ_1^0 and Δ_2^0 definitions. So these degree-theoretic notions are "logical" in nature, or in other words, expressible in terms of how complicated it is to define a set. In this section, we want to study the combinatorial properties of degrees. They focus on mathematical properties such as limit and domination (rates of growth), and naturally address the "computing power" of degrees.

I.2.1 Set properties

The study of combinatorial properties in classical recursion theory probably started with set properties ([Po44]). In these notions, we are interested in how sets in a degree interact with different other sets, especially the r.e. ones. Examples of such properties are *simple*, *immune*, *hypersimple* and *hyperimmune*, etc.

There have been some detailed analysis of the definability power of $\mathcal{E} = \{\mathcal{R}, \subset\}$, where $\mathcal{R}$ denotes the collection of r.e. sets. In particular there are full characterizations of which jump classes are definable in $\mathcal{E}$ (see [Epsxx, Section 1.3]). In our scheme, this is a typical example of building connections between degree-theoretic properties (jump classes) and combinatorial properties (set inclusion).

I.2.2 Pointwise properties

FPF degrees

We are also interested in properties of functions at each point. For example, a function f is *diagonally nonrecursive* (*DNR*) if $f(n) \neq \varphi_n(n)$ for each n (see notions in §I.3.2). They are natural examples of functions which are not recursive. A degree is *fixed-point-free* (**FPF**) if it contains a DNR function. The term fixed-point-free comes from the fact that a degree computes a DNR function if and only if it computes a fixed-point-free function, i.e., a function f such that $\varphi_e \neq \varphi_{f(e)}$ for any e (note that the Recursion Theorem asserts that such a function f cannot be recursive).

To list a few theorems about **FPF** degrees: Arslanov's completeness criterion says that an **FPF** r.e. degree has to be complete, i.e., $\geq \mathbf{0}'$ ([Ars81]); Kučera's injury-free solution to Post's problem used his theorem that every **FPF** degree below $\mathbf{0}'$ bounds a nonrecursive r.e. degree ([Ku86]). We will, in Chapter X, give a new proof of a generalization of Arslanov's completeness criterion that every **FPF** n-**REA** degree is complete.

FPF degrees are upward-closed and they represent a class of degrees which are powerful, or similar to $\mathbf{0}'$. One can naturally ask whether there can be a minimal degree which is **FPF** ([Sac85]). This was answered positively by Kumabe in an unpublished paper and a simplification of the proof can be found in [KLxx]. Jointly with Greenberg, we generalize Kumabe and Lewis' construction and build minimality chains with relativized **FPF** properties. A preliminary two-step construction is included in this dissertation as Chapter VIII.

PA degrees

A degree is **PA** if it computes a complete extension of Peano Arithmetic. We think of it as a pointwise property (similar to the **FPF** degrees) because a degree being **PA** is equivalent to it computing a 0-1 valued DNR function. One important property is that every **PA** degree has the cupping property ([Ku94]), and so they do not have strong minimal covers. They also corresponds to degrees that can compute a path in any given nonempty Π^0_1 class.

The **PA** degrees are also upward closed, and represent a class of degrees that are powerful. However, we will in Chapter III give an alternative proof of the existence of a **PA** degree without the join property, which is originally proved in [Lewb]. This contrasts, for example, to the result that all $\overline{\mathbf{GL}_2}$ degrees have the join property ([DGLMxx]). We thank Adam Day for pointing out this corollary.

I.2.3 Limit approximation

limit computable

A function $f(n)$ is *limit computable* if there is a recursive function $\lambda(n, s)$ such that $\lim_{s\to\infty} \lambda(n, s)$ exists and is equal to $f(n)$ for each n. This combinatorial property is strongly connected to a degree-theoretic property by the Shoenfield Limit Lemma ([Shn59]): a degree is below $\mathbf{0}'$ if and only if every function recursive in it is limit computable. The Limit Lemma is used very widely in classical recursion theory, since it bridges two of the most important notions: the Turing jump and limit approximations.

n-r.e. degrees

In a 0-1 valued limit approximation $\lim_{s\to\infty} \lambda(n, s)$, if we can in addition bound the number of changes in the approximation for each n, then we get stronger notions. For example, if we can bound the number of changes in every limit approximation by 1, then the limit we get is an r.e. set, and conversely every r.e. set can be limit computed in such an approximation (assuming that each approximation starts with 0). If the bound is a fixed constant number n, then we get the notion of *n-r.e. sets*. A degree is *n-r.e.* if it contains an n-r.e. set. This is called the Ershov Hierarchy (see [Ars10] for a survey). One can regard the n-r.e. degrees as generalizations of the r.e. degrees as a combinatorial property.

In Chapter IX (which is a joint work with Shore and Slaman), we show several results about the global structure of the n-r.e. degrees. We show that the theory of the n-r.e. degrees is undecidable, and we prove that the class of n-r.e. degrees does not form a Σ_1-elementary substructure of the $n+1$-r.e. degrees. The latter generalizes a result of Yang and Yu from the r.e. case to the n-r.e. case.

I.2.4 Domination

One intuitive idea in computability is that, functions which grow fast are powerful in computing. Given two functions f and g, we say f *dominates* g if $f(x) \geq g(x)$ for all but finitely many x. This is an important notion in many fields of mathematics, especially in analysis. Domination properties also play important roles in classical recursion theory, typically with the two complementary pairs: hyperimmune and hyperimmune-free; array nonrecursive and array recursive, which we will discuss in detail in the following sections. Besides these complimentary pairs, there are also other interesting notions. For example, say a function is *dominant* if it dominates every recursive function. A degree computes a dominant function if and only if its jump is above $\mathbf{0}''$ ([Ma66], see [Ler83, Lemma IV.3.3]). In the generalized high/low hierarchies, the class of $\mathbf{GL_2}$ degrees also corresponds to a domination property: a degree $\mathbf{a}$ is $\mathbf{GL_2}$ if and only if there is a function recursive in $\mathbf{a} \vee \mathbf{0}'$ which dominates every function recursive in $\mathbf{a}$ ([JP78]). We will investigate properties of the degrees which are not $\mathbf{GL_2}$ in Chapter II using this characterization by domination properties and comparing them with the **ANR** degrees.

hyperimmunity

The term "hyperimmune" originally came from a set property, and it turns out to be equivalent to a domination property which is more useful and much easier to understand. A degree is *hyperimmune* if it computes a function which is not dominated by any recursive function; and a degree is *hyperimmune-free* if it is not hyperimmune, i.e., every function recursive in it is dominated by a recursive function.

Hyperimmune and hyperimmune-free degrees have been studied for a long time, and a lot of interesting theorems were discovered (see [MM68], [JSo72] and [Lew07], see also discussion in Chapter IV). It was an old question ([MM68]) whether they are definable in terms of degree-theoretic properties (in $\mathcal{L}_0$, $\mathcal{L}_1$ or $\mathcal{L}_2$). However this question is still open. We will in Chapter IV, show some nondefinablity results of hyperimmune degrees, and in particular, we confirm a conjecture in [MM68] that there is no quantifier-free $\mathcal{L}_1$-definition of hyperimmune degrees.

Lewis' theorem says every hyperimmune-free non-**FPF** degree has a strong minimal cover ([Lew07]), so in order to build a minimal degree without strong minimal covers (as a negative answer to Yates' question), we have to find a minimal degree which is either **FPF** or hyperimmune. (See §I.2.2 for information about connections between **FPF** and minimality.)

We develop a direct construction of a hyperimmune minimal degree in Chapter V, and the coding idea there turns out to be useful in Chapters VI and VII.

array nonrecursiveness

The notion of array nonrecursive (**ANR**) degrees was introduced in [DJS96] to generalize the notion of being $\overline{\mathbf{GL}_2}$. The **ANR** degrees share a lot of nice properties with $\overline{\mathbf{GL}_2}$ degrees, including the 1-generic bounding property, the cupping property and relative recursive enumerability ([DJS96] and [CSh12]). To define them we will also need the notion of the *modulus function*, which we will discuss in detail in Chapter II.

In Chapter II (joint with Shore) we give a relativized definition of array non-recursiveness and also develop a forcing schema for $\overline{\mathbf{GL}_2}$ and **ANR** degrees. In particular, we show that every **ANR** degree is **RRE**. Continuing this topic, in Chapter III, we show that a degree is **ANR** if and only if every degree above it is **RRE**, therefore **ANR** degrees are definable in $\mathcal{L}_2$. We also mention two interesting applications of this theorem (see Section III.5).

Relating to minimality, there is an old result that all 1-minimal degrees are array recursive, i.e., not **ANR**; in Chapter V, we prove that there is a 2-minimal degree which is **ANR**. This also answers a generalized version of Yates' question. In addition, it shows that the class of degrees which are "low for **ANR**" is a proper subclass of the array recursive degrees.

I.2.5 Tracing

The basic idea of *tracing* is quite easy: a function f might not be recursive, in which case we have no way of recursively telling what $f(n)$ is; however if we have an effective guessing procedure that makes a number of guesses at the value of $f(n)$, and guarantee that one of our guesses is the correct answer, then we call such an effective guessing procedure a *trace* and we call a degree traceable if all or some (partial) functions recursive in it have traces which satisfy some specified property. Depending on what functions we are tracing and how good the traces are, we have a number of different notions.

We start with *r.e. traceable* degrees, a notion introduced in [Ish99] to answer the strong minimal cover problem for r.e. degrees. In Chapter III we extend the result there from the r.e. degrees to the n-**REA** ones, but not following the same route. So in Chapter X we try to follow the original method as in [Ish99] and show that for n-**REA** degrees, r.e. traceability is equivalent to array recursiveness.

We show two other such equivalence results for tracing notions about the n-**REA** degrees. One of them gives a new proof of an old result in [JLSS89]. We hope to see more such connections of tracing with other degree-theoretic or combinatorial properties.

In studying Yates' question, we introduce a notion of *tree traceable* degrees (where the traces are related to trees) and prove several results about this notion (see Chapter XI). In particular, we show that if we want to construct a minimal degree without strong minimal covers (and hence give a negative answer to Yates' question), then we need to find a minimal degree which is not tree traceable and in the construction we must use trees that are "branchy" enough so that some function recursive in the degree is not traced by any trees specified in the definition of tree traceability.

I.3 Basic Definitions and Notations

We introduce here some common notations used in this dissertation. For some notions (such as tree systems) which are defined differently in different chapters, we will make their definitions separately later. This is mainly because that in each theorem one definition might be easier to work with than others. It is also possible that we have special restraints for using letters in each chapter separately, or we have a slightly different notation than what is described here (in which case we will specify it).

I.3.1 Strings

A *string* is a (finite or infinite) sequence of natural numbers. A *binary string* is a string whose entries are either 0 or 1. We use $2^{<\omega}$ (2^ω) to denote the set of all finite (infinite) binary strings, and $\omega^{<\omega}$ (ω^ω) to denote the set of all finite (infinite) strings.

We usually use lower case Greek letters to denote finite or infinite strings. For two finite strings σ and τ, we write $\sigma * \tau$ for the usual concatenation of σ with τ. If i is a natural number, we also write $\sigma * i$ for the usual concatenation of σ with $\langle i \rangle$, as we confuse the number i with the string $\langle i \rangle$.

We use $|\sigma|$ to denote the length of a string σ. We write $\sigma \subset \tau$ if σ is an initial segment of τ (and so τ is an extension of σ), and $\sigma \subsetneq \tau$ if the relation is proper, i.e., $\sigma \subset \tau$ and $\sigma \neq \tau$. We also use the standard interval notation to denote all strings between two strings, e.g., (σ, τ) denotes all strings strictly between σ and τ. We write $\sigma | \tau$ to denote that σ and τ are incompatible.

If α is an infinite string, we also write $\sigma \subset \alpha$ if for every $i < |\sigma|$, $\sigma(i) = \alpha(i)$. We also call σ an initial segment of α, and β an (infinite) extension of σ.

We use upper case letters $A, B, C, ...$ to denote sets of natural numbers. We often view sets of natural numbers as infinite binary strings ($A(x) = 1 \iff x \in A$) and may write $\sigma \subset A$ to denote that σ is an initial segment of A. We may also say that σ is *compatible* with A (or A-compatible) if $\sigma \subset A$.

We use $\sigma \oplus \tau$ to denote the join of two (possibly infinite) strings σ and τ. In the case that their lengths are different (so at least one of them is finite), we define $\sigma \oplus \tau$ to have length $2n$ where n is the length of the shorter one.

I.3.2 Turing functionals

Let φ_e be the e-th oracle Turing functional. Our convention is that the computation of $\varphi_e^\tau(x)$ only run for $|\tau|$ many steps, and so if it converges then it must converge within $|\tau|$ many steps. This makes it recursive to decide whether $\varphi_e^\tau(x)$ converges for any triple (e, τ, x). Another convention is that, when we only care about the totality of functions (such as in minimality constructions), we will assume that if $\varphi_e^\tau(x)$ converges then $\varphi_e^\tau(y)$ also converge for all $y < x$, that is, we try to compute $\varphi_e^\tau(x)$ only when we have finished computing $\varphi_e^\tau(y)$ for all $y < x$.

We say that two strings σ and τ *e-split* if there is an x such that $\varphi_e^\sigma(x) \downarrow \neq \varphi_e^\tau(x) \downarrow$. We denote this relation by $\sigma |_e \tau$, and we also say that σ and τ form an *e-splitting pair* or an *e-splitting*.

I.3.3 Trees

We have some slightly different notions of trees in different chapters, and here we only list some common definitions.

A *tree* is a possibly partial function T from a set of strings to another set of strings satisfying the following properties.

1. (Substring property) $(\sigma \subset \tau \wedge T(\tau)\downarrow) \Rightarrow T(\sigma)\downarrow$.
2. (Order preserving) $(T(\sigma)\downarrow \wedge T(\tau)\downarrow) \Rightarrow (\sigma \subsetneq \tau \Leftrightarrow T(\sigma) \subsetneq T(\tau))$.
3. (Nonorder preserving) $(T(\sigma)\downarrow \wedge T(\tau)\downarrow) \Rightarrow (\sigma|\tau \Leftrightarrow T(\sigma)|T(\tau))$.

In most cases we will choose $2^{<\omega}$ or $\omega^{<\omega}$ as the domain or the codomain.

Note that a tree T naturally induces a (possibly partial) function from infinite strings to infinite strings, sending each infinite α to $\cup_{\sigma\subset\alpha}T(\sigma)$. We abuse notation by using T to denote this function as well.

Given a set A and a tree T, we say A is a *path* on T if there is an infinite string α such that $A = T(\alpha)$. We write $[T]$ to denote the set of all paths on T.

A *node* on T is a string in the range of T. A *leaf* or a *terminal node* is a node that does not have proper extensions on the tree. A string is *even* if it is of even length. An *even-node* is a node whose preimage is even. Note that an "even node" and an "even-node" have different meanings, but in this dissertation we will not use the former notion.

A tree is *finite* if its domain is finite. If the domain is exactly 2^n or ω^n, then we also call the number n the *height* of the tree. The *level* of a node is defined as the length of its preimage. Equivalently the level of a node τ on T is the number of the nodes on T which are proper substrings of τ.

Given two trees T and S, we say that S is a *subtree* of T if every node on S is a node on T. A sequence of trees $\langle T_i\rangle_{i\in\omega}$ is *nested* if each T_{i+1} is a subtree of T_i.

A tree is *e-splitting* if every pair of incompatible nodes on the tree is an e-splitting pair. A tree is (A-)*recursive* if it is a partial recursive function (in A). Here is the usual computation lemma for e-splitting trees in minimal degree constructions. It will be used in our construction for minimal degrees.

Lemma I.3.1. *If T is an e-splitting recursive tree and $A \in [T]$, then $A \leq_T \varphi_e^A$.*

Proof. Standard. See [Ler83, Lemma V.2.6]. □

CHAPTER II
DOMINATION AND FORCING (JOINT WITH SHORE)

This chapter will appear as a paper in the Journal of Symbolic Logic.

II.1 Introduction

We present some abstract theorems showing how domination properties equivalent to being $\overline{\mathbf{GL}_2}$ or array nonrecursive can be used to construct sets generic for different notions of forcing. These theorems are then applied to give simple proofs of some known results. We also give a direct uniform proof of a recent result of Ambos-Spies, Ding, Wang and Yu ([ASDWY09]) that every degree above any in $\overline{\mathbf{GL}_2}$ is recursively enumerable in a 1-generic degree strictly below it. Our major new result is that every array nonrecursive degree is r.e. in some degree strictly below it. Our analysis of array nonrecursiveness and construction of generic sequences below **ANR** degrees also reveal a new level of uniformity in these types of results.

The motivations for the work presented here were twofold. The first was the similarity between certain constructions of degrees below a nonzero recursively enumerable one and the analogous ones for degrees that are $\overline{\mathbf{GL}_2}$ or **ANR** (array nonrecursive). (A Turing degree $\mathbf{a}$ is in $\mathbf{GL}_2$ if $\mathbf{a}''=(\mathbf{a}\vee\mathbf{0}')'$. If $\mathbf{a}\notin\mathbf{GL}_2$ we say $\mathbf{a}\in\overline{\mathbf{GL}_2}$. An equivalent condition is that for every function $g\leq_T \mathbf{a}\vee\mathbf{0}'$, there is an $f\leq_T\mathbf{a}$ which is not dominated by g, i.e. $\exists^\infty x(g(x)<f(x))$ (see [Ler83, IV.3.4]).) A degree $\mathbf{a}$ is *array nonrecursive* (**ANR**) if, for every function $g\leq_{wtt}0'$, there is an $f\leq_T\mathbf{a}$ which is not dominated by g ([DJS96]). In particular, there are theorems (see [Sh81, Lemma 4.2] and [Sh07, Theorem 4.1]) about coding sets which are Σ_3^A below $\mathbf{a}$ for $A\in\mathbf{a}$ when $\mathbf{a}$ is either r.e. and nonrecursive or **ANR**. These results played crucial roles in the proofs, respectively, that the theory of $\mathcal{D}(\leq\mathbf{0}')$, the degrees below $\mathbf{0}'$, is recursively isomorphic to that of true (first order) arithmetic and that the Turing jump operator is directly definable in any jump ideal containing $\mathbf{0}^{(\omega)}$, the degree of the truth set of (first order) arithmetic. Both theorems were proved by fairly complicated but in some ways similar constructions. The first used what is called r.e. permitting and the second **ANR** permitting. Both, like many constructions below r.e., **ANR** or $\overline{\mathbf{GL}_2}$ degrees, depend on domination properties of the given degree to carry out a type of forcing argument (meeting various dense sets) in a type of priority construction.

Thus it seemed that these and other results could be simplified by proving that **ANR** (and so *a fortiori* $\overline{\mathbf{GL}_2}$) degrees are *relatively recursively enumerable*, **RRE**, i.e. recursively enumerable in some degree strictly below them. The point here is that, if this were true, then the need for some of the separate proofs for **ANR** or $\overline{\mathbf{GL}_2}$ degrees could be eliminated by simply citing the corresponding ones for r.e. degrees. Moreover, it seemed desirable to formulate a general theorem about meeting classes of dense sets for specified notions of forcing based on the relevant domination properties characterizing the **ANR** and $\overline{\mathbf{GL}_2}$ degrees that would unify the various constructions exploiting these properties.

The second motivation for this work was the paper of Ambos-Spies, Ding, Wang and Yu ([ASDWY09]). They proved the following:

Theorem II.1.1 (Ambos-Spies, Ding, Wang, Yu [ASDWY09, Theorem 1.5])**.** *Every* $\mathbf{a}\in\overline{\mathbf{GL_2}}$ *is* **RRE** *and, in fact, every* $\mathbf{b}$ *above any* $\mathbf{a}\in\overline{\mathbf{GL_2}}$ *is r.e. in some*

1-*generic degree* $\mathbf{c} < \mathbf{b}$.

In [ASDWY09] the authors then raised a number of interesting questions asking for characterizations of the degrees $\mathbf{a}$ such that every $\mathbf{b} \geq \mathbf{a}$ is $\mathbf{RRE}$ and the relation between being $\mathbf{RRE}$ and (the apparently stronger) property of being r.e. in a strictly smaller 1-generic degree. (A set G is 1-generic if, for every r.e. set of binary strings S, there is a binary string $\sigma \subset G$ such that either $\sigma \in S$ or $(\forall \tau \supseteq \sigma)(\tau \notin S)$. A degree is 1-generic if it contains a 1-generic set.)

We present in §II.2 an analysis of the domination properties characterizing array nonrecursiveness that provide an appropriate definition for a relativized version of the notion. The analysis also proves that $\mathbf{ANR}$ degrees satisfy a stronger domination property with greater uniformity than previously established. This property is closer to that characterizing $\overline{\mathbf{GL}_2}$ and allows us to give a framework for a general theorem about meeting dense sets recursively in either $\overline{\mathbf{GL}_2}$ or $\mathbf{ANR}$ degrees. Even in the $\overline{\mathbf{GL}_2}$ case, our Theorem II.2.9 is more general than the ones in the literature that typically deal only with Cohen forcing. In particular, it allows for notions of forcing that are recursive in the given $\overline{\mathbf{GL}_2}$ or $\mathbf{ANR}$ degree and so are directly applicable to results that, for example, involve statements about coding the given set. It thus applies directly to results about cupping (join) properties and jump inversion and can be used to simplify the proofs of such theorems from [JP78]. It also includes notions of forcing whose conditions are objects such as finite trees which are more complicated than binary strings. The proof we provide is also simpler than the standard ones (even) for oracle constructions as in [JP78] or [Ler83, Section III.5] in that we eliminate the procedure of, given the current string σ (an initial segment of our eventual generic G), appointing a string $\tau \supset \sigma$ as a target (to satisfy some density requirement) and moving toward τ one step at a time while at every step checking to see if some target for a higher priority requirement has been located. While this procedure makes sense for binary strings, it is hard to see what to make of it in more general settings when the forcing conditions are more complicated. Instead, we provide a method that, at every step, attempts to satisfy the highest possible requirement not currently satisfied. While we cannot always immediately satisfy the next requirement as in the simpler Kleene-Post finite extension arguments, these attempts eventually succeed for each requirement. We then present a couple of illustrative applications for known results (at times extended from $\overline{\mathbf{GL}_2}$ to $\mathbf{ANR}$) that are proven by ad hoc arguments in the literature. New results for $\mathbf{ANR}$ degrees including Theorem II.3.2 that $\mathbf{ANR}$ degrees are $\mathbf{RRE}$ are presented in §II.3 . This supplies the result needed to unify those of [Sh81] and [Sh07] as described above.

In §II.4, we provide direct proofs of weaker natural variants, as well as the full result, of Theorem II.1.1 for $\overline{\mathbf{GL}_2}$ degrees. To be more precise, we note that the proof of Theorem II.1.1 ([ASDWY09, Theorem 1.5]) , while very ingenious and clever, is quite indirect and nonuniform. It proceeds by first establishing that any $\mathbf{a} \leq \mathbf{0}'$ which is not $\mathbf{L}_2$ (i.e. $\mathbf{a}'' > \mathbf{0}''$) is $\mathbf{RRE}$. This argument relies on results of [Har98] to convert the problem into one of finding an infinite ascending or descending chain in a linear order constructed from $\mathbf{a}$ and then on (a modification of) one of Hirschfeldt and Shore ([HSh07]) to (nonuniformly) produce such a chain that is low and even 1-generic. [ASDWY09] then use a modification of a result of Jockusch and Posner ([JP78]) for $\overline{\mathbf{GL}_2}$ degrees (proved below for $\mathbf{ANR}$ as Theorem II.3.1) and one of Yu ([Yu06]) as well as the Robinson Jump Interpolation Theorem ([Rob71]) and another result of Jockusch and Posner ([JP78]) to reduce the general case to a relativization of the one for degrees below $\mathbf{0}'$ not in $\mathbf{L}_2$. In contrast, our proof that even **ANR** degrees are **RRE** is uniform in a witness that the given degree is **ANR** as defined and explained in Definition II.2.5, Proposition II.2.6 and Theorem II.3.2. In Theorem II.4.1 we extend a more elaborate coding strategy introduced in Definition II.3.5 that provides a 1-generic $\mathbf{c}$ in which the given $\overline{\mathbf{GL}_2}$ degree is **RRE**. We close our treatment of $\overline{\mathbf{GL}_2}$ degrees by using a notion of forcing in which conditions are finite trees to give a direct proof (Theorem II.4.4) of the full version of Theorem II.1.1 that is uniform relative to the choice of a function witnessing the specific instance of the degree being in $\overline{\mathbf{GL}_2}$ required by the construction.[1]

II.2 Domination and Forcing

In this section we begin by analyzing the definition of array nonrecursiveness. We have in mind two goals. One (motivated in part by Theorem II.3.4 and Question II.2.8) is to develop the correct relativized version. The other is to strengthen the known domination properties for these functions and degrees. The strengthenings will be make the notion seem more similar to the domination characterization of $\overline{\mathbf{GL}_2}$ degrees. They will also provide a stronger general theorem about meeting dense sets to construct generic sequences for a larger class of collections of dense sets than had been previously handled. Indeed, they will provide a common framework for the construction of generic sequences below both **ANR** and $\overline{\mathbf{GL}_2}$ degrees.

Recall that the basic domination theoretic definition of array nonrecursiveness as given originally in [DJS96] is for degrees:

Definition II.2.1. A degree $\mathbf{a}$ is **ANR** if for every function $g \leq_{wtt} 0'$ there is an $f \leq_T \mathbf{a}$ such that f *is not dominated by* g, i.e. $\exists^\infty x(f(x) > g(x))$.

What then should be the correct relativized definition that $\mathbf{a}$ is array nonrecursive relative to $\mathbf{b}$? We should at least have $\mathbf{b} \leq \mathbf{a}$. One might first try also requiring that for some (or perhaps any) $B \in \mathbf{b}$ and any $g \leq_{wtt} B'$ there is an $f \leq_T \mathbf{a}$ not dominated by g. This, however, would not be sufficient to relativize the usual results about **ANR** degrees to the realm above $\mathbf{b}$. (In fact, it is not hard to see that there is a single function recursive in $\mathbf{0}'$ which dominates every g wtt below any set X.) Another possibility might be to include all functions f computable from B' with use bounded by a function recursive in A. This too is insufficient. One needs to allow unbounded access to B. Along these lines a stronger version would be that for any g computable from $B \oplus B'$ such that the use from B' is bounded by a function recursive in B there is an $f \leq_T \mathbf{a}$ not dominated by g. Other variations also seem plausible. A simpler route is provided by an alternate characterization of **ANR** degrees from [DJS96] that depends (in the unrelativized case) on only a single function g, the modulus of $0'$:

Definition II.2.2. We let m be the least modulus function of $0'$, i.e. $m(x)$ is the least $s \geq x$ such that $0'_s \upharpoonright x = 0' \upharpoonright x$ where $0'_s$ is the standard enumeration of $0'$. Note that m is nondecreasing. Similarly we let m_h (or m_A) be the least modulus function for the standard enumeration of h' (A') relative to h (A). (We view sets as represented by their characteristic functions.)

[1] Answering [ASDWY09, Question 4.2], Wang ([Wangxx]) has now shown that every **RRE** degree is r.e. in a 1-generic below it. In contrast to our proofs here, however, his are not uniform.

Remark II.2.3. *The common notation for m, the least modulus function of $0'(= K)$, is m_K, which is inconsistent with our notations here. In all other chapters we will use the common notation to write m_K for m.*

Proposition II.2.4 ([DJS96])**.** *A degree* $\mathbf{a}$ *is* **ANR** *if and only if there is a function $f \leq_T \mathbf{a}$ which is not dominated by m.*

We propose to turn this proposition into a definition which relativizes in an obvious way.

Definition II.2.5. A function f is ANR if it is not dominated by m. It is ANR relative to h if $h \leq_T f$ and f is not dominated by m_h. A degree $\mathbf{a}$ is **ANR** relative to $\mathbf{b}$, $\mathbf{ANR}(\mathbf{b})$, if there are $f \in \mathbf{a}$ and $h \in \mathbf{b}$ such that f is ANR relative to h, $ANR(h)$.

Note that when $\mathbf{b} = \mathbf{0}$ this definition agrees with the standard one for $\mathbf{a}$ being **ANR** by Proposition II.2.4. (One also needs to note that, in general, if there is a $g \leq_T X$ not dominated by h then there is an $f \equiv_T X$ which is also not dominated by h. Simply take $f(n) = 2g(n) + X(n)$.)

We now provide a domination property that characterizes being $\mathbf{ANR}(\mathbf{h})$ but is stronger than the ones previously presented in the literature even in the unrelativized case. It also shows that our (seemingly weak) definition in the relativized case is stronger than all the other possible definitions mentioned after Definition II.2.1. It also makes **ANR** seem much more similar to $\overline{\mathbf{GL_2}}$ than did previous definitions. Recall that $\mathbf{a} \in \overline{\mathbf{GL_2}}$ if for every function $g \leq_T \mathbf{a} \vee \mathbf{0'}$, there is an $f \leq_T \mathbf{a}$ which is not dominated by g. Our proposition similarly says that if f is ANR and $g = \Theta(f \oplus 0')$ with $0'$ use bounded by a function $r \leq_T f$ (not just a recursive function) then there is a $k \leq_T f$ which is not dominated by g. We now state and prove the relativized version by substituting an arbitrary h' for $0'$ and also make the uniformities explicit.

Proposition II.2.6. *If f is $ANR(h)$ and $g = \Theta(f \oplus h')$ with the h' use bounded by a function $r \leq_T f$ then there is a $k \leq_T f$ which is not dominated by g. Moreover k can be found uniformly in f in the sense that there is a recursive function $s(e, i, j)$ such that if $\Theta = \Phi_e$, $r = \Phi_i(f)$ and $h = \Phi_j(f)$ then $\Phi_{s(e,i,j)}(f)$ will serve as the required k.*

Proof. Without loss of generality or uniformity we may assume that f, g and r are increasing. We define the required $k \leq_T f$ as follows: To compute $k(n)$ compute, for each $m > n$ in turn, $\Theta_{fr(m)}(f \oplus (h')_{fr(m)}; n)$ (i.e. compute $fr(m)$ many steps in the standard enumeration of h' from h and then, using this set as the second component of the oracle (and f for the first), compute Θ at n for $fr(m)$ many steps) until the computation converges and then add 1 to get the value of $k(n)$. (This procedure must converge as $\Theta(f \oplus h'; n)$ converges.) Now as m_h does not dominate f, there are infinitely many n such that there is a $j \in [r(n), r(n+1))$ with $f(j) > m_h(j)$. For such n we have $fr(n+1) > f(j) > m_h(j) \geq m_h r(n)$. Thus $(h')_{fr(m)} \upharpoonright r(n) = (h') \upharpoonright r(n)$ for every $m > n$. So the computation of $\Theta(f \oplus h'; n)$ is, step by step, the same as that of $\Theta(f \oplus (h')_{fr(m)}; n)$ for each $m > n$ as all the oracles agree on the actual use of the true computation. So eventually we get an $m > n$ such that $\Theta_{fr(m)}(f \oplus (h')_{fr(m)}; n) \downarrow$ and the output must be $\Theta(f \oplus h'; n)$. Thus for these n, $k(n) = g(n) + 1 > g(n)$ as required. The uniformity of the definition of k from f and the functions of the hypotheses is clear. (Noting that we can uniformly, in f and the reduction of h to f, compute the standard enumeration of h' from h.) $\square$

Corollary II.2.7. *A degree* $\mathbf{a}$ *is* $\mathbf{ANR}(\mathbf{b})$ *if and only if for every* $h \in \mathbf{b}$ *there is a* $k \in \mathbf{a}$ *such that* k *is* $ANR(h)$.

Proof. The only if direction is immediate from Definition II.2.5. The other direction follows easily from above the proposition since given one $h \in \mathbf{b}$ with $f \in \mathbf{a}$ which is $ANR(h)$, the modulus function of any $\hat{h} \equiv_T h$ is given by a function of the type specified in the hypotheses of the proposition and so the function $k \leq_T f$ provided by the proposition is not dominated by $m_{\hat{h}}$ and as noted above we may as well take $k \equiv_T f$. It is then the required $ANR(\hat{h})$ function. $\square$

Iterating the notion of relative array recursiveness also provides some interesting questions. (A degree $\mathbf{a}$ is array recursive relative to $\mathbf{b}$ if it is not $\mathbf{ANR}(\mathbf{b})$.)

Question II.2.8. *If* $\mathbf{0} = \mathbf{a}_0 < \mathbf{a}_1 < \mathbf{a}_2 < \cdots < \mathbf{a}_n$ *is a sequence of degrees such that* $\mathbf{a}_{i+1}$ *is array recursive relative to* $\mathbf{a}_i$ *for each* i*, what can be said about* $\mathbf{a}_n$*? Showing that* $\mathbf{a}_n \notin \mathbf{GH}_1$ *would provide interesting information.*

We want to consider notions of forcing $\mathcal{P}$ in which the underlying set P of conditions (thought of as a subset of $\mathbb{N}$) and the extension relation $\leq_{\mathcal{P}}$ are both recursive in a specified set A. We call such notions of forcing A-recursive (or $\mathbf{a}$-*recursive* when $A \in \mathbf{a}$). Rather than using $\mathcal{C}$-generic filters (for a class $\mathcal{C} = \{D_n | n \in \mathbb{N}\}$) of dense sets) we work with $\mathcal{C}$-generic sequences $\langle p_i \rangle$ of conditions: $\forall i (p_i \leq_{\mathcal{P}} p_{i+1})$ and $\forall n \exists i (p_i \in D_n)$ and a *density function* $d(p, n)$ such that $\forall p \in P \forall n (d(p, n) \leq_{\mathcal{P}} p \ \&\ d(p, n) \in D_n)$. (We actually construct the generic sequences we need using such functions. Going from the sequence to the filter it generates is not always a recursive operation.)

The basic fact about degrees $\mathbf{a} \in \overline{\mathbf{GL_2}}$ being able to construct generic sequences (for Cohen forcing) is [JP78, Lemma 6] of stating that if $\mathcal{C} = \langle D_n \rangle$ is a sequence of dense sets (in $2^{<\omega}$) uniformly recursive in $A \oplus 0'$ for any $A \in \mathbf{a}$ then there is a $\mathcal{C}$-generic sequence recursive in $\mathbf{a}$. (In fact, as [DJS96] pointed out, it is easy to see that this condition also implies (and so is equivalent to) $\mathbf{a} \in \overline{\mathbf{GL_2}}$.) We wish to generalize this result to arbitrary notions of forcing that are $\mathbf{a}$-recursive. We give a construction more direct than the original and usual one in that at each step we move (if at all) directly to the condition that seems to get into the first D_n that our sequence does not yet seem to have met rather searching for a "best possible" target then moving towards it step by step and perhaps changing our mind before reaching it. Also note that the idea of moving toward a condition p step by step that makes natural sense when the conditions are binary sequences does not make any obvious sense when they are arbitrary numbers under an arbitrary order relation.

We also give an argument that works (under the appropriate assumptions and with minor variations) for both $\overline{\mathbf{GL}_2}$ and $\mathbf{ANR}$ degrees. For $\mathbf{a} \in \overline{\mathbf{GL_2}}$ the natural formulation of the necessary condition on the sequence $\langle D_n \rangle$ of dense sets that we want to meet is that it is uniformly recursive in $A \oplus 0'$. What we actually use in the construction is a density function. In this setting, the existence of the desired function d always follows from, and is usually equivalent to, the density of the D_n and their being uniformly recursive in $A \oplus 0'$. This is no longer the case when we move from $\overline{\mathbf{GL}_2}$ to $\mathbf{ANR}$ and so from Turing reducibility to wtt reducibility. (For example, one cannot get the required $d \leq_{wtt} A \oplus 0'$ from the assumption that the D_n are dense and uniformly wtt reducible to $A \oplus 0'$ as its definition requires an unbounded search.) Thus to handle **ANR** degrees we would naturally turn to density functions as is done for Cohen forcing in [DJS96]. To make the proofs in the two cases as similar as possible, we use them for the $\overline{\mathbf{GL}_2}$ case as well. Note that, by Proposition II.2.6,we can actually get by with a weaker hypothesis in the **ANR** case than might be expected that is closer to that for $\overline{\mathbf{GL}_2}$. For notational convenience we state and prove the unrelativized versions of the theorem but, given the definitions and results above, relativization (to $\overline{\mathbf{GL}_2}(\mathbf{b})$ and $\mathbf{ANR}(\mathbf{b})$) is routine.

Theorem II.2.9. *Suppose $\mathcal{P}$ is an A-recursive notion of forcing, $\mathcal{C} = \langle D_n \rangle$ a sequence of sets dense in $\mathcal{P}$ with a density function $d(x, y) = \Psi(A \oplus 0'; x, y)$.*

(i) If $A \in \overline{\mathbf{GL}_2}$ then there is a $\mathcal{C}$-generic sequence recursive in A.

(ii) If $A \in \mathbf{ANR}$ and the use from $0'$ in the computation of $\Psi(A \oplus 0'; x, y)$ is bounded by a function $\hat{r} \leq_T A$, then there is also a $\mathcal{C}$-generic sequence recursive in A.

*In both cases the sequence $\langle p_s \rangle$ constructed is $\mathcal{C}$-generic because $\forall n \exists s(p_{s+1} = d(p_s, n))$. Moreover, in the **ANR** case the generic sequence is uniformly computable from any ANR $f \in \mathbf{a}$ (as a function of the indices of Ψ and of $\hat{r}$ relative to f).*

Proof. Let $\hat{r}(x, y)$ be a function that bounds the $0'$ use in the computation of $\Psi(A \oplus 0'; x, y)$. Without loss of generality we may assume that $\hat{r}(x, y)$ is increasing in both x and y. In case (i) we may clearly take $\hat{r} \leq_T A \oplus 0'$ and in case (ii) we may take $\hat{r} \leq_T A$ by hypothesis. Next note that the nondecreasing function $m\hat{r}(s, s)$ in case (i) is recursive in $A \oplus 0'$ and in case (ii) satisfies the hypotheses of Proposition II.2.6, i.e. it is computable from $A \oplus 0'$ and its $0'$ use is bounded by a function $(\hat{r}(s, s))$ recursive in A. Finally note that the maximum of the running times of $\Psi(A \oplus 0'; x, y)$ for $x, y \leq s$ is also is such a function in each case. (We run Ψ on each input and then output the sum of the number of steps needed to converge.) Finally, we let r be the maximum of these three functions so it too is of the desired form. We now have, by the basic characterization of $\overline{\mathbf{GL}_2}$ degrees or Proposition II.2.6, an increasing function $g \leq_T A$ not dominated by r. We use g to construct the desired generic sequence p_s by recursion.

We begin with $p_1 = \mathbf{1}$. At step $s + 1$ we have (by induction) a nested sequence $\langle p_i | i \leq s \rangle$ with $p_i \leq s$. We calculate $0'_{g(s+1)}$ and see if there are any changes on the use from $0'$ in a computation based on which some D_m was previous declared satisfied. If so, we now declare it unsatisfied. Suppose n is the least $m < s + 1$ such that D_m is not now declared satisfied. (There must be one as we declare at most one m to be satisfied at every stage and none at stage 1.) We compute $\Psi_{g(s+1)}(A \oplus 0'_{g(s+1)}; p_s, n)$. If the computation does not converge or gives an output q such that $q > s + 1$ or $q \not\leq_{\mathcal{P}} p_s$ we end the stage and set $p_{s+1} = p_s$. Otherwise, we end the stage, declare D_n to be satisfied on the basis of this computation of the output q and set $p_{s+1} = q$.

We now try to verify that the sequence constructed is $\mathcal{C}$-generic and indeed $\forall n \exists s(p_{s+1} = d(p_s, n))$. Clearly if we ever declare D_n to be satisfied (and define p_{s+1} accordingly) and it never becomes unsatisfied again then $p_{s+1} = d(p_s, n)$.

Moreover, if we ever declare D_n to be satisfied (and define p_{s+1} accordingly) and it remains satisfied at a point of the construction at which we have enumerated $0'$ correctly up to $r(p_s, n)$, then by definition $p_{s+1} = d(p_s, n)$ and D_n is never declared unsatisfied again. We now show that this happens.

Suppose all D_m for $m < n$ have been declared satisfied by s_0 and are never declared unsatisfied again. Let $s+1 \geq s_0$ be least such that $g(s+1) \geq r(s+1)$. If D_n was declared satisfied at some $t+1 \leq s$ on the basis of some computation of $\Psi_{g(t+1)}(A \oplus 0'_{g(t+1)}; p_t, n)$ and there is no change in $0'$ on the use of this computation by stage $g(s+1)$ then the computation is correct, $p_{t+1} = \Psi(A \oplus 0'; p_t, n) \in D_n$ and D_n is never declared unsatisfied again. (The point here is that by our choice of s, $g(s+1) > m\hat{r}(s+1, s+1) \geq m\hat{r}(p_t, n)$ and so $0'_{g(s)} \restriction \hat{r}(p_t, n) = 0' \restriction \hat{r}(p_t, n)$.) Otherwise, D_n is unsatisfied at s and the least such. By construction we compute $\Psi_{g(s+1)}(A \oplus 0'_{g(s+1)}; p_s, n)$. The definition of r along with our choice of g and s guarantee that this computation converges and is correct and so unless $d(p_s, n) > s+1$ we declare D_n satisfied, set $p_{s+1} = d(p_s, n)$ and D_n is never declared unsatisfied again. If $d(p_s, n) > s+1$, we set $p_{s+1} = p_s$ and, as D_n remains unsatisfied and the computations already found do not change, we continue to do this until we reach a stage $v+1 \geq d(p_s, n)$ at which point $p_v = p_s$ and we set $p_{v+1} = d(p_v, n)$ declare D_n satisfied and it is never unsatisfied again.

The uniformity required in the **ANR** case is immediate from Proposition II.2.6 and our construction. $\square$

The uniformity provided in the **ANR** case of this Theorem carries over to most constructions of degrees recursive in a given **ANR** one. We describe them explicitly in a number of results below. One classic example is the result of DJS that every **ANR** degree bounds a 1-generic degree. Our construction shows that there is a single e such that, for every ANR function f, $\Phi_e(f)$ is 1-generic (see also Proposition II.3.6).

II.3 ANR degrees are RRE

In this section we give a number of applications of the basic Theorem II.2.9 for **ANR** degrees including the result that they are **RRE**. We begin by extending a theorem of [JP78] from $\overline{\mathbf{GL}_2}$ to **ANR**. Even for the $\overline{\mathbf{GL}_2}$ case, it does not fall under the usual paradigm since it makes demands on coding that require a notion of forcing that is **a**-recursive but not recursive.

Theorem II.3.1. *If* $\mathbf{a} \in \mathbf{ANR}$ *and* $\mathbf{c} \geq \mathbf{a} \vee \mathbf{0}'$ *and is r.e. in* $\mathbf{a}$*, then there is a* $\mathbf{g} \leq \mathbf{a}$ *s.t.* $\mathbf{g}' = \mathbf{c}$*.*

Proof. First fix an $A \in \mathbf{a}$ and an A-recursive enumeration $\langle C_s \rangle$ of C. The conditions in our notion of forcing $\mathcal{P}$ are binary strings σ but extension is defined to reflect the given enumeration of C. We let $F(\sigma) = \{e | \Phi_e^\sigma(e) \downarrow\}$. (We employ the usual conventions so that, for example, the computation of $\Phi_e^\tau(x)$ requires at least x many steps to converge and runs only for $|\tau|$ many steps so F is a recursive function and its values are finite sets.)

If $\tau \supseteq \sigma$ (and so $F(\tau) \supseteq F(\sigma)$) and for any $e \leq \min(\{|\sigma|\} \cup (F(\tau) - F(\sigma)))$, and for any $\langle e, s \rangle \in [|\sigma|, |\tau|)$, $\tau(\langle e, s \rangle) = C_{|\sigma|}(e)$ we say that $\tau \leq_{\mathcal{P}} \sigma$. We make this into the required partial order by taking the transitive closure of this relation.

The transitive closure is also recursive in A because we can lay out and check the finitely many possible one step paths between any p and q to see if any of them satisfy the original relation at each step. The intuition (as in the jump theorem in [Shn59]) is that, whenever we try to extend a string, we want to make sure that some (eventually growing) initial segment of columns respects the enumeration of C in sense that $\tau(\langle e,s\rangle) = C_{|\sigma|}(e)$ and so $\lim_{s\to\infty} G(\langle e,s\rangle = C(e)$. This makes $C \leq_T G'$ by the Shoenfield limit lemma.

Define our sequence $\mathcal{C}$ of sets as follows:

$$D_{n,j} = \{\sigma : |\sigma| > j \ \&\ [\Phi_n^\sigma(n)\downarrow \text{ or } \forall\tau \supset \sigma[\Phi_n^\tau(n)\uparrow \text{ or}$$

$$\exists s \exists e < \min(\{|\sigma|\} \cup (F(\tau) - F(\sigma)))(\langle e,s\rangle \in [|\sigma|, |\tau|) \ \&\ \tau(\langle e,s\rangle) \neq C_{|\sigma|}(e))]]\}.$$

We calculate the required density function $d(\sigma, \langle n,j\rangle)$ for the $D_{n,j}$ as follows: Given any σ, n and j we may as well assume (by, recursively in A, taking a long enough extension ρ with $\rho(\langle e,s\rangle) = C_{|\sigma|}(e)$ for every $\langle e,s\rangle > |\sigma|, \langle e,s\rangle \leq j$) that $|\sigma| > j$. Now check whether $\Phi_n^\sigma(n)\downarrow$, if so, set $d(\sigma\, \langle n,j\rangle) = \sigma \in D_{n,j}$ and we are done. If not, use A to get all the values of $C_{|\sigma|}(e)$ for $e \leq |\sigma|$. Ask ($0'$) whether we can find an extension τ of σ with the property that for all $e \leq \min(\{|\sigma|\} \cup (F(\tau) - F(\sigma)))$ and all s such that $\langle e,s\rangle \in [|\sigma|, |\tau|)$, we have $\tau(\langle e,s\rangle) = C_{|\sigma|}(e)$, and $\Phi_n^\tau(n)\downarrow$. If so, we let $d(\sigma, \langle n,j\rangle)$ be the first such τ (found in a standard ordering of computations). It is immediate that $d(\sigma, \langle n,j\rangle) \leq_{\mathcal{P}} \sigma$ and $\tau \in D_{n,j}$. Otherwise, we let $d(\sigma, \langle n,j\rangle) = \sigma \in D_{n,j}$.

As we determined $C_{|\sigma|}$ recursively in A, the $0'$ use for the question asked is clearly bounded by a function recursive in A. Thus, by Theorem II.2.9(ii), we have a $\mathcal{C}$-generic sequence $\langle\sigma_i\rangle$ recursive in A. We let $G = \cup\{\sigma_{p_i}\} \leq_T A$.

First, we claim that $C \leq_T G'$ and, in particular, $C(n) = \lim G(n,s)$ for every n. Given n, there is obviously a j such that for every $e \leq n$, $e \in G' \Leftrightarrow \Phi_e^{\sigma_j}(e)\downarrow$ and $C_{|\sigma_j|}(e) = C(e)$. By the definition of our forcing notion, $G(n,t) = C_{|\sigma_j|}(n) = C(n)$ for $t \geq |\sigma_j|$.

To see that $G' \leq_T C$, assume we have determined $G' \restriction n$ and want to decide if $n \in G'$. Recursively in $C \geq_T A \vee 0'$ find j and k large enough so that $C \restriction n = C_j \restriction n$, $\sigma_{k+1} = d(\sigma_k, \langle n,j\rangle)$ and $G' \restriction n = F(\sigma_{k+1}) \restriction n$. (It is clear, first, that there are such j and k and then that they can be recognized recursively in C which computes both the sequence σ_i and d.) It is now clear from the definition of $D_{n,j}$ and our notion of forcing that $n \in G' \Leftrightarrow \Phi_n^{\sigma_k}(n)\downarrow$. □

We now apply our general theorem to an A-recursive notion of forcing chosen to produce relative recursive enumerability.

Theorem II.3.2. *If* $\mathbf{a} \in \mathbf{ANR}$ *then* $\mathbf{a}$ *is* **RRE**. *Indeed, there are e and i such that, for every ANR function f, $\Phi_e(f) <_T f$ and $W_i^{\Phi_e(f)} \equiv_T f$.*

Proof. Suppose $f \in \mathbf{a}$ is ANR. Let A be the graph of f (and so $A \equiv_T f$). We use an A-recursive notion of forcing $\mathcal{P}$ with conditions $p = \langle p_0, p_1, p_2\rangle$, $p_i \in 2^{<\omega}$ such that

1. $|p_0| = |p_1|$, $p_0(d_n) = A(n-1)$, $p_1(d_n) = 1 - A(n-1)$ where d_n is n^{th} place where p_0, p_1 differ and

2. $(\forall e < |p_0 \oplus p_1|)(e \in p_0 \oplus p_1 \Leftrightarrow \exists x(\langle e, x\rangle \in p_2))$.

Extension in this notion of forcing is defined simply by $q \leq_{\mathcal{P}} p \Leftrightarrow q_i \supseteq p_i$ but note that this applies only to p and q in P. Membership in P and $\leq_{\mathcal{P}}$ are clearly recursive in A.

Our plan is to define a class $\mathcal{C}$ of dense sets D_n with a density function $d(p, n)$ recursive in $A \oplus 0'$ with $0'$ use recursively bounded. Theorem II.2.9(ii) then supplies a $\mathcal{C}$-generic sequence $\langle p_s\rangle \leq_T A$ from which we can define the required $G \leq_T A$ in which **a** is r.e. If $p_s = \langle p_{s,0}, p_{s,1}, p_{s,2}\rangle$ we let $G_i = \cup\{p_{s,i} | s \in \mathbb{N}\}$ for $i = 0, 1, 2$ so $G_i \leq_T A$. Then, if we can force G_0 and G_1 to differ at infinitely many places, $G_0 \oplus G_1 \equiv_T A$. On the other hand, the definition of the notion of forcing obviously makes $G_0 \oplus G_1$ r.e. in G_2. Thus **a** will be r.e. in $\mathbf{g} = \deg(G_2)$. We will have other requirements that make $\mathbf{g} < \mathbf{a}$ as well.

We begin with the dense sets that provide the differences we need:

$$D_{2n} = \{p \in \mathcal{P} : p_0, p_1 \text{ differ at at least } n \text{ points}\}.$$

We define the required function $d(r, 2n)$ by recursion on n. Given r and $n + 1$, we suppose we have calculated $d(r, 2n) = p = \langle p_0, p_1, p_2\rangle \in D_{2n}$ with $p \leq_{\mathcal{P}} r$. If $p \notin D_{2n+2}$, we need to compute a $q = \langle q_0, q_1, q_2\rangle \in D_{2n+2}$ with $q \leq_{\mathcal{P}} p$. Let $q_0 = p_0\hat{\ }A(n)$, $q_1 = p_1\hat{\ }(1 - A(n))$. Choose $i \in \{0, 1\}$ such that $q_i(|p_0|) = 1$. Define $q_2 \supseteq p_2$ by choosing x large and setting $q_2(\langle 2|p_0| + i, x\rangle) = 1$ and $q_2(z) = 0$ for all $z \notin dom(p_2)$ and less than $\langle 2|p_0| + i, x\rangle$. Now $q = \langle q_0, q_1, q_2\rangle$ satisfies the requirements to be a condition in P. It obviously extends p and is in D_{2n+2}. This computation is clearly recursive in A.

We must now add dense sets to guarantee that $A \not\leq_T G_2$. A direct route is to let

$$\begin{aligned} D_{2n+1} = \{p \in \mathcal{P} :\ & \exists x(\Phi_n^{p_2}(x) \downarrow\neq A(x)) \text{ or } \forall(\sigma_0, \sigma_1 \supseteq p_2)[\exists x(\Phi_n^{\sigma_0}(x) \downarrow\neq \Phi_n^{\sigma_1}(x) \downarrow) \\ & \Rightarrow (\exists i \in \{0, 1\})(\exists \langle e, x\rangle)(e < |p_0 \oplus p_1|\ \sigma_i(\langle e, x\rangle) = 1 \neq (p_0 \oplus p_1)(e))]\}. \end{aligned}$$

Of course, the first alternative guarantees that $\Phi_n^{G_2} \neq A$ while the second that $\Phi_n^{G_2}$, if total, is recursive. The point here is that if some p_s in our generic sequence satisfies the second clause then, we can, for any z, calculate $\Phi_n^{G_2}(z)$ by finding any $\sigma \supseteq p_{s,2}$ such that $\Phi_n^{\sigma}(z) \downarrow$ and taking its value as $\Phi_n^{G_2}(z)$. There is such a $\sigma \subseteq G_2$ as $\Phi_n^{G_2}$ is assumed to be total and $G_2 \supseteq p_{s,2}$. If there were some other $\tau \supseteq p_{s,2}$ with $\Phi_n^{\tau}(z) \downarrow\neq \Phi_n^{\sigma}(z) \downarrow$ then, by our choice of s and the definition of D_{2n+1}, there is no $\langle e, x\rangle$ with $e < |p_0 \oplus p_1|$ such that $\tau(\langle e, x\rangle) = 1 \neq (p_0 \oplus p_1)(e)$. Thus we could form a condition $q \leq_{\mathcal{P}} p_s$ with $q_2 = \tau$ by extending p_0 and p_1 by setting $q_1(w) = q_2(w) = 1$ (for $w \geq |p_0|$) if either $\langle 2w, v\rangle$ or $\langle 2w + 1, v\rangle$ is in τ for any v. In this way no new differences between q_0 and q_1 (not already in p_0 and p_1) occur and the definition of being a condition is satisfied. Thus q is a condition extending $p_{s,2}$ with $\Phi_n^{q_2}(z) \downarrow\neq A(z)$ contradicting our choice of s.

We compute the required density function $d(q, 2n+1)$ as follows. Given q we ask one question of $0'$ determined recursively in q: Are there extensions σ_0, σ_1 of q_2 that would show that q does not satisfy the second disjunct in the definition of D_{2n+1}. If not, let $d(q, 2n + 1) = q$ which is already in D_{2n+1}. If so, we find the first such pair (appearing in a recursive search) and ask A which σ_i gives an answer different from $A(x)$. We now need a condition $r = d(q, 2n + 1)$ extending q with third coordinate r_2 extending σ_i. For each $\langle e, x\rangle$ with $e \geq |q_1 \oplus q_2|$) and $\sigma_i(\langle e, x\rangle) = 1$

we define $r_j(z) = 1$ for both $j \in \{0,1\}$ for the z that makes $(r_0 \oplus r_1)(e) = 1$ and otherwise we let $r_j(u) = 0$ for all other u less than the largest element put into either r_0 or r_1 by the previous procedure. We now extend σ_i to the desired r_2 by putting in $\langle k, y\rangle$ for a large y for all those $k \geq |q_1|$ put into $r_0 \oplus r_1$ for which there is no $\langle k, w\rangle$ in σ_i. Otherwise we extend σ_i by 0 up to the largest element put in by this procedure. It is clear that this produces a condition r as required. (No points of difference between r_0 and r_1 are created that were not already present in q.)

We now apply Theorem II.2.9 to get a $\mathcal{C}$-generic sequence $\langle p_s\rangle \leq_T A$. As promised, we let $G_i = \cup\{p_{s,i} | s \in \mathbb{N}\}$ for $i = 0, 1, 2$ and, as described above, $A \equiv_T G_0 \oplus G_1$ which is r.e. in G_2. In addition, the conditions in D_{2n+1} guarantee (as above) that $\Phi_n^{G_2} \neq A$ as well. The uniformity assertions follow immediately from those in Theorem II.2.9 and our construction. □

We now point out some additional information about G_2 that can be extracted from this construction.

Proposition II.3.3. *For the G_2 constructed in the proof of Theorem II.3.2 such that the given $A \in$ **ANR** is r.e. in and strictly above G_2 we also have $G_2' \equiv_T A \oplus 0'$ and so if $A \in \overline{\mathbf{GL}_2}$ then A is also $\overline{\mathbf{GL}_2}(G_2)$, i.e. $(A \oplus G_2')' <_T A''$.*

Proof. We first claim that $G_2' \leq_T A \vee 0'$. To see if $e \in G_2'$, recursively compute an n such that, for every τ, $\Phi_n^\tau(x) = \tau(x)$ if $\Phi_e^\tau(e)\downarrow$ and is divergent otherwise. Now, recursively in $A \oplus 0'$ find an s such that $p_{s+1} = d(p_s, 2n+1)$. If p_{s+1} is in D_{2n+1} because of the first clause of the definition then $\Phi_n^{p_{s+1,2}}(x)\downarrow$ for some x and so $e \in G_2'$. Otherwise we claim $e \notin G_2'$. Suppose for the sake of a contradiction that, for some t, $\Phi_e^{p_{t,2}}(e)\downarrow$. It is now easy to get extensions of $p_{t,2}$ that show that p_{s+1} does not satisfy the second clause of the definition of D_n for the desired contradiction. Simply choose $y > 2|p_{s,0}|, |p_{t,2}|$ and extend $p_{t,2}$ by 0 up to $\langle y, 0\rangle$ and then with $i = 0, 1$ at $\langle y, 0\rangle$ to get the required σ_i. On the other hand, as $\mathbf{a}$ is r.e. in G_2, $A \oplus 0' \leq_T G_2'$ and $G_2' \equiv_T A \oplus 0'$ as desired. If $\mathbf{a} \in \overline{\mathbf{GL}_2}$, $\mathbf{a}'' > (\mathbf{a} \vee \mathbf{0}')'$ and so $\mathbf{a}'' > (\mathbf{a} \vee \mathbf{g_2'})'$ as required. □

A reasonable question now would be to ask for an analogous result for **ANR** degrees to that given in this proposition for $\overline{\mathbf{GL}_2}$ based on our Definition II.2.5 of relative array nonrecursiveness.

Theorem II.3.4. *If $\mathbf{a}$ is* **ANR** *then there is a $\mathbf{g}$ relative to which $\mathbf{a}$ is both r.e. and* **ANR**.

Proof. We replace the sets D_{2n+1} of the proof of Theorem II.3.2 with new ones (also called D_{2n+1}) that force a maximal number of convergences of $\Phi_n^{G_2}(m)$. We here directly specify the (calculation procedure for the) associated density functions $d(p, 2n+1)$: We ask 2^p many questions of $0'$. For each subset F of $\{i | i < p\}$ we ask if there is a $\sigma \supseteq p_2$ "adding no new numbers" (i.e. $\neg\exists\langle e, x\rangle\,(e < |p_0 \oplus p_1|\ \&\ \sigma(\langle e, x\rangle) = 1 \neq (p_0 \oplus p_2)(\langle e, x\rangle))$ that makes $\Phi_n^\sigma(m)\downarrow$ for every $m \in F$. We take a maximal such F and find the first extension $\sigma > p$ witnessing the convergences for $m \in F$. We then get an extension q of p with third coordinate extending σ as before. Note that by the usual coding of binary sequences and triples, $q > p$ as well. By induction then if $g(m)$ is the mth stage s at which we have $p_{s+1} = d(p_s, 2n+1)$ for some n, $p_{g(m)} > m$. Note that this procedure also satisfies the hypotheses of Theorem II.2.9(ii). The D_{2n+1} are declared satisfied and unsatisfied as before but note that they become unsatisfied when we discover that the F associated with the extension used was not maximal (this is, after all, part of the computation on which we based the calculation of d). By the proof of Theorem II.2.9, there are infinitely many m such that the D_{2n+1} declared satisfied at $g(m)$ is never declared unsatisfied. So in particular for such m, for every $e < m$ with $e \in G_2'$, $\Phi_e^{p_{g(m)+1,2}}(e) \downarrow$. Thus if we now define $h(m)$ as the stage in the standard enumeration of G_2' at which all the numbers e such that $\Phi_e^{p_{g(m)+1,2}}(e) \downarrow$ have been enumerated in G_2', then, for each of these infinitely many m, $h(m)$ will be at least as large as the modulus function for G_2' (relative to G_2) at m. $\square$

In the next section we will improve Theorem II.3.2 when $\mathbf{a} \in \overline{\mathbf{GL_2}}$ by making the degree witnessing that $\mathbf{a}$ is **RRE** 1-generic. Here we present another extension to **ANR** of a result from [ASDWY09] about $\overline{\mathbf{GL_2}}$ both as an illustration of our general methodology as well as an introduction of the coding techniques that will be exploited in the next section.

Definition II.3.5. For any set G, we define a relationship $\mathbf{as}(e, G) = \sigma$, the number e is *assigned to (the string)* σ to hold when σ is the shortest $\tau \subset G$ such that $\Phi_e^\tau(e) \downarrow$. If $\mathbf{as}(e, G) = \sigma$, we define the *weak order of* e *along* G, $\mathbf{wo}(e, G)$, recursively as one more than the weak order of e' along G where e' is the unique $x < e$ such, for some $\tau \subset \sigma$, $\mathbf{as}(x, G) = \tau$ and for no $e'' < x$ and ρ with $\tau \subset \rho \subset \sigma$ is $\mathbf{as}(e'', G) = \rho$. If there is no such x, then $\mathbf{wo}(e, G) = 0$. We define $\mathbf{as}(e, \rho) = \sigma$ and the weak order of e along ρ similarly for strings ρ.

Intuitively, we first assign numbers to strings along G, and then for each e assigned to σ, we search downward from σ for the first $e' < e$ assigned. The weak order of e is then the weak order of e' plus 1. We use these notions to code $\mathbf{a}$ into $\mathbf{g}'$ in Proposition II.3.6.

Without loss of generality we may clearly choose our master list of computations so that for every τ there is at most one e such that $\Phi_e^\tau(e) \downarrow$ but $\Phi_e^{\tau^-}(e) \uparrow$. So along any G each node is assigned at most one number.

For every string σ, we let σ^- be the initial segment of σ gotten by removing the last number in σ. If there is a set G (string τ) that is clear from the context such that $\sigma \subset G$ (τ), σ^+ will denote $\sigma\hat{\ }G(|\sigma|)$ ($\sigma\hat{\ }\tau(|\sigma|)$).

Proposition II.3.6. *If* $\mathbf{a} \in \mathbf{ANR}$ *then there is an* 1*-generic degree* $\mathbf{g}$ *recursive in* $\mathbf{a}$ *such that* $\mathbf{g}' = \mathbf{a} \vee \mathbf{0}'$. *Indeed, there is an* e *such that, if* f *is ANR, then* Φ_e^f *is 1-generic and* $(\Phi_e(f))' \equiv_T f \oplus 0'$ *(and this equivalence is also given uniformly).*

Proof. We again begin with A being the graph of f. Our forcing conditions are binary strings. Membership of ρ in $\mathcal{P}$ is defined recursively: if $\sigma \subsetneq \rho$ is the longest initial segment of ρ such that $\mathbf{as}(e, \rho) = \sigma$ for some e then we require that $\sigma^+(|\sigma|) = A(\mathbf{wo}(e, \rho))$; moreover, if τ is the longest initial segment of σ such that $\mathbf{as}(e', \rho) = \tau$ for some $e' < e$ then τ^+ is also (recursively required to be) in $\mathcal{P}$. (By default, for the base case, if no number is assigned to any $\sigma \subset \tau$, then $\tau \in \mathcal{P}$.) Thus $\mathcal{P}$ is clearly recursive in A. The order $\sigma \leq_{\mathcal{P}} \tau$ for our notion of forcing is the usual extension relation on strings, $\sigma \supseteq \tau$.

Our dense sets will be:

$$D_n = \{\sigma \in \mathcal{P} | \Phi_n^\sigma(n) \downarrow \text{ or } (\forall \tau \supset \sigma)(\Phi_n^{\tau^-}(n) \downarrow \text{ or } \Phi_n^\tau(n) \uparrow \text{ or } (\forall i)(\tau\hat{\ }i \notin \mathcal{P}))\}.$$

Now we define an appropriate density function $d(p,n)$. Consider any $\rho \in \mathcal{P}$. (If $\rho \notin \mathcal{P}$, we let $d(\rho,n) = 1$ for any n.) In order to find $\sigma \leq_{\mathcal{P}} \rho$ in D_n, we first check whether $\Phi_n^{\rho}(n)\downarrow$, if so we are done, if not then ask whether there is $\sigma \supset \rho$ such that $\Phi_n^{\sigma}(n)\downarrow$, $\Phi_n^{\sigma^-}(n)\uparrow$ and $\sigma\hat{\ }A(\mathbf{wo}(n,\sigma)) \in \mathcal{P}$. If so, $\sigma\hat{\ }A(\mathbf{wo}(n,q)) \in D_n$ and $\sigma\hat{\ }A(\mathbf{wo}(n,q) \leq_{\mathcal{P}} \rho$. If not, $\rho \in D_n$. The second question only requires $A \upharpoonright n$ to determine the question it asks of $0'$. The rest of the procedure is recursive in A and so d satisfies the hypotheses of Theorem II.2.9(ii).

Theorem II.2.9(ii) now provides a sequence $\langle\sigma_i\rangle \leq_T A$ meeting all the D_n. Our desired G is $\cup\sigma_i$. So $G \leq_T A$. To see that G is 1-generic, we need to check that for every e there is a node $\sigma \subset G$ which forces $e \in G'$ or $e \notin G'$. Note that there is an i such that σ_i decides if $e' \in G'$ for every $e' < e$ (by induction) and $\sigma_{i+1} \in D_e$ (by the genericity of the sequence and the closure of the D_n under extension in $\mathcal{P}$). Of course, if $\Phi_e^{\sigma_{i+1}}(e)\downarrow$ we are done. Otherwise, we claim that σ_{i+1} forces $e \notin G'$. If not, then we can find the first $\tau \supset \sigma_i$ such that $\Phi_e^{\tau}(e)\downarrow$, take a one-bit extension of τ by coding in $A(\mathbf{wo}(e,\tau))$, then this $\tau^+ \in \mathcal{P}$ because all $e' < e$ in G' are forced in by strings shorter than σ_i. This contradicts the fact that $\sigma_{i+1} \in D_n$.

Now we have $G' \leq_T G \vee 0' \leq_T A \vee 0'$ and it remains to show that $A \leq_T G'$. We say some pair (e,τ) has true weak order $n = \mathbf{wo}(e,\tau)$, if no string extending τ can have an assignment e' with weak order n, i.e., any e' assigned after τ is greater than e. For such a pair (e,τ), we must have $\tau^+(|\tau|) = A(\mathbf{wo}(e,\tau)) = A(n)$. Finally, for every n we can use G' to find the unique pair (e,τ) with true weak order n, so $A \leq_T G'$.

As usual the desired uniformities are immediate from those of Theorem II.2.9(ii) and our construction. ☐

II.4 $\overline{\mathbf{GL_2}}$ degrees

Theorem II.3.2 provides a procedure that, given any **ANR** degree $\mathbf{a}$, produces a $\mathbf{g} < \mathbf{a}$ in which $\mathbf{a}$ is r.e. We now show how to make $\mathbf{g}$ 1-generic if $\mathbf{a} \in \overline{\mathbf{GL_2}}$ much more directly than is done in [ASDWY09].

Theorem II.4.1 ([ASDWY09])**.** *If $\mathbf{a} \in \overline{\mathbf{GL}_2}$, then it is r.e. in and above a 1-generic degree.*

Proof. Given a finite string σ, we define the *rank* of σ ($\mathbf{rk}(\sigma)$) as the maximum weak order along σ.

Define a notion of forcing $\mathcal{P}$ consisting of all $p = \langle p_0, p_1, p_2\rangle$, $p_0, p_1, p_2 \in 2^{<\omega}$ such that:

1. p_0 and p_1 are of the same length and if m is the nth position at which they differ, $p_0(m) = 1 - p_1(m) = A(n-1)$.
2. For $n < |p_0 \oplus p_1|$, $n \in p_0 \oplus p_1$ if and only if there exists $\sigma \subsetneq p_2$ and an e assigned to σ with weak order n and $p_2(|\sigma|) = 1$ (i.e. $\sigma^+(|\sigma|) = 1$).
3. $2|p_0| = \mathbf{rk}(p_2)$.

Let $q \leq_{\mathcal{P}} p$ if $q_i \supset p_i$ for each i. Clearly this notion of forcing is A-recursive. Intuitively, if we get $\langle A_0, A_1, G\rangle$ from a sufficiently generic sequence (recursive in

A), then $A \equiv_T A_0 \oplus A_1$ which is r.e. in (and above) G. To determine whether $n \in A_0 \oplus A_1$, one simply runs through G checking whether there is a node with weak order n and a 1 coded right after that. We also want to guarantee the 1-genericity of G.

Given $p \in \mathcal{P}$, we say $\tau \supset p_2$ *respects* p if there is no (n, σ) such that $n < |p_0 \oplus p_1|$, $n \notin p_0 \oplus p_1$, σ is between p_2 and τ with some number e assigned to it and of weak order n and $\tau(|\sigma|) = 1$. So if τ does not respect p, then we cannot extend p to a condition q with q_2 extending τ because any such extension would violate clause (2) in the definition of our notion of forcing. Conversely, the following holds:

Lemma II.4.2. *If $p \in \mathcal{P}$, $\mathbf{rk}(p_2) > n$, $\Phi_n^{p_2}(n) \uparrow$, $\tau \supset p_2$ respects p and n is assigned to τ, then there exists a $q \leq_{\mathcal{P}} p$ such that $q_2 \supset \tau$ and so one can be found recursively in A.*

Proof. We first try to extend p_0 and p_1. If we use τ and blindly follow the dictates of clause (2) of our definition of $\mathcal{P}$ to extend p_0, p_1, we might violate clause (1). However, it is easy to see that we can fix this by extending τ by putting 1's after some nodes with certain weak orders between $\mathbf{rk}(p_2)$ and $\mathbf{rk}(\tau)$. Notice that $\mathbf{wo}(n, \tau) \leq n$ and $\mathbf{rk}(p_2) > n$, and those weak orders that need adjustments are $> n$, so we can extend τ and wait for those weak orders to appear, then code in 1's after some of them so that we meet the requirements of clause (1). $\square$

Define dense sets:

$$D_n = \{p : \Phi_n^{p_2}(n) \downarrow \text{ or } \forall \tau \supset p_2(\Phi_n^{\tau}(n) \uparrow)\}.$$

These D_n will guarantee that $G = \cup_i \{p_{i,2}\}$ is 1-generic. We need to define a density function recursively in $A \oplus 0'$.

Given p, assume $\Phi_n^{p_2}(n) \uparrow$ and $\mathbf{rk}(p_2) > n$ (as we can always extend a string respectfully). We first ask whether there is a τ extending p_2 which makes $\Phi_n^{\tau}(n) \downarrow$ and respects p. If so we find the first such τ and notice that n is assigned to τ. Then, by Lemma II.4.2, we can find $q \in D_n$ extending p. If not, we claim that we can find $q \leq_{\mathcal{P}} p$ with $q \in D_n$ and $\Phi_n^{q_2}(n) \uparrow$, i.e. q forces $n \notin G'$.

Next we ask ($0'$) whether there is a τ extending p_2 which makes $\Phi_n^{\tau}(n) \downarrow$. If not then p is already in D_n. If so, find the first one τ_1, then, by the negative answer to our first question, we know that τ_1 does not respect p. Find the first initial segment η_1 of τ which does not respect p, i.e., η_1^- is assigned a number with weak order $< \mathbf{rk}(p_2)$, $\eta_1(|\eta_1| - 1) = 1$ but the definition of η_1 respecting τ would require it to be 0. Then put $\xi_1 = \eta_1^- * 0$, that is, we change the last bit to respect p.

Note that $\Phi_n^{\xi_1}(n) \uparrow$ because ξ_1 respects p. Now ask ($0'$) whether ξ_1 forces $n \notin G'$. If so, then we are done by (a slight variation of) Lemma II.4.2. If not, we repeat this process: find $\tau_2 \supset \xi_1$ which makes $\Phi_n^{\tau_2}(n) \downarrow$, then find the first initial segment of τ_2 which does not respect p, change the last bit and get a ξ_2 which respects p.

We can continue this process but at each repetition we need some new extension assigned a number with weak order $< \mathbf{rk}(p_2)$. Therefore this process cannot continue forever, i.e. we will eventually stop and get a ξ_i which respects p and which forces $n \notin G'$. Finally use Lemma II.4.2 to get $q \leq_{\mathcal{P}} p$, $q_2 \supset \xi_i$ and $q \in D_n$.

To make sure that A_0 and A_1 have infinitely many points of difference we add another sequence of dense sets:

$$D_n^* = \{p : p_0, p_1 \text{ differ at at least } n \text{ positions}\}.$$

This is similar to the last part of Lemma II.4.2: for any p, one has to be careful extending p_2 while still satisfying clauses (1) and (2) and yet adding a point of difference. Since we don't have any other requirements, this process is easy and recursive in A. $\square$

Our final step is to prove that if $\mathbf{a} \in \overline{\mathbf{GL}_2}$ then every $\mathbf{b} \geq \mathbf{a}$ is r.e. in a 1-generic strictly below it. We prove a seemingly quite different proposition from which this result will easily follow.

Proposition II.4.3. *Every $\mathbf{a} \in \overline{\mathbf{GL}_2}$ computes an infinite binary tree T in $2^{<\omega}$ such that for every path $C \in [T]$, C is 1-generic, C does not compute $\mathbf{a}$ and $\mathbf{a}$ is r.e. in C (but not necessarily above C).*

Proof. We now also arrange our master list of computations so that there is no τ of even length such that $\Phi_e^{\tau}(e)\downarrow$ but $\Phi_e^{\tau^-}(e)\uparrow$. Thus no string of even length is assigned a number.

We say a string τ is A-admissible if there exist p_0, p_1 s.t. $\langle p_0, p_1, \tau\rangle$ is a forcing condition in the sense of the previous construction. Note that p_0 and p_1 are uniquely determined by τ. We will denote them by $p_0(\tau)$ and $p_1(\tau)$, respectively.

We say τ respects σ if τ respects $\langle p_0(\sigma), p_1(\sigma), \sigma\rangle$ as in the previous construction.

Now we define a new notion of forcing: $\mathcal{P}$ consists of finite binary trees such that every leaf is A-admissible. We let $q \leq_{\mathcal{P}} p$ if q is a binary tree extending p, and each leaf of q respects the corresponding leaf in p that it extends.

We define dense sets:

$$D_n = \{p: \text{ for every leaf } \sigma \text{ of p, } \Phi_n^{\sigma}(n)\downarrow \text{ or } \forall\tau \supset \sigma(\Phi_n^{\tau}(n)\uparrow)\}.$$

$$D_n^* = \{p: \text{ for every leaf } \sigma \text{ of p, } p_0(\sigma) \text{ and } p_1(\sigma) \text{ differ at at least } n \text{ positions}\}.$$

These two types of dense sets are handled in the same way as in the previous construction. For every leaf, after we find an extension satisfying the conditions in D_n (or D_n^*), we can assume that it has even length and extend it by 0 and 1 to split it into two leaves. This preserves A-admissibility since no number is assigned to nodes of even length.

Next, we want to make sure that no path C can compute A. Define additional dense sets as follows:

$$E_n = \{p: \text{ for every leaf } \sigma \text{ of } p, [\exists x \Phi_n^{\sigma}(x)\downarrow \neq A(x)$$
$$\text{or } \exists x\forall\tau \supset \sigma(\tau \text{ respects } \sigma \Rightarrow \Phi_n^{\tau}(x)\uparrow)]\}.$$

Now given σ, a leaf of p, we first fix a recursive list of strictly increasing indices $n_0 < n_1 < \cdots < n_i < \ldots$ such that if $\Phi_n^{\tau}(i)\downarrow$ then for any $\tau' \supset \tau$ which is large enough to allow for the spacing required by the conditions we imposed on our master list of computations, $\Phi_{n_i}^{\tau'}(n_i)\downarrow$, and conversely $\Phi_{n_i}^{\tau'}(n_i)\uparrow$ if no $\tau \subset \tau'$ makes $\Phi_n^{\tau}(i)\downarrow$.

Let $\sigma_i = \sigma * 0^j$ where j is the least such that $\mathbf{rk}(\sigma * 0^j) > n_i$ and $\mathbf{rk}(\sigma * 0^j)$ is even. Using $0'$ and A we go through the σ_i asking whether:

$$\forall\tau \supset \sigma_i(\tau \text{ respects } \sigma_i \Rightarrow \Phi_n^{\tau}(i)\uparrow).$$

If we ever get a“yes” answer for some σ_i, we output this σ_i (note that σ_i is always A-admissible and respects σ). If we get a “no” answer for σ_i, we then find the first such $\tau_i \supset \sigma_i$ which respects σ_i and which makes $\Phi_n^{\tau_i}(i)\downarrow$. If $\Phi_n^{\tau_i}(i) = A(i)$we proceed to $i+1$. If $\Phi_n^{\tau_i}(i) \neq A(i)$, then extend τ_i to $\eta_i = \tau_i * 0^k$ for the first k such that η_i is assigned the number n_i. Now this η_i respects σ_i and by Lemma II.4.2 we can find an extension η of η_i which is A-admissible, and then output η.

Now we prove that we always halt in this process: Suppose not, then for any σ_i we would always get a “yes” answer and could find the first $\tau_i \supset \sigma_i$ which respects σ_i and $\Phi_n^{\tau_i}(i) = A(i)$. That would make A recursive.

Finally we get extensions of all leaves of p and then branch each them into two in the same way as in our analysis of D_n and D_n^*. Now E_n forces that, for each path C, either Φ_n^C is not total, or it is not A. □

Theorem II.4.4. *If $\mathbf{a} \in \overline{\mathbf{GL}_2}$ and $\mathbf{b} \geq \mathbf{a}$, then $\mathbf{b}$ is r.e. in and strictly above a 1-generic $\mathbf{c}$. Moreover, a $C \in \mathbf{c}$ can be found uniformly effectively in any $B \in \mathbf{b}$ from an index for an $A \in \mathbf{a}$ as a set recursive in B and an index for a function (recursive in A and hence B) not dominated by a particular effectively determined function recursive $A \oplus 0'$.*

Proof. Let T be the tree recursive in A constructed in the above proposition. Given $B \geq_T A$, we let C be the path in the tree gotten by following B, i.e., $C = T(B)$. It is easy to see that $B \equiv_T A \oplus C$, so B is r.e. in and above C which is, of course, 1-generic. Moreover, since $A \not\leq_T C$, C is strictly below B.

As for the uniformity assertions, we explain what we mean by describing the procedure. We are given B and an index computing A from B. From this information we can effectively find indices (from $A \oplus 0'$) for the density functions for the sets D_n, D_n^* and E_n and then for the associated function r (from $A \oplus 0'$) used in the proof of Theorem II.2.9. The noneffective step is now to produce an index for the function $g \leq_T A$ which is not dominated by r. Given that index for g, the rest of the construction in the proof of Theorem II.2.9 proceeds effectively in A and provides the generic sequence $\langle p_i \rangle$ for our construction here and an index for it from A. Going from the sequence to the corresponding tree T and then to the path $C = T(B)$ is then also uniformly effective in B. □

CHAPTER III
ARRAY NONRECURSIVENESS AND RELATIVE RECURSIVE ENUMERABILITY

This chapter is mainly based on a paper which will appear in the Journal of Symbolic Logic.

III.1 Introduction

In the previous chapter we show that every **ANR** degree is **RRE**. In this chapter we prove that if a degree **a** is array recursive (**AR**) then there is a degree $\mathbf{b} \geq \mathbf{a}$ which is not **RRE**. **ANR** degrees are closed upwards by definition, therefore a degree is **ANR** if and only if every degree above it is **RRE**.

This gives a nice definition of the **ANR** degrees in $\mathcal{L}_2$, i.e., the language of partial order, jump and r.e. relation. Using this result we can directly answer [ASDWY09, Question 4.4] and also [ASDWY09, Question 4.3] (see last section in this chapter). In addition, we show two other applications: one is an interesting corollary that every n-**REA** degree has a strong minimal cover if and only if it is array recursive; the other is to show the existence of a **PA** degree without the join property (we thank Adam Day for pointing this out).

In this chapter we will only use trees that are (possibly partial) functions from $2^{<\omega}$ to $2^{<\omega}$ (i.e., binary trees).

III.2 A Framework for Subtree Constructions

As mentioned in the introduction, we are about to prove:

Theorem III.2.1. *For every array recursive degree* **a***, there is a degree* $\mathbf{b} \geq \mathbf{a}$ *which is not* **RRE**.

Suppose we are given a set A of degree **a**. Our aim is to find a set B which computes A and for every C with $C \equiv_T B$ and C is r.e. in a set $D \leq_T B$, then $B \leq_T D$. Equivalently, B is constructed to satisfy the following list of requirements, for all i, j, k, l:

- $R_{i,j,k,l}$: $\neg[\varphi_j^{\varphi_i^B} = B \ \wedge\ \varphi_i^B = W_l^{\varphi_k^B} \ \wedge B \not\leq_T \varphi_k^B]$.

This basically says that we cannot find $C = \varphi_i^B$ which has the same Turing degree as B via a reduction with index j such that C is enumerated by index l with oracle $D = \varphi_k^B$ and D does not compute B. For convenience we can assume that all strings we handle here are binary.

In our construction, we will approximate B in 2^ω by total A-recursive trees $T : 2^{<\omega} \to 2^{<\omega}$.

For every total A-recursive tree T and every $\sigma \in 2^{<\omega}$, the tree T^* defined by $T^*(\tau) = T(\sigma * \tau)$ is also total and A-recursive. This is the usual Full Subtree above a node and we will denote it by $\mathbf{FSub}(T, \sigma)$.

A tree T *forces* a sentence $P(B)$, if $P(B)$ is true for every $B \in [T]$. Given a tree T, a node σ *forces* a sentence $P(B)$ (on T), if $\mathbf{FSub}(T, \sigma)$ forces $P(B)$.

Suppose we are given a total A-recursive tree T and a quadruple $\langle i,j,k,l\rangle$. We plan to find a total A-recursive subtree T' of T such that T' forces $R_{i,j,k,l}$.

First we ask whether the following is true:

$$\exists\sigma\exists x\forall\tau\supset\sigma\ (\varphi_i^{T(\tau)}(x)\uparrow).$$

That is, we ask whether there is a σ which forces φ_i^B to be partial, i.e., $\varphi_i^B(x)$ diverges for every $B\in[\mathbf{FSub}(T,\sigma)]$. If so, we set $T'=\mathbf{FSub}(T,\sigma)$ and hence satisfy $R_{i,j,k,l}$. Otherwise, we next find an A-recursive subtree T^* of T which forces totality.

We define $T^*(\emptyset)=T(\mu)$ where μ is the first node in a recursive search such that $\varphi_i^{T(\mu)}(0)$ converges. By induction, we assume that $T^*(\sigma)=T(\tau)$ has been defined and now we want to define $T^*(\sigma*0)$ and $T^*(\sigma*1)$. For $n=0,1$, we search above $T(\tau*n)$ on the tree T and find the first $T(\tau')$ where $\varphi_i^{T(\tau')}(|\sigma|+1)$ converges, and let $T^*(\sigma*n)$ be this $T(\tau')$. We can always find such $T(\tau')$ because we have a negative answer to the question above. We denote T^* by $\mathbf{Tot}(T,i)$, the i-Total Subtree of T, and replace T with $\mathbf{Tot}(T,i)$ for convenience of notation. This is a classical totality forcing subtree, and it is more important here to point out a nice fact which will be used later (here $\varphi_i^{T(\sigma)}$ is considered as a string):

Fact III.2.2. *For every σ, $|\varphi_i^{T(\sigma)}|\geq|\sigma|+1$.*

Similarly we can assume that φ_k is total for every path on T by replacing T with $\mathbf{Tot}(T,k)$, as otherwise we could find a full subtree to force φ_k to be nontotal. We can also find a recursive function p such that $\varphi_{p(i,j)}^B=\varphi_j^{\varphi_i^B}$, so by replacing T with $\mathbf{Tot}(T,p(i,j))$ we can assume that for every path $B\in[T]$, $\varphi_j^{\varphi_i^B}$ is also total. In addition we have:

Fact III.2.3. *For every σ, $|\varphi_j^{\varphi_i^{T(\sigma)}}|\geq|\sigma|+1$, and $|\varphi_k^{T(\sigma)}|\geq|\sigma|+1$.*

Now we try to satisfy $\varphi_j^{\varphi_i^B}\neq B$ by asking whether the following is true:

$$\exists\sigma\exists x(\varphi_j^{\varphi_i^{T(\sigma)}}(x)\downarrow\neq(T(\sigma))(x)\downarrow).$$

That is, we ask whether there is a σ that forces $\varphi_j^{\varphi_i^B}\neq B$. If so, we set $T'=\mathbf{FSub}(T,\sigma)$ and claim that the requirement has been satisfied. Otherwise we know:

Fact III.2.4. *For every σ, $\varphi_j^{\varphi_i^{T(\sigma)}}\subseteq T(\sigma)$, and so by totality, for every $B\in[T]$, $\varphi_j^{\varphi_i^B}=B$.*

In this case, we try to satisfy $\varphi_i^B\neq W_l^{\varphi_k^B}$, by asking whether the following is true (note that φ_i^B is binary):

$$\begin{aligned}(\dagger):\exists\sigma\exists x\{&(\varphi_i^{T(\sigma)}(x)=0\ \wedge\ W_l^{\varphi_k^{T(\sigma)}}(x)=1)\\ &\text{or }(\varphi_i^{T(\sigma)}(x)=1\ \wedge\ \forall\tau\supset\sigma(W_l^{\varphi_k^{T(\tau)}}(x)=0))\}.\end{aligned}$$

This sentence may be a bit difficult to parse. The predicate on the first line says that $\varphi_i^{T(\sigma)}(x) = 0$ but x has been enumerated into $W_l^{\varphi_k^{T(\sigma)}}$, so σ forces $\varphi_i^B \neq W_l^{\varphi_k^B}$. The predicate on the second line says that $\varphi_i^{T(\sigma)}(x) = 1$, but x will never be enumerated into $W_l^{\varphi_k^{T(\sigma')}}$ for nodes $T(\sigma')$ extending $T(\sigma)$ on T. Thus this σ also forces $\varphi_i^B \neq W_l^{\varphi_k^B}$. In either case, we set $T' = \mathbf{FSub}(T, \sigma)$ and claim that the requirement has been satisfied.

Now suppose the answer is no. We again construct a new subtree T^* of T as follows: In the base case, notice that $\varphi_i^{T(\emptyset)}(0)$ is defined by Fact III.2.2. If it is 0, by the failure of (†), we know that 0 will never be enumerated into $W_l^{\varphi_k^B}$ for $B \in [T]$ and simply define $T^*(\emptyset) = T(\emptyset)$. If it is 1, then also by the failure of (†), we know that there must be some node $T(\sigma)$ on T where 0 has been enumerated into $W_l^{\varphi_k^{T(\sigma)}}$, so we can find the first such node on T and let it be $T^*(\emptyset)$.

By induction, suppose we have defined $T^*(\sigma) = T(\tau)$ and $|\sigma| \leq |\tau|$. This means that $\varphi_i^{T(\tau * n)}(|\sigma| + 1)$ is defined by Fact III.2.2 for $n = 0, 1$. If this value is 0, then let $T^*(\sigma * n) = T(\tau * n)$. If it is 1, then search above $T(\tau * n)$ on T for the first node which enumerates $|\sigma| + 1$ into $W_l^{\varphi_k}$, and let it be $T^*(\sigma * n)$. It is easy to check that T^* is a total A-recursive subtree of T and we denote it by $\mathbf{TEum}(T, i, k, l)$, the True Enumeration Subtree. A similar key fact about this subtree is the following:

Fact III.2.5. *For $T^* = \mathbf{TEum}(T, i, k, l)$ and for every σ, the first $|\sigma|$ many values of $W_l^{\varphi_k^{T^*(\sigma)}}$ are equal to those of $\varphi_i^{T^*(\sigma)}$ and will never change in the enumeration $W_l^{\varphi_k}$ for nodes extending $T^*(\sigma)$ on T^* (i.e., for every $\sigma' \supset \sigma$, $W_l^{\varphi_k^{T^*(\sigma')}} \upharpoonright |\sigma| = W_l^{\varphi_k^{T^*(\sigma)}} \upharpoonright |\sigma|$). Finally for every $B \in [T^*]$, $W_l^{\varphi_k^B} = \varphi_i^B$.*

For convenience of notation we again replace T with T^*. Now the only chance to win is to make φ_k^B compute B for every path $B \in [T]$. We will introduce a complimenting structure in the next section and finish the proof later. Some "diagram-chasing" ideas will be used in the following "tree diagram", which shows the relations between different copies of 2^ω in our construction.

$$
\begin{tikzcd}
& & \mathbf{III}(2^{<\omega}) \arrow[dl, shift left, "\varphi_j"] \\
\mathbf{I}(2^{<\omega}) \arrow[r, "T"] & \mathbf{II}(2^{<\omega}) \arrow[ur, shift left, "\varphi_i"] \arrow[dr, "\varphi_k"'] & \\
& & \mathbf{IV}(2^{<\omega}) \arrow[uu, "W_l"']
\end{tikzcd}
$$

III.3 Tree Systems

In this section we introduce some terminology needed to prove our main theorem. Some of the ideas are motivated by Lewis' proof that every hyperimmune-free degree which is not **FPF** (fixed-point-free) has a strong minimal cover (see [Lew07, Section 6]).

A *base* is an r.e. subset of $2^{<\omega}$. We use the letter Π to denote bases, possibly with superscripts or subscripts. A base Π is a *strong base* if there is a recursive enumeration $\{\Pi_s\}_{s\in\omega}$ of Π such that $|\Pi_{s+1} - \Pi_s| \leq 1$, and at each step of the enumeration, the only string enumerated at that step has no extensions that have already been enumerated at the previous steps. This is equivalent to saying that, at every step, we only enumerate finitely many strings which extend the strings previously enumerated. We define the *level* of a string in a (strong) base as the number of its proper predecessors. Note that in a strong base or a recursive tree, the level function is recursive.

We always view a (strong) base as a partial order, and it will be convenient to call the immediate predecessor of a string its *parent*, and its immediate successors its *children*.

Suppose we have a total binary tree S recursive in A, we can write $S = \Psi(A)$ where Ψ is a recursive functional. For convenience we assume that Ψ has the following properties:

1. For every σ, $\Psi(\sigma)$ is a finite binary tree, and we can recursively find its height.
2. For every $\tau \supset \sigma$, $\Psi(\tau)$ preserves the tree structure of $\Psi(\sigma)$, i.e., $\Psi(\sigma)$ is a subtree of $\Psi(\tau)$, and those nodes on $\Psi(\tau)$ but not on $\Psi(\sigma)$ must extend leaves of $\Psi(\sigma)$.

Definition III.3.1. Given such a tree functional Ψ and an A such that $S = \Psi(A)$ is a total binary tree, we call a pair (Π, Φ) a *Tree System* for (Ψ, A, S) if the following properties hold:

1. Π is a strong base.
2. There are infinitely many initial segments of A that are in Π.
3. For every $\sigma \in \Pi$, $\Phi(\sigma)$ is a subtree of $\Psi(\sigma)$.
4. For every σ, τ both in Π, if τ properly extends σ, then $\Phi(\tau)$ properly extends $\Phi(\sigma)$.

In addition, if the following holds, we call (Π, Φ) a *Splitting Tree System*:

5. For every pair of incomparable strings σ_0, σ_1 at the same level of Π, all leaves of $\Psi(\sigma_0)$ and $\Psi(\sigma_1)$ are pairwise incomparable.

Intuitively a tree system is an r.e. "tree of trees" and from the enumeration we can find a certain path A which computes a subtree S' of a given tree $S \leq_T A$. Note that conditions (2) and (4) above guarantee that $S' = \Phi(A) = \cup_{\sigma\subset A, \sigma\in\Pi}\Phi(\sigma)$ is a total binary subtree of S.

Lewis actually used Splitting Tree Systems in his main construction (see [Lew07, Section 6]). In our construction (Section III.4) we will only use Tree Systems. We won't use the following lemma, but it illuminates some ideas about our main construction.

Lemma III.3.2. *If (Π, Φ) is an Splitting Tree System for (Ψ, A, S), then every infinite path on $S' = \Phi(A)$ computes A.*

Proof. Suppose we have $C \in [S']$, we compute A as follows: Enumerate Π, for every $\sigma \in \Pi$ check whether there is a leaf on $\Phi(\sigma)$ which is an initial segment of C. If so, σ is an initial segment of A, since any string in Π incompatible with A would give us incomparable leaves by Definition III.3.1 (5). The initial segments of A appear in Π infinitely often so we can compute A from C. $\square$

III.4 ANR and RRE

We now finish proving our main theorem.

Proof (of Theorem III.2.1). Recall that we are given a set A of array recursive degree and we want to find B which computes A. This requirement can be easily satisfied as follows: A string σ is *A-pointed* if for every n for which $\sigma(2n)$ is defined, $\sigma(2n) = A(n)$. That is to say, the even positions of σ code the corresponding information about A. If we write $Even(\sigma)$ to denote the even substring of σ, i.e., $Even(\sigma)(n) = \sigma(2n)$, then σ being A-pointed is the same as $Even(\sigma) \subset A$. A tree is *A-pointed* if every node on the tree is A-pointed. Note that every infinite path on an A-pointed tree can compute A.

It is easy to see that we can start our construction with the A-pointed tree T_0 defined by $T_0(\sigma) = A\restriction_{|\sigma|} \oplus \sigma$, and only need to satisfy the requirements $R_{i,j,k,l}$ one by one in our construction.

Now given an A-recursive tree T and a requirement $R_{i,j,k,l}$, we first go through the subtree constructions in Section III.2 and so we can assume that our tree T satisfies Facts III.2.2 to III.2.5. For convenience we summarize those facts as follows:

Fact III.4.1. *For every σ of length n, let $\tau = T(\sigma)$, we know:*

1. $|\tau| \geq n$.
2. $|\varphi_i^\tau| \geq n$.
3. $|\varphi_j^{\varphi_i^\tau}| \geq n$ *and it is an initial segment of* τ.
4. $|\varphi_k^\tau| \geq n$.
5. $W_l^{\varphi_k^\tau} \restriction n = \varphi_i^\tau \restriction n$.
6. $W_l^{\varphi_k^{T(\sigma)}} \restriction n$ *will never change, i.e.,* $W_l^{\varphi_k^{T(\sigma')}} \restriction n = W_l^{\varphi_k^{T(\sigma)}} \restriction n$ *for every* $\sigma' \supset \sigma$.

Now the only way to satisfy our requirement is to find a subtree T' of T and guarantee that, for every path $B \in [T']$, $B \leq_T \varphi_k^B$. Also T' needs to be recursive in A to continue our construction. This is why we consider tree systems as in Definition III.3.1.

To simplify the proof, we first take the splitting subtree T^* of T with respect to k defined as follows: $T^*(\emptyset) = T(\emptyset)$, and, by induction, once we have defined $T^*(\sigma) = T(\tau)$, we can search above τ for k-splitting pairs, i.e., $\tau_0, \tau_1 \supset \tau$ such that $\varphi_k^{T(\tau_0)}$ is incomparable with $\varphi_k^{T(\tau_1)}$. By Fact III.4.1(3,5) we know such a splitting exists, so we can find the length-lexicographically first such pair τ_0, τ_1 and define

$T^*(\sigma * n) = T(\tau_n)$ for $n = 0, 1$. We denote T^* by $\mathbf{Spl}(T, k)$ and replace T with such T^*. Note that Fact III.4.1 is preserved.

Let S be $\varphi_k \circ T$ mapping from $\mathbf{I}$ to $\mathbf{IV}$. As T is a k-splitting tree, S is also a total binary tree. In addition, we can define a bijection between the nodes of T and nodes of S. Any node τ on T maps to φ_k^τ on S, and for every node η on S, we can use A to compute T and search for the τ which maps to η. This bijection is recursive in A.

This is to say, for every path $C \in [S]$, if $A \leq_T C$, then C computes its preimage under φ_k, i.e., our requirement is satisfied. Now, our plan is to find a total binary subtree S' of S, also recursive in A, such that for every $C \in [S']$, $A \leq_T C$. Then the corresponding T', which induces $S' = \varphi_k \circ T'$, is also a total binary tree recursive in A, and the requirement is satisfied for every path on T'.

Let $S = \Psi(A)$. We will find S' by constructing a tree system (Π, Φ) for (Ψ, A, S) and make sure that every path on $S' = \Phi(A)$ computes A.

A primitive version of the idea to be used here is that we want to set up some space between the leaf level and a "pseudo-leaf" level (which will be defined in the following construction) of a finite binary tree so that the binary structure of the tree can be used to code some information.

Define a strong base Π_0 and a level function lev as follows: In the base case, we put $\emptyset$ into Π_0 and define $lev(\emptyset) = 0$ (and assume that $\Psi(\emptyset)$ has height 0). At each step s, we consider the next string σ in the length-lexicographical order (i.e., $\sigma = s$ in the canonical coding). Let σ' be the longest predecessor of σ already in Π_0. Suppose $lev(\sigma') = k$ and $\Psi(\sigma')$ is of height n. Also assume that $\Psi(\sigma)$ is of height m. We check whether the following are true:

1. $m \geq n + [\log_2(k+4)] + 2$.
2. For every node τ on $\Psi(\sigma)$ which is $[\log_2(k+4)] + 1$ levels down from the leaf (i.e., a node of level $m - [\log_2(k+4)] - 1$, which we call a *pseudo-leaf* of $\Psi(\sigma)$), we have:
$$Even(\varphi_j^{W_l^\tau \upharpoonright (m-[\log_2(k+4)]-1)}) \supset \sigma'.$$
We call $m - [\log_2(k+4)] - 1$ the *pseudo-height* of $\Psi(\sigma)$.
3. For every pseudo-leaf τ of $\Psi(\sigma)$ extending a pseudo-leaf τ' of $\Psi(\sigma')$, W_l^τ doesn't add new elements to $W_l^{\tau'} \upharpoonright n'$, where n' is the pseudo-height of $\Psi(\sigma')$.

If so, we enumerate σ into Π_0 and define $lev(\sigma) = k + 1$; otherwise we don't do anything and continue to the next step.

Using Fact III.4.1(3,5,6), it is easy to prove by induction that there are infinitely many substrings of A that are in Π_0. Let $\alpha_0, \alpha_1, \ldots, \alpha_n, \ldots$ be all of them in ascending order (so $\alpha_0 = \emptyset$). Define $f(n) \leq_T A$ to be the step s at which α_n enters Π_0 in this enumeration.

By the definition of array recursive degrees, we know that m_K dominates f (see Remark II.2.3). By changing finitely many columns of the standard limit computation of m_K, we can assume that there is a recursive approximation $\lambda(n, s)$ whose limit $g(n) = \lim_{s\to\infty} \lambda(n, s)$ exists and is greater than $f(n)$ for each n. In addition, the number of changes along each column $\{\lambda(n, s)\}_{s\in\omega}$ is bounded by n.

Now we define a strong base $\Pi \subset \Pi_0$ and a function grp (group function) for strings in Π. First put $\emptyset$ into Π and define $grp(\emptyset) = 0$. At step s, there are finitely many strings in $\Pi(s)$: pick anyone of them, say τ with $lev(\tau) = k$, and check whether there are extensions τ' of τ enumerated into Π_0 between steps $\lambda(k+1, s)$ and $\lambda(k+1, s+1)$ with $lev(\tau') = k+1$ (i.e., they are immediate successors of τ in Π_0). If so, add those nodes into $\Pi(s+1)$ and define $grp(\tau')$ to be the number of changes in $\{\lambda(k+1, t)\}_{t\in\omega}$ up to $t = s+1$, i.e. $|\{r \leq s : \lambda(k+1, r) \neq \lambda(k+1, r+1)\}|$. Then do the same thing for every $\tau \in \Pi(s)$. It is easy to prove by induction that Π contains infinitely many initial segments of A.

For every σ in Π, its immediate successors are "grouped" into at most $lev(\sigma)+2$ many groups. The space we have from the pseudo-leaves to leaves is used to code the group numbers in the construction below.

For $k \geq 1$ and $i \in \{0, 1, \ldots, k\}$, define $\theta_{k,i}$ to be the i-th (in lexicographical order) binary string of length $[\log_2(k+1)] + 1$. For example, $\theta_{2,0} = 00$, $\theta_{2,1} = 01$ and $\theta_{2,2} = 10$.

Given a $\sigma \in \Pi$ at level k ($k \geq 1$), let τ be a pseudo-leaf of $\Psi(\sigma)$, we define $Code(\tau, \sigma, i, k+2)$ to be the leaf of $\Psi(\sigma)$ extending τ with $\theta_{k+2,i}$ coded along the tree: Find ρ such that $\tau = [\Psi(\sigma)](\rho)$, then put $Code(\tau, \sigma, i, k+2) = [\Psi(\sigma)](\rho * \theta_{k+2,i})$. Note that there are exactly $[\log_2(k+3)] + 1$ levels from the pseudo-leaves to leaves in the tree $\Psi(\sigma)$, so $Code(\tau, \sigma, i, k+2)$ is a leaf.

For convenience we assume that all nodes in Π are compatible with α_2, the third node along A which is in Π. We define a functional Φ such that for every $\sigma \in \Pi$, $\Phi(\sigma)$ is a subtree of $\Psi(\sigma')$ where σ' is the parent of σ in Π, and every leaf of $\Phi(\sigma)$ is a pseudo-leaf of $\Psi(\sigma')$.

In the base case, we pick one pseudo-leaf τ of $\Psi(\alpha_1)$ and define $\Phi(\alpha_2)$ to be the singleton tree with only one node τ.

Suppose we have defined, for σ of level $k+1$, $\Phi(\sigma)$ to be a subtree of $\Psi(\sigma')$ where σ' is the parent of σ. Let $\sigma_0, \sigma_1, \ldots, \sigma_r$ list all the immediate successors (children) of σ in group i, i.e., $grp(\sigma_0) = grp(\sigma_1) = \cdots = grp(\sigma_r) = i$. For every τ, a leaf of $\Phi(\sigma)$ and pseudo-leaf of $\Psi(\sigma')$, we let $\tau^* = Code(\tau, \sigma', i, k+2)$, the leaf on $\Psi(\sigma')$ which extends τ after coding in $\theta_{k+2,i}$. We then find two incomparable extensions τ_0, τ_1 of τ^* at the pseudo-leaf level of $\Psi(\sigma)$ (note that in the construction of Π_0 we have at least one level between the leaves of $\Psi(\sigma')$ and the pseudo-leaves of $\Psi(\sigma)$). After we have done this for every leaf τ of $\Phi(\sigma)$, we will have a binary subtree R of $\Psi(\sigma)$ and the leaves of R are pseudo-leaves of $\Psi(\sigma)$. We then define $\Phi(\sigma_0) = \Phi(\sigma_1) = \cdots = \Phi(\sigma_r) = R$ (intuitively, we code the group number of σ_0 in every path of $\Phi(\sigma_0)$ with respect to $\Psi(\sigma')$). This finishes the construction and it is easy to see that we now have a tree system (Π, Φ) for (Ψ, A, S).

It suffices to show that, for every infinite path C on $S' = \Phi(A)$, C computes A. We prove this by showing that any such C can compute the sequence $(\alpha_0, \alpha_1, \ldots, \alpha_n, \ldots)$. We can assume that we are given $\alpha_0, \alpha_1, \alpha_2$ and $grp(\alpha_3)$; and at the inductive step we assume that we have α_k and $grp(\alpha_{k+1})$, and we want to find α_{k+1} and $grp(\alpha_{k+2})$.

First we can recursively list all the children $\sigma_0, \sigma_1, \ldots, \sigma_r$ of α_k in group $grp(\alpha_{k+1})$. Of course, this list includes α_{k+1}. We now find the only leaf τ of $\Phi(\sigma_0)(= \Phi(\sigma_1) = \cdots = \Phi(\sigma_r))$ which is an initial segment of C and a pseudo-leaf of $\Psi(\alpha_k)$. With C, τ and $\Psi(\alpha_k)$ we can figure out the (coded) group number x of α_{k+2}: Let $\tau = (\Psi(\alpha_k))(\eta)$, then by our construction, $\theta_{k+2,x}$ can be found as the last $[\log_2(k+3)] + 1$ many bits of η. Finally we can, of course, decode $x = grp(\alpha_{k+2})$ from $\theta_{k+2,x}$.

Now we can compute:

$$X_i = \{\tau : \tau \supset \sigma_i \ \& \ lev(\tau) = k+2 \ \& \ grp(\tau) = x\},$$

i.e., the set of all the children of σ_i in group x. Then we need the following lemma:

Lemma III.4.2. *(With the assumptions above) let $\tau_0 \in X_p$, $\tau_1 \in X_q$ and $p \neq q$. If η_0 is a pseudo-leaf of $\Psi(\tau_0)$, η_1 is a pseudo-leaf of $\Psi(\tau_1)$, and both are initial segments of C, then $\eta_0 \neq \eta_1$. In addition, if $\eta_0 \subset \eta_1$, then τ_0 is not an initial segment of A.*

Proof. Let n_i be the pseudo-height of $\Psi(\tau_i)$. By the construction of Π_0 we know that:

$$Even(\varphi_j^{W_l^{\eta_0}\restriction n_0}) \supset \sigma_p \ \& \ Even(\varphi_j^{W_l^{\eta_1}\restriction n_1}) \supset \sigma_q.$$

Since σ_p is incomparable with σ_q, η_0 and η_1 cannot be the same. By assumption η_0 is shorter, and if we assume that τ_0 is an initial segment of A, then it will contradict Fact III.4.1(6), as $W_l^{\eta_0} \restriction n_0$ is never going to change along any path $C \in [S]$ extending η_0, but it does change since $W_l^{\eta_1} \restriction n_1$ computes an incomparable string via φ_j. □

Finally we find all strings τ in $X = \cup_{i=0}^{r} X_i$, compute all pseudo-leaves of $\Psi(\tau)$ for each one and pick those τ's whose $\Psi(\tau)$ have a pseudo-leaf compatible with C. By the previous lemma, those τ's that have the longest C-compatible pseudo-leaf in $\Psi(\tau)$ must be in exactly one of those X_i's, and the corresponding σ_i is the next initial segment α_{k+1} of A. This finishes the proof. □

Finally this corollary easily follows from our main theorem and Theorem II.3.2:

Corollary III.4.3. *A degree* $\mathbf{a}$ *is* **ANR** *if and only if every* $\mathbf{b} \geq \mathbf{a}$ *is* **RRE**.

III.5 Applications and Remarks

III.5.1 Strong minimal covers

In [ASDWY09] the authors also asked whether $\mathcal{B}$ has a minimal element. The answer is no, as every **ANR** degree has another **ANR** degree strictly below it (e.g. by [DJS96, Theorem 2.5]).

To see that there is no maximal **AR** degree, note that any Spector minimal cover of an **AR** degree is hyperimmune-free relative to that degree, and so it is also **AR**.

Alternatively, in the proof of Theorem III.2.1 one can easily adapt requirements from the minimal cover construction to guarantee that B is minimal above A (i.e. the interval $(\mathbf{a}, \mathbf{b})$ is empty).

To be explicit, we add the following list of requirements:

P_e φ_e^B is total $\Rightarrow$ ($\varphi_e^B \leq_T A$ or $B \leq_T \varphi_e^B \oplus A$).

Given T and P_e, we ask whether the following holds:

$$\exists\sigma\forall\tau_0,\tau_1 \supset \sigma[\neg(\exists x(\varphi_e^{T(\tau_0)}(x)\downarrow \neq \varphi_e^{T(\tau_1)}(x)\downarrow))].$$

If so, we take $T' = \mathbf{FSub}(T,\sigma)$ and T' forces φ_e^B to be A-recursive whenever it is total. Otherwise, we take $T' = \mathbf{Spl}(T,e)$ and T' forces $B \leq_T \varphi_e^B \oplus A$.

Now we have the following theorem:

Theorem III.5.1. *For every array recursive degree* $\mathbf{a}$*, there is a minimal cover* $\mathbf{b} > \mathbf{a}$ *which is not* **RRE**.

This $\mathbf{b}$ is certainly **AR** by Theorem II.3.2. In addition, if $\mathbf{a}$ is an r.e. degree, then the $\mathbf{b}$ in Theorem III.5.1 is actually a strong minimal cover of $\mathbf{a}$ (i.e. $\mathcal{D}(<\mathbf{b}) = \mathcal{D}(\leq \mathbf{a})$). If not, then there is a $\mathbf{c} < \mathbf{b}$ which is not recursive in $\mathbf{a}$. Consequently $\mathbf{b} = \mathbf{a} \vee \mathbf{c}$ and $\mathbf{b}$ would be r.e. in $\mathbf{c}$. This would make $\mathbf{b}$ an **RRE** degree.

Note that no **ANR** degree has a strong minimal cover (e.g., they all have the cupping property by [DJS96, Theorem 3.4]). We thus have a new proof of the following result from [Ish99].

Corollary III.5.2 (Ishmukhametov)**.** *An r.e. degree has a strong minimal cover if and only if it is array recursive.*

In fact, this result can be generalized to the n-**REA** degrees.

Definition III.5.3. A degree is 1-**REA** if it is an r.e. degree. A degree is $(n+1)$-**REA** if it is r.e. in and strictly above an n-**REA** degree.

Corollary III.5.4. *An* n-**REA** *degree has a strong minimal cover if and only if it is array recursive.*

Proof. Suppose we are given $\mathbf{a}_n > \mathbf{a}_{n-1} > \cdots > \mathbf{a}_1 > \mathbf{a}_0 = \mathbf{0}$ where each $\mathbf{a}_{i+1}$ is r.e. in and strictly above $\mathbf{a}_i$, and $\mathbf{a}_n$ is **AR**. By Theorem III.5.1 we can find a minimal cover $\mathbf{b}$ of $\mathbf{a}_n$ such that $\mathbf{b}$ is not **RRE**. We claim that this $\mathbf{b}$ is actually a strong minimal cover of $\mathbf{a}_n$.

Suppose not, then there is a $\mathbf{c} < \mathbf{b}$ which is not recursive in $\mathbf{a}_n$. Consider $\mathbf{c}_1 = \mathbf{c} \vee \mathbf{a}_1$. $\mathbf{c}_1$ must be strictly below $\mathbf{b}$, as otherwise $\mathbf{b}$ would be r.e. in and above $\mathbf{c}$. Inductively define $\mathbf{c}_{i+1} = \mathbf{c}_i \vee \mathbf{a}_{i+1}$, and using the same argument we know that $\mathbf{c}_{i+1}$ is strictly below $\mathbf{b}$. Then $\mathbf{c}_n = \mathbf{c} \vee \mathbf{a}_1 \vee \mathbf{a}_2 \vee \cdots \vee \mathbf{a}_n = \mathbf{c} \vee \mathbf{a}_n$ would be a degree strictly between $\mathbf{a}_n$ and $\mathbf{b}$. This contradicts the fact that $\mathbf{b}$ is a minimal cover of $\mathbf{a}_n$. □

III.5.2 More on definability

[DJS96] raised the question whether the **ANR** degrees are definable in the Turing degrees with only the order relation. From this point of view, Corollary III.4.3 asserts that the **ANR** degrees are definable in the Turing degrees with the order relation and with an additional predicate "$\mathbf{x}$ is r.e. in and above $\mathbf{y}$".

Theorem III.2.1 also shows that a degree $\mathbf{a}$ is **ANR** if and only if every degree $\mathbf{b} \geq \mathbf{a}$ is n-**REA** relativized to some degree strictly below $\mathbf{b}$ for some n. Hence the **ANR** degrees are also definable from the predicate "$\mathbf{x}$ is n-**REA** in $\mathbf{y}$ for some

n", which might be easier to define than "$\mathbf{x}$ is r.e. in and above $\mathbf{y}$". Shore ([Sh85, Question 5.6]) conjectured that a degree is $\mathbf{x}$ is n-**REA** if and only if for every $\mathbf{y}$, $\mathbf{x} \vee \mathbf{y}$ is not a minimal cover of $\mathbf{y}$.

Since it is known that the jump is definable in the Turing degrees, one might try to define the **ANR** degrees by replacing "being r.e. (Σ_1) in a degree strictly below" with "being Δ_2 relativized to some degree strictly below". However, this attempt fails. First note that every nonempty Π^0_1 class of infinite binary strings has a member of array recursive degree. For example, in the proof of the low basis theorem ([JSo72, Theorem 2.1]), $\mathbf{0}'$ can decide the jump of the low path G only using recursively bounded information, i.e., G' is weak truth table below $\mathbf{0}'$. For each computation $\varphi^G_a(b)$ one can recursively translate it into $\varphi^G_n(n)$ for some n, and so it is easy to see that there is a function $f \leq_T G \oplus K$ such that the use from K is recursively bounded and $f(n)$ bounds all convergent $\varphi^G_a(b)$ for every pair $a, b \leq n$. Such a function f then dominates all functions recursive in G and therefore $\mathbf{deg}(G)$ is array recursive (see [CSh12]).

There is a Π^0_1 class in 2^ω such that every member of it is PA and every PA degree has the cupping property ([Ku94]). It follows that there is a low array recursive PA degree $\mathbf{a}$ such that degree $\mathbf{b}$ above $\mathbf{a}$ is the join of $\mathbf{a}$ and some $\mathbf{c} < \mathbf{b}$, and so $\mathbf{b} \leq \mathbf{c} \vee \mathbf{0}' \leq \mathbf{c}'$, i.e., $\mathbf{b}$ is Δ_2 relativized to $\mathbf{c}$.

III.5.3 Join property and PA degrees

There has been various results about the join property. For example, in [GMS04] it was proved that $\mathbf{GH}_1$ degrees have the complementation property, which is stronger than the join property; later it is proved that $\overline{\mathbf{GL}_2}$ degrees have the join property ([DGLMxx]). So it seems that degrees that are higher above have the join property, and then it is natural to ask what is the best upward-closed property that ensures the cupping property. Some natural guesses might be the **ANR** ones or the **PA** ones. However, Lewis ([Lewb]) proved that in fact all low **FPF** degrees fail to have the join property, and as a consequence there are **PA** degrees and **ANR** degrees that fail to have the join property. Here we give an alternative proof that there is a **PA** degree without the join property by our main theorem.

Following the discussion in the previous subsection, we have an array recursive **PA** degree $\mathbf{a}$ and by Kučera's Theorem ([Ku86]) there is a nonrecursive r.e. degree $\mathbf{d} \leq \mathbf{a}$. By Theorem III.2.1, there is a degree $\mathbf{b} \geq \mathbf{a}$ which is not **RRE**. This $\mathbf{b}$ is still **PA** since **PA** degrees are upward closed, and it is easy to see that $\mathbf{b}$ fails to have the join property via $\mathbf{d}$.

CHAPTER IV
DOMINATION AND DEFINABILITY: SOME NEGATIVE RESULTS (JOINT WITH SHORE)

IV.1 Introduction

Given two functions f and g, we say f *dominates* g if $f(x) \geq g(x)$ for all but finitely many x. Domination plays an important role in classical recursion theory. The intuition is that, a function which dominates (or is not dominated by) some certain fast-growing function is powerful enough to run certain corresponding computations; in contrast, if a degree only contains slow-growing functions then the degree cannot be too powerful. For example, one can define a function $f(n) \leq_T \mathbf{0}'$ by letting $f(n)$ be the number of steps in the computation of $\varphi_n(n)$ if it converges, and 0 if it diverges. Then it is easy to see that a degree contains a function which dominates f if and only if it is above $\mathbf{0}'$.

The complementary pair of hyperimmune and hyperimmune-free degrees was originally defined by a property involving intersection with uniform sequences of recursive sets, but it is more natural to define them in terms of domination properties. A degree is *hyperimmune* if it computes a function which is not dominated by any recursive function; and a degree is *hyperimmune-free* if it is not hyperimmune, i.e., every function recursive in it is dominated by a recursive function. More recently "$\mathbf{0}$-dominated" has often been used in place of "hyperimmune-free".

Miller and Martin's paper ([MM68]) proved a number of interesting theorems about these two classes of degrees. For example, all nonrecursive degrees below $\mathbf{0}'$ are hyperimmune, but there are nonrecursive hyperimmune-free degrees below $\mathbf{0}''$. To list a few more theorems: Jockusch and Soare [JSo72] proved a basis theorem for hyperimmune-free degrees: every nonempty Π^0_1 class contains a hyperimmune-free member. Lewis ([Lew07]) proved that every hyperimmune-free non-FPF degree has a strong minimal cover (a degree is FPF if it computes a function f such that $\varphi_e \neq \varphi_{f(e)}$ for any e).

Miller and Martin suggested that the project of "characterizing a notion" (such as hyperimmuneness) is to give definitions of such a combinatorial property in the language of degree theory. For example, if we let $\mathcal{L}_0 = \{\leq\}$, the language of partial order, then we can ask whether there is an $\mathcal{L}_0$-formula $\varphi(x)$ defining the hyperimmune degrees, i.e., $\mathbf{x}$ is hyperimmune if and only if $\varphi(x)$ holds. Note that Miller and Martin allow constants in the formulae, so we can actually write out some nontrivial examples even by quantifier-free formulae. For example, by the Friedberg's Jump Inversion Theorem ([Fri57]), $\{\mathbf{d} : \exists \mathbf{a} : \mathbf{a}' = \mathbf{d}\}$ is the same as $\{\mathbf{d} : \mathbf{d} \geq \mathbf{0}'\}$, so there is a quantifier-free $\mathcal{L}_0$-formula defining the degrees which are jumps of other degrees. As another example, let $\mathbf{a}$ and $\mathbf{b}$ be the exact pair of the sequence $\mathbf{0}, \mathbf{0}', \mathbf{0}'', \ldots$ (see [Sp56]), then we can get a quantifier-free definition of the arithmetic degrees using $\mathbf{a}$ and $\mathbf{b}$.

Similarly we can extend the language, using the notation in [MM68], to $\mathcal{L}_1 = \{\leq, '\}$ and $\mathcal{L}_2 = \{\leq, ', R\}$ where R stands for the relation "r.e. in" (aRb if a is r.e. in b). By results of Shore and Slaman ([SS99] and [Sh07]) the jump $'$ is definable in the language of partial order $\mathcal{L}_0$, so $\mathcal{L}_0$ and $\mathcal{L}_1$ have the same definability power, but they are still different in terms of quantifier complexity (see [Sh07]). It is still an open question whether R is definable in $\mathcal{L}_0$.

Miller and Martin showed that in $\mathcal{L}_0$, one cannot define the hyperimmune degrees using a quantifier-free formula, and they conjectured that if we use $\mathcal{L}_1$, this is still true. We confirm this conjecture in Section IV.2 (we do not know whether it is still true that no quantifier-free $\mathcal{L}_2$-formula defines the hyperimmune degrees). In addition, we can also show that in $\mathcal{L}_0$, there is no one-quantifier formula defining the hyperimmune degrees (see Sections IV.3 and IV.4).

Besides the complementary pair of hyperimmune and hyperimmune-free degrees, array nonrecursive and array recursive also form an important and interesting pair of domination properties. To define them, we first need the *modulus function* m_K *of* K, the halting problem, where $m_K(n)$ is defined as the least stage s at which the initial segment of length n settles down in the standard enumeration of K. A degree is *array recursive* if every function recursive in it is dominated by m_K; and a degree is *array nonrecursive* (**ANR**) if there is a function recursive in it which is not dominated by m_K.

The notion of **ANR** degrees was introduced in [DJS96] to generalize a jump class namely $\overline{\mathbf{GL_2}}$ degrees. See [DJS96] and [CSh12] for discussions and some theorems about **ANR** degrees. In terms of definability, though it is unknown whether the **ANR** degrees have an $\mathcal{L}_0$-definition, it is actually a recent result that they are definable in $\mathcal{L}_2$ by a Π_2 formula ([CSh12] and [Cai12], see also Chapters II and III). So it might be interesting to discuss whether this definition is sharp in terms of quantifier complexity. We do not know the answer but guess that it probably is. In the last section we briefly discuss the nondefinability of the array nonrecursive degrees and show that what we have done for hyperimmune degrees all applies to array nonrecursive degrees, i.e., there is no quantifier-free $\mathcal{L}_1$ or one-quantifier $\mathcal{L}_0$ formula which defines the array nonrecursive degrees.

IV.2 Nondefinability in $\mathcal{L}_1$: Quantifier-free Case

As we mentioned in the introduction, Miller and Martin showed that in $\mathcal{L}_0$ one cannot define the hyperimmune degrees by a quantifier-free formula. Briefly, they first assume that there is such a formula defining the hyperimmune degrees and then find a hyperimmune-free degree which satisfies the same formula for a contradiction. They then conjectured that it is also true for $\mathcal{L}_1$, i.e., with the jump operator added one still cannot define the hyperimmune degrees by a quantifier-free formula.

The initial difficulty is that we do not have a nice characterization of the jumps of hyperimmune-free degrees, and it would be difficult to construct a hyperimmune-free degree whose jump is a fixed degree or satisfies some given properties. However, we do know the jumps of hyperimmune degrees: every degree above $\mathbf{0}'$ is a jump of a hyperimmune degree. (In fact we conjecture that every degree above $\mathbf{0}'$ is the jump of a hyperimmune minimal degree.) So it seems that it is better to handle this problem in the direction opposite to Miller and Martin's approach, and in fact from this point of view it turns out to be fairly easy.

To get a contradiction, we start with a quantifier-free $\mathcal{L}_1$-formula $\varphi(x)$ which purportedly defines the hyperimmune-free degrees, i.e., a degree $\mathbf{x}$ is hyperimmune-free if and only if $\varphi(\mathbf{x})$ holds.

We write φ in disjunctive normal form, i.e., $\psi_0 \vee \psi_1 \vee \ldots \vee \psi_k$, where each ψ_i is a conjunction of atomic formulae. We regard all the jumps of constants as constants, i.e., for each constant c, if c' appears in the formula we will add a new constant $\tilde{c}$

and add $c' = \tilde{c}$ as a conjunct (for convenience we still write c' but regard it as a constant).

Without loss of generality we assume that each ψ_i is a complete diagram about all the degrees and their jumps mentioned in φ, i.e., for each (ordered) pair of degrees $\mathbf{a}$ and $\mathbf{b}$, we either have $\mathbf{a} \leq \mathbf{b}$ or $\mathbf{a} \not\leq \mathbf{b}$.

In addition, we remove all those ψ_i's that are redundant, i.e., there are no hyperimmune-free degrees satisfying ψ_i. We also remove all conjuncts that are trivial, for example $a \leq x$ where $\mathbf{a} = \mathbf{0}$, i.e., we remove all conjuncts that are satisfied by all degrees. This allows us to remove formulae that involve a degree and its own jump (or iterated jumps) such as $x \leq x'$ or $x \not\leq x'$ (either no degree or all degrees satisfy these formulae).

First we argue that there must be at least one ψ_i which does not have any conjunct of the form $x \leq a$ or $a \leq x$. Otherwise there is a finite list of degrees $\mathbf{a_0}, ..., \mathbf{a_k}$ such that every hyperimmune-free degree is comparable with at least one of them. However there always exists a hyperimmune-free minimal degree which avoids all these lower cones (and automatically avoids upper cones by minimality), so this cannot happen.

Now we pick a formula ψ_i (for simplicity we use ψ to denote it) which does not have any conjunct of the form $x \leq a$ or $a \leq x$. We can write ψ as follows:

$$P(x') \wedge \bigwedge_{i \in I_1} (x \not\leq a_i) \wedge \bigwedge_{i \in I_2} (b_i \not\leq x),$$

where $P(x')$ is the collection of all conjuncts about x' (or higher jumps such as x'' and x''') but not about x. Note that we have removed all trivial atomic formulae which include any atomic formulae about both x and x' (or higher jumps) such as $x \leq x'$.

So by our assumption there is a hyperimmune-free degree $\mathbf{x}$ which satisfies ψ. Our plan is to build a hyperimmune degree $\mathbf{y}$ whose jump is $\mathbf{x}'$ (so $\mathbf{y}'$ automatically satisfy $P(x')$) and $\mathbf{y}$ also satisfies all other conjuncts in ψ. Then we get a contradiction because $\mathbf{y}$ satisfies ψ and hence φ but is hyperimmune.

To get such a degree we build $\mathbf{y}$ low above $\mathbf{x}$: this guarantees that its jump is $\mathbf{x}'$ and it is hyperimmune since it is relatively Δ_2 ([MM68, Theorem 1.2]). We use finite forcing to build initial segments σ_i recursively in $\mathbf{x}'$ and in the end $\cup_i \sigma_i = Y$. We take $\mathbf{y} = \deg(X \oplus Y)$ where $X \in \mathbf{x}$, so guarantee $\mathbf{y} \geq \mathbf{x}$. In the construction we force the jump of $X \oplus Y$ step by step, therefore $(X \oplus Y)' \leq_T X'$, i.e., $\mathbf{y}' = \mathbf{x}'$. To finish the proof we discuss how to satisfy all the requirements in the conjuncts listed in ψ.

Avoiding lower cones ($y \not\leq a_i$) is automatic: each $\mathbf{a}_i$ is not above $\mathbf{x}$, so it cannot be above $\mathbf{y}$ as $\mathbf{y} \geq \mathbf{x}$.

For upper cones, note that we only need to consider those $\mathbf{b_i}$'s that are below $\mathbf{x}'$: otherwise $\mathbf{y}$ cannot be above such $\mathbf{b_i}$. Now given σ we can ask whether there are two extensions τ_0 and τ_1 of σ such that $(X \oplus \tau_0 |_e X \oplus \tau_1)$, i.e., there is an n such that $\varphi_e^{X \oplus \tau_0}(n) \downarrow \neq \varphi_e^{X \oplus \tau_1}(n)$: If so we take the one which differs with a fixed $B_i \in \mathbf{b_i}$; if not we can easily argue that the function $\varphi_e^{X \oplus Y}$ is recursive in X if it is total, and so it is not equal to B_i as $\mathbf{b_i} \not\leq \mathbf{x}$. In either case we can satisfy the requirement $\mathbf{b_i} \not\leq \mathbf{y}$.

IV.3 Nondefinability in $\mathcal{L}_0$: Π_1 Case

We want to show that in $\mathcal{L}_0$, one cannot define the hyperimmune degrees by a one-quantifier formula, i.e., we can define neither the hyperimmune-free nor the hyperimmune degrees by a Σ_1 formula.

We start with the easy case: there is no Σ_1 formula which defines the hyperimmune-free degrees (i.e., there is no Π_1 formula which defines the hyperimmune degrees). Following an argument similar to Section IV.2, we first assume that $\exists d_0 \exists d_1 \ldots \exists d_n \varphi(x, d_0, d_1, \ldots, d_n)$ defines the hyperimmune-free degrees using constants $c_0, c_1, \ldots, c_m$. We write φ in disjunctive normal form $\psi_0 \vee \psi_1 \vee \cdots \vee \psi_k$ where each ψ_i is a conjunction of atomic formulae. We also assume that each ψ_i is a complete diagram. Without loss of generality we can also assume that each ψ_i can be realized by at least one nonrecursive hyperimmune-free degree $\mathbf{x}$, i.e., we can find witnesses $\mathbf{d_0}, \mathbf{d_1}, \ldots, \mathbf{d_n}$ such that $\psi_i(\mathbf{x}, \mathbf{d_0}, \mathbf{d_1}, \ldots, \mathbf{d_n})$ holds.

By the same argument as in Section IV.2 we can find a ψ among these ψ_i's such that it does not have any atomic formula of the form $x \leq c_i$ or $c_j \leq x$, i.e., x is not in any of the lower cones or upper cones of the c_i's.

We pick a hyperimmune-free degree $\mathbf{x}$ and a sequence of witnesses $\mathbf{d_0}, \mathbf{d_1}, \ldots, \mathbf{d_n}$ which realize ψ. Now our plan is to find a new $\mathbf{y}$ which is hyperimmune and a new sequence of witnesses $\mathbf{d_0^*}, \mathbf{d_1^*}, \ldots, \mathbf{d_n^*}$ which satisfy the same ψ.

We will build such a $\mathbf{y}$ above $\mathbf{x}$; for those $\mathbf{d_i}$'s such that $\mathbf{x}$ is not below $\mathbf{d_i}$, we will have $\mathbf{d_i^*} = \mathbf{d_i}$; and for $\mathbf{d_i}$'s above $\mathbf{x}$, we will build new degrees $\mathbf{d_i^*}$ above $\mathbf{y}$. For convenience we let $\mathbf{d_0}, \ldots, \mathbf{d_l}$ be the $\mathbf{d_i}$'s which are above $\mathbf{x}$ and regard the other $\mathbf{d_i}'s$ as constants in our construction.

It is easy to see that ψ does not have $d_i \leq c$ as a conjunct for some $i \leq l$ and constant c. The reason is that otherwise we would have $x \leq c$ by transitivity which contradicts our assumption.

So we need to satisfy the following list of requirements:

1. $\mathbf{d_i^*} \leq \mathbf{d_j^*}$ for each $d_i \leq d_j$ appearing in ψ, $i, j \leq l$
2. $\mathbf{d_i^*} \not\leq \mathbf{d_j^*}$ for each $d_i \not\leq d_j$ appearing in ψ, $i, j \leq l$
3. $\mathbf{d_i^*} \not\leq \mathbf{c_{i,j}}$ for each $d_i \not\leq c_{i,j}$ appearing in ψ, $i \leq l$
4. $\mathbf{c_j} \not\leq \mathbf{d_i^*}$ for each $c_j \not\leq d_i$ appearing in ψ, $i \leq l$
5. $\displaystyle\bigvee_{j: c_j \leq d_i} \mathbf{c_j} \leq \mathbf{d_i^*}$ for each $i \leq l$

For convenience we let $\mathbf{a_i}$ be the join of $\mathbf{c_j}$'s in the fifth requirement, that is, we need to make $\mathbf{a_i} \leq \mathbf{d_i^*}$. Note that if $\mathbf{d_i} \leq \mathbf{d_j}$ then $\mathbf{a_i} \leq \mathbf{a_j}$ by transitivity.

Let $\mathbf{g_{-1}}, \mathbf{g_0}, \ldots, \mathbf{g_l}$ be a sequence of mutually 1-generic degrees relative to all other degrees mentioned, i.e., each $\mathbf{g_i}$ is 1-generic relative to the join of $\mathbf{c_{i,j}}$'s, $\mathbf{d_i}$'s, $\mathbf{x}$ and $\bigvee_{j \neq i} \mathbf{g_j}$.

Then we let:

$$\mathbf{d_i^*} = \mathbf{a_i} \vee \bigvee_{d_j \leq d_i} \mathbf{g_j} \vee \mathbf{y} \text{ and } \mathbf{y} = \mathbf{x} \vee \mathbf{g_{-1}}.$$

We claim that these $\mathbf{d_i^*}$'s satisfy all the requirements.

It is easy to see that the requirements of types (1) and (5) are automatically satisfied. Requirements of types (2) and (3) are easily satisfied by 1-genericity. For example, if we have some $d_i \not\leq d_j$ in ψ, then we know that $\mathbf{g_i}$ is in the join of $\mathbf{d_i^*}$ but not in the join of $\mathbf{d_j^*}$ as in our formula above. Therefore $\mathbf{g_i}$ is 1-generic relative to $\mathbf{d_j^*}$ and so $\mathbf{d_i^*} \not\leq \mathbf{d_j^*}$.

For requirements of type (4), we need the following easy lemma:

Lemma IV.3.1. *If $\mathbf{c} \not\leq \mathbf{a}$ and $\mathbf{g}$ is 1-generic in $\mathbf{c} \vee \mathbf{a}$, then $\mathbf{c} \not\leq \mathbf{a} \vee \mathbf{g}$.*

Proof. Define sets $S_e = \{\sigma : \exists n(C(n) \neq \varphi_e^{A\oplus\sigma}(n)\downarrow)\}$ which is $\Sigma_1^{A\oplus C}$. By 1-genericity, either G meets S_e, which makes $C \neq \varphi_e^{A\oplus G}$; or there is an initial segment σ of G which forces G to avoid S_e, i.e., there is no extension τ of σ which is in S_e: in which case if $\varphi_e^{A\oplus G}$ is total, then it is recursive in A and so is not equal to C. □

Now for requirements of type (4), if we have some $c_j \not\leq d_i$ in ψ, then we know that $\mathbf{c_j} \not\leq \mathbf{a_i} \vee \mathbf{x}$ (since $\mathbf{c_j} \not\leq \mathbf{d_i}$ and $\mathbf{a_i} \vee \mathbf{x} \leq \mathbf{d_i}$). By iterating the lemma above and adding the $\mathbf{g_i}$'s appearing in the formula defining $\mathbf{d_i^*}$, one can see that $\mathbf{c_j} \not\leq \mathbf{d_i^*}$.

Finally $\mathbf{y}$ is hyperimmune since 1-generic degrees are hyperimmune and being hyperimmune is upward closed in the degrees.

IV.4 Nondefinability in $\mathcal{L}_0$: Σ_1 Case

Now we show that there is no Σ_1 formula which defines the hyperimmune degrees. We follow the same outline and use the same notations as in the previous section. That is, we have a hyperimmune degree $\mathbf{x}$ and a sequence of witnesses $\mathbf{d_0}, \ldots, \mathbf{d_n}$ which satisfy a Σ_1 formula $\exists d_0 \exists d_1 \ldots \exists d_n \varphi(x, d_0, d_1, \ldots, d_n)$ with constants $c_0, \ldots, c_m$. Similarly we write φ in disjunctive normal form and pick one disjunct ψ (which is a conjunction of atomic formulae) which says that $\mathbf{x}$ is not in the lower cone or the upper cone of any $\mathbf{c_i}$. Our plan is to find some new $\mathbf{d_0^*}, \ldots, \mathbf{d_n^*}$ and a new hyperimmune-free $\mathbf{y}$ which satisfy ψ.

To make our construction smoother, we use the following $*$-notation: given any degree $\mathbf{z}$ appearing in $\psi(\mathbf{x})$, $\mathbf{z}^*$ denotes the corresponding degree appearing in $\psi(\mathbf{y})$. That is, if $\mathbf{z} = \mathbf{x}$, then $\mathbf{z}^* = \mathbf{y}$; if $\mathbf{z} = \mathbf{d_i}$, then $\mathbf{z}^* = \mathbf{d_i^*}$; and if $\mathbf{z} = \mathbf{c_i}$, then $\mathbf{z}^* = \mathbf{c_i}$. Similarly, if we have a set S of degrees, then $S^* = \{\mathbf{z}^* : \mathbf{z} \in S\}$. We call the non-$*$ degrees *originals* and their corresponding $*$-versions their *photocopies*.

We need to impose some extra conditions on ψ before beginning our construction. Our plan is as follows: First we find $\mathbf{y}$ and some degrees $\mathbf{d_i^*}$'s below it such that the these degrees form the same partial order S^* as the partial order structure S of $\mathbf{x}$ and the $\mathbf{d_i}$'s below $\mathbf{x}$. Then use a construction similar to the one used in the previous section to build the other $\mathbf{d}^*$'s above members of S^*.

For convenience we let $\mathbf{d_0}, \ldots, \mathbf{d_l}$ be the $\mathbf{d}$'s below $\mathbf{x}$. The problem is that, for example, if $\mathbf{d_0}$ is below some $\mathbf{c_i}$, then it is difficult to build our $\mathbf{d_0^*}$: in particular, it might be very hard to make it hyperimmune-free as to make $\mathbf{y}$ hyperimmune-free.

We thus in addition assume that in ψ, there is no formula of the form $d_i \le c$ for $0 \le i \le l$ and some constant c. The reason we can make this assumption is that, if no disjunct ψ has this property, then we can construct a hyperimmune degree $\mathbf{z}$ which avoids all the following upper cones: for any constant $\mathbf{c}$ and any $\mathbf{d} \le \mathbf{c}$ we have $\mathbf{z} \not\ge \mathbf{d}$. This is a countable list of requirements and we can even construct a hyperimmune minimal degree which automatically avoids these cones. It would contradict the fact that our formula defines the hyperimmune degrees.

Now we follow our plan: first we build $\mathbf{y}$ and $\mathbf{d_0^*}, \dots, \mathbf{d_l^*}$ which form the same partial order S^* as S (the structure of $\mathbf{x}$ and $\mathbf{d_0}, \dots, \mathbf{d_l}$), and make sure that $\mathbf{y}$ is hyperimmune-free and the following holds:

($*$) For any subset of degrees T^* of S^*, any finite set R of constants and any fixed degree $\mathbf{z}$ among the $\mathbf{x}$, $\mathbf{c_i}$'s and $\mathbf{d_i}$'s, if the join of all degrees in $T \cup R$ is not above $\mathbf{z}$, then the join of all degrees in $T^* \cup R$ is not above $\mathbf{z}^*$.

This property seems ad hoc, but in fact it is natural and essential in the construction. For example, it implies that $\mathbf{y}$ is not above or below any of the $\mathbf{c_i}$'s. Later this property will guarantee that when we construct other $\mathbf{d}^*$'s, we do not get a degree which is too high by joining it with some degrees we already have.

First of all, to see that we can carry out the construction of S^* with property ($*$), we can use, for example, the uniform tree constructions using usl tables (which are used to build initial segments, see [Ler83, Chapter VI]) with some extra requirements to satisfy ($*$). Given a tree and a requirement $Z^* \neq \varphi_e^{\bigoplus D_i^* \oplus \bigoplus C_i}$, if Z is some constant C, then we simply ask whether we can find e-splittings on the tree for $\varphi_e^{\bigoplus D_i^* \oplus \bigoplus C_i}$: if so we can choose one to diagonalize against Z; if not then it is easy to see that $Z \le_T \bigoplus C_i$ which contradicts our assumption. If Z is X or some D_j, then we pick two nodes τ_0, τ_1 on the tree which are congruent modulo $\bigoplus D_i^*$ but disagree on Z^* (say, at x). This is possible since $Z \not\le_T \bigoplus D_i$ and by our usl representation we are able to make $Z^* \not\le_T \bigoplus D_i^*$. Then we try to extend τ_0 to τ' where we have a convergent $\varphi_e^{\bigoplus D_i^* \oplus \bigoplus C_i}(x)$. If there is no such τ', then we take the full subtree above τ_0 to satisfy our requirement. If there is such a τ' then we extend τ_1 in the same way to τ'' such that τ' and τ'' are congruent modulo $\bigoplus D_i^*$ (by uniformity). Therefore at τ' and τ'', the computations $\varphi_e^{\bigoplus D_i^* \oplus \bigoplus C_i}(x)$ give the same value. We can pick one whose $Z^*(x)$ is different from this value and take the full subtree above it.

Then we need to show that these $\mathbf{d_i^*}$'s satisfy the same relations with the constants as their originals: For each $\mathbf{d_i}$ ($i = 1, 2, \dots, l$), it is not in any of the upper or lower cone of constant $\mathbf{c}$, so we only need to show that $\mathbf{d_i^*}$ is incomparable with any of the constants. It is not difficult to see that these relations are guaranteed by property ($*$).

Now let $\mathbf{d_{l+1}}, \dots, \mathbf{d_k}$ be the $\mathbf{d}$'s that are above any degree in S. We need to build new $\mathbf{d_{l+1}^*}, \dots, \mathbf{d_k^*}$ which satisfy the same diagram (we regard the remaining $\mathbf{d}$'s as constants). This construction is almost the same as the one in Section IV.3, by replacing one degree $\mathbf{y}$ with a sequence of degrees $\mathbf{y}, \mathbf{d_0^*}, \dots, \mathbf{d_l^*}$. We find $\mathbf{g_{l+1}}, \dots, \mathbf{g_k}$ which are mutually 1-generic, and we let $\mathbf{d}^*$'s be the following joins:

$$\mathbf{d_i^*} = \bigvee_{\mathbf{d_j} \le \mathbf{d_i}} \mathbf{g_j} \vee \bigvee_{\mathbf{z} \le \mathbf{d_i}, \mathbf{z} \in S \cup R} \mathbf{z}^*,$$

where R is the set of all constants, and in particular, $\bigvee_{\mathbf{z} \le \mathbf{d_i}, \mathbf{z} \in R} \mathbf{z}^*$ here is analog to the $\mathbf{a_i}$ as in Section IV.3.

The arguments and proofs are very similar to the ones in Section IV.3. We need to show that each $\mathbf{d_i^*}$ (among $\mathbf{d_{l+1}^*}, \ldots, \mathbf{d_k^*}$) satisfies the same relations with $\mathbf{z}^*$ as its original $\mathbf{d_i}$ with $\mathbf{z}$. We discuss the cases:

If $\mathbf{z}$ is below $\mathbf{d_i}$, then it is automatic by the definition of $\mathbf{d_i^*}$ that $\mathbf{z}^* \leq \mathbf{d_i^*}$. If $\mathbf{z}$ is above $\mathbf{d_i}$, then it can only be the case that $\mathbf{z}$ is some $\mathbf{d_j}$, and we are back to the first case. So we only need to discuss the case that $\mathbf{z}$ is incomparable with $\mathbf{d_i}$. Here we further divide into subcases:

If $\mathbf{z}$ is $\mathbf{x}$ (or a constant $\mathbf{c}$, or a degree $\mathbf{d}$ in S, as the arguments for them are essentially the same), we need to show that $\mathbf{d_i^*}$ is incomparable with $\mathbf{y}$: first $\mathbf{y}$ is not above $\mathbf{d_i^*}$ because $\mathbf{g_i}$ is not below $\mathbf{y}$; $\bigvee_{\mathbf{z} \leq \mathbf{d_i}, \mathbf{z} \in S \cup R} \mathbf{z}^*$ is not above $\mathbf{y}$ by property $(*)$ (since their originals do not join to some degree above $\mathbf{x}$), and so by Lemma IV.3.1 joining it with some mutually 1-generic degrees does not make $\mathbf{d_i^*}$ above $\mathbf{y}$.

If $\mathbf{z}$ is some other $\mathbf{d_j}$ among $\mathbf{d_{l+1}}, \ldots, \mathbf{d_k}$, then since $\mathbf{d_i}$ and $\mathbf{d_j}$ are incomparable, $\mathbf{g_i}$ and $\mathbf{g_j}$ only appear in one join but not the other. By 1-genericity $\mathbf{d_i^*}$ and $\mathbf{d_j^*}$ are incomparable.

This gives the desired contradiction that $\mathbf{y}$ with witnesses $\mathbf{d}^*$'s satisfy the same formula $\exists d_0 \exists d_1 \ldots \exists d_n \varphi(x, d_0, d_1, \ldots, d_n)$ but $\mathbf{y}$ is hyperimmune-free.

IV.5 Change to Array Nonrecursive Degrees

First, in Section IV.4, the argument is the same since $\mathbf{y}$ being hyperimmune-free automatically guarantees array recursiveness. In Sections IV.2 and IV.3, if we want to make our new $\mathbf{y}$ array nonrecursive in addition to hyperimmune, we only need to code in some numbers larger than m_K infinitely often. This can be easily handled since we are using finite forcing and we have $\mathbf{x}'$ or higher oracle. Therefore the theorems we prove above all apply to the array nonrecursive degrees, i.e., we cannot define them by quantifier-free formulae in $\mathcal{L}_1$ or by one-quantifier formulae in $\mathcal{L}_0$.

CHAPTER V
HYPERIMMUNE MINIMAL DEGREE AND ANR 2-MINIMAL DEGREE

This chapter is mainly based on a paper in the Notre Dame Journal of Formal Logic (Volume 51, Number 4, 2010, Pages 443–455).

V.1 Introduction

In this chapter, we develop a new method for constructing hyperimmune minimal degrees and construct an **ANR** degree which is a minimal cover of a minimal degree.

The primary motivation is a long-standing question of Yates:

Question V.1.1 (Yates)**.** *Does every minimal degree have a strong minimal cover?*

A recent and remarkable result in the positive direction for Question V.1.1 is Lewis' Theorem [Lew07, Theorem 4.3].

Theorem V.1.2 (Lewis)**.** *Every hyperimmune-free degree which is not* **FPF** *has a strong minimal cover.*

So in order to give a negative answer to Question V.1.1, we have to look for minimal degrees that are either **FPF** or hyperimmune.

On one hand, it was also an old question whether **FPF** minimal degrees exist. Kumabe gave a positive solution to this question in an unpublished note and Lewis simplified the proof in [KLxx].

On the other hand, the observation might seem easy: it is well known that there is a minimal degree below $\mathbf{0}'$ ([Sac66, Theorem 1]) and any nonrecursive degree below $\mathbf{0}'$ is hyperimmune ([MM68, Theorem 1.2]). However this argument doesn't apply to the degrees that are not recursive in $\mathbf{0}'$. Miller and Martin ([MM68, Section 3]) then asked whether there is a hyperimmune minimal degree which is not recursive in $\mathbf{0}'$.

Our second motivation is the finite maximal chain problem for $\overline{\mathbf{GL_2}}$ degrees, i.e., those degrees $\mathbf{a}$ such that $\mathbf{a}'' > (\mathbf{a} \vee \mathbf{0}')'$. A degree $\mathbf{a}$ has the *finite maximal chain property* if there is a chain $\mathbf{0} = \mathbf{a}_0 < \mathbf{a}_1 < \cdots < \mathbf{a}_n = \mathbf{a}$ where each $\mathbf{a}_{i+1}$ is a minimal cover of $\mathbf{a}_i$. Lerman ([Ler83, Section IV.3]) asked whether there is a $\overline{\mathbf{GL_2}}$ degree with the finite maximal chain property. Note that $\mathbf{0}$ is $\mathbf{GL_2}$ (i.e., not $\overline{\mathbf{GL_2}}$). Moreover, it is a classical result that all minimal degrees are $\mathbf{GL_2}$ ([JP78, Theorem 1]), and so in order to find a $\overline{\mathbf{GL_2}}$ degree with the finite maximal chain property, we at least need a maximal chain of length 3, i.e., $\mathbf{0} < \mathbf{a} < \mathbf{b}$ where $\mathbf{a}$ is minimal and $\mathbf{b}$ is a minimal cover of $\mathbf{a}$.

Regardless of the length of such a maximal chain, we need to build a $\overline{\mathbf{GL_2}}$ degree $\mathbf{b}$ minimal over a $\mathbf{GL_2}$ degree $\mathbf{a}$. It is not difficult to prove (see Proposition V.5.1) that if $\mathbf{b}$ is a minimal cover of a $\mathbf{GL_2}$ degree $\mathbf{a}$, and if $\mathbf{b}$ is hyperimmune-free relativized to $\mathbf{a}$ or $\mathbf{b} \leq \mathbf{a}'$, then $\mathbf{b}$ is also $\mathbf{GL_2}$. That is to say, in order to find a $\overline{\mathbf{GL_2}}$ degree with the finite maximal chain property, we need a relativized version of "a hyperimmune minimal degree not recursive in $\mathbf{0}'$".

Cooper ([Coo73]) answered Miller and Martin's question using an indirect argument. He proved that any $\mathbf{d} \geq \mathbf{0}'$ is the jump of a minimal degree and showed that any minimal degree whose jump is $\mathbf{0}''$ is not below $\mathbf{0}'$. In addition, he used Jockusch's result that $\mathbf{a}' \geq \mathbf{0}''$ implies that $\mathbf{a}$ is hyperimmune ([Joc69]). Therefore any minimal degree whose jump is $\mathbf{0}''$ is hyperimmune and not below $\mathbf{0}'$.

The technical difficulty with producing a direct proof is that in the standard minimal degree construction we don't have a method which explicitly forces the minimal degree to be hyperimmune. In this chapter we will provide a new minimal degree construction and use it to directly construct a hyperimmune minimal degree. In addition, our construction can be easily augmented, given any degree $\mathbf{d}$, to construct a hyperimmune minimal degree which is not recursive in $\mathbf{d}$. Note that this can be done by Cooper's argument: one can find a minimal degree whose jump is $\mathbf{d}''$, and apply Jockusch's result as above to show that it is hyperimmune; in addition, such a minimal degree cannot be recursive in $\mathbf{d}$ because otherwise its jump would be below $\mathbf{d}'$.

The third motivation comes from some recent research on array nonrecursive (**ANR**) degrees. They share a lot of nice properties with $\overline{\mathbf{GL_2}}$ degrees (see [DJS96] and [CSh12]). So it is also natural to ask whether there is an **ANR** degree with the finite maximal chain property.

In addition, the following theorems about **ANR** degrees are quite interesting (in the sense of Question V.1.1):

Theorem V.1.3 (Downey, Jockusch, Stob [DJS96, Theorem 2.1])**.** *No* **ANR** *degree is minimal.*

Theorem V.1.4 (Downey, Jockusch, Stob [DJS96, Theorem 3.4])**.** *No* **ANR** *degree has a strong minimal cover.*

In some sense, **ANR** degrees are "high" in the Turing degrees and these degrees with strong minimal covers must be "lower" than **ANR** ones. Minimal degrees are of course the lowest possible nonrecursive degrees. From this point of view, Question V.1.1 is asking whether these "lowest" degrees have strong minimal covers.

Similarly to Question V.1.1, one might also ask whether every n-minimal degree (degrees that are "finitely many steps" away from $\mathbf{0}$) has a strong minimal cover. The answer is no and actually we will construct an **ANR** degree which is 2-minimal. This result gives an **ANR** degree with the finite chain property and together with Theorem V.1.4 implies the following corollary:

Corollary V.1.5. *There is a* 2*-minimal degree that does not have a strong minimal cover.*

This also shows that the class of degrees which are "low for **ANR**" is a proper subclass of array recursive degrees: a degree $\mathbf{a}$ is *low for* **ANR** if the **ANR** degrees above it are exactly the degrees **ANR** relative to $\mathbf{a}$ (see definition of the relativization in Chapter II). One can similarly define low for hyperimmune-free or low for **PA**, etc. It is easy to see that the degrees that are low for hyperimmune-free are exactly the hyperimmune-free ones. However, our example shows that that this is not true for the **ANR** case: such a 2-minimal degree is a **ANR** degree which is array recursive in an array recursive 1-minimal degree.

To guarantee hyperimmunity or array nonrecursiveness, we need to code in sufficiently large numbers in a minimal degree or a minimal cover construction, so we work with approximations in $\omega^{<\omega}$ rather than $2^{<\omega}$. We use *Even* to denote the set of all strings of even length, i.e., a string is *even* if it is in *Even*. A node on T is an *even-node* if it is the image of an even string.

V.2 A Hyperimmune Minimal Degree

Theorem V.2.1 (Cooper)**.** *For any degree* $\mathbf{d}$*, there is a hyperimmune minimal degree which is not recursive in* $\mathbf{d}$*.*

Proof. We approximate an infinite string α in ω^ω by a sequence of (partial) recursive trees $\langle T_i \rangle$ all of which map from $2^{<\omega}$ to $\omega^{<\omega}$.

Definition V.2.2. A tree T is *special* if it satisfies the following *properties*:

1. $T(\emptyset)\downarrow$.
2. For any string $\sigma \in Even$, if $T(\sigma)\downarrow$, then neither $T(\sigma)$ nor $T(\sigma * 0)$ is a leaf.
3. For any σ, $T(\sigma * 0)\downarrow \Leftrightarrow T(\sigma * 1)\downarrow$.

It is worth noting that although on a special tree some nodes do not branch, the paths on a special tree still contain a copy of the Cantor space, and this allows us to carry out Spector's minimal degree construction in a slightly different way.

In this construction we will require all trees to be special. In order to code in some sufficiently large numbers we want all trees in the construction to have another property as well:

Definition V.2.3. A tree T is *recursively unbounded*, if there is no recursive function that dominates every path on T.

In particular, for any special tree T considered in the construction, we also provide a recursive function $f(n)$ such that $f(n)$ is even for every n and for every string σ of length $f(n)$, if $T(\sigma)\downarrow$, then $\varphi_n(|T(\sigma * 1)|)\downarrow$ if and only if $T(\sigma * 1 * 0)\downarrow$. In addition, if both converge, then for $i = 0, 1$, $T(\sigma * 1 * i) \supset T(\sigma * 1) * (\varphi_n(|T(\sigma * 1)|) + i + 1)$. We call such a function a *witness function* for T.

It is easy to see that if we have a witness function $f(n)$ for a special tree T, then T is recursively unbounded, and we can take an appropriate full subtree to satisfy one requirement for hyperimmunity. In our construction, a common example of such a function is $f(n) = 2n$.

The following lemma is very easy to prove but it is convenient to make it explicit.

Lemma V.2.4. *If* T *is a special tree and* σ *is an even string in the domain of* T*, then* $FS(T, \sigma)$*, the full subtree of* T *above* σ *defined by* $FS(T, \sigma)(\sigma') = T(\sigma * \sigma')$*, is also special.*

Proof. Immediate. □

V.2.1 Requirements

First of all, as in a standard minimal degree construction, we need the following requirements to guarantee that $\deg(\alpha)$ is minimal:

R_e either φ_e^α is not total, or it is recursive, or $\alpha \leq_T \varphi_e^\alpha$.

It is easy to see that hyperimmunity of a degree $\mathbf{a}$ is equivalent to having a function $f \leq_T \mathbf{a}$ which is not dominated everywhere by any recursive function, i.e., for every recursive function h, there is an x such that $f(x) > h(x)$. We will require α to be such a function.

P_e either φ_e is not total, or $\exists x(\alpha(x) > \varphi_e(x)\downarrow)$.

To make sure that $\alpha \not\leq_T \mathbf{d}$, we fix a set $D \in \mathbf{d}$ and the following requirements.

Q_e either φ_e^D is not total, or $\exists x(\alpha(x) \neq \varphi_e^D(x)\downarrow)$.

A tree T *forces* a requirement R_e, P_e or Q_e if the requirement holds for every path $\alpha \in [T]$. Next, we provide basic modules to find a subtree T' of any given special T with a witness function such that T' forces one requirement R_e, P_e or Q_e. It is then easy to construct a sequence of trees $\langle T_i \rangle$ which approximates a string α satisfying all the requirements.

V.2.2 Initial tree T_0

We begin our construction with a special tree T_0.

First let $T_0(\emptyset) = \emptyset$, $T_0(0) = 0$, $T_0(1) = 1$, $T_0(00) = 00$ and $T_0(01) = 01$. Compute $\varphi_0(1)$. If it does not converge then $T(1) = 1$ is a leaf; if it converges, let $T_0(10) = 1 * (\varphi_0(1) + 1)$ and $T_0(11) = 1 * (\varphi_0(1) + 2)$.

Inductively for any σ of length $2k$, if we have defined $T_0(\sigma) = \tau$, then we let $T_0(\sigma * 0) = \tau * 0$, $T_0(\sigma * 1) = \tau * 1$, $T_0(\sigma * 00) = \tau * 00$ and $T_0(\sigma * 01) = \tau * 01$. Next we compute $\varphi_k(2k+1)$. If it does not converge, $\tau * 1$ is then a leaf; if it converges, let $T_0(\sigma * 10) = \tau * 1 * (\varphi_k(2k+1) + 1)$ and $T_0(\sigma * 11) = \tau * 1 * (\varphi_k(2k+1) + 2)$.

Note that T_0 is a special tree and $f(n) = 2n$ is a witness function for it.

V.2.3 Force α to be hyperimmune

Suppose we are given a special tree T with a witness function $f(n)$ and we want to force P_e. If φ_e is not total then we are done. If it is total, then we need a subtree T' of T which forces α to be not dominated everywhere by φ_e, i.e., $\alpha(x) > \varphi_e(x)$ for some x.

First find any σ of length $f(e)$ in the domain of T. By the definition of a witness function we know that $\varphi_e(|T(\sigma * 1)|)\downarrow$ if and only if $T(\sigma * 1 * 0)\downarrow$. By assumption $\varphi_e(|T(\sigma * 1)|)$ converges, so $T(\sigma * 1 * i) \supset T(\sigma * 1) * (\varphi_n(|T(\sigma * 1)|) + i + 1)$. We can now take the full subtree of T above $\sigma * 1 * 0$ as T'. By Lemma V.2.4 it is also a special tree. By the Padding Lemma we can recursively find a witness function f' for T' from f.

V.2.4 Force $\alpha \not\leq_T \mathbf{d}$

Given a special tree T and an index e, we want to find a subtree T' of T which forces $\alpha \neq \varphi_e^D$, if φ_e^D is total.

Pick any two different even-nodes τ_0 and τ_1 on T. One of them must be incompatible with φ_e^D if it is total. Without loss of generality we can assume that τ_0 is incompatible with φ_e^D, i.e., $\exists x(\tau_0(x)\downarrow \neq \varphi_e^D(x)\downarrow)$; then we take $T' = FS(T, \sigma_0)$ and again by Lemma V.2.4 and the Padding Lemma we are done.

V.2.5 Force α to be minimal

Now given a special tree T, its witness function f and an index e, we need to find a subtree of T to force R_e, i.e., either φ_e^α is not total, or it is recursive, or $\alpha \leq_T \varphi_e^\alpha$.

Note that Posner's Lemma is usually used to show that the nonrecursiveness of α is guaranteed by a splitting tree construction. Here this is automatically guaranteed by the requirements, because hyperimmune degrees are not recursive.

As in a standard minimal degree construction, we first try to ask whether there are e-splitting pairs above every node, i.e., whether we can construct a splitting subtree of T. However, we need to be careful here. Some nodes on T are leaves, and these nodes cannot be extended on T. For example, we start from the root of T and search for two nodes above it which form an e-splitting pair. If the two nodes we find are leaves, then we cannot extend them at the next step.

A possible way to solve this problem is to ask a similar but more specific question. First let $E = Even \cap dom(T)$, i.e., the collection of even strings in the domain of T, then ask whether the following is true:

$$\exists \sigma \in E\ \forall \sigma_0, \sigma_1 \supset \sigma, \sigma_0, \sigma_1 \in E[\neg(T(\sigma_0)|_e T(\sigma_1))].$$

Intuitively, property 2 of special trees guarantees that $T(\sigma_0)$ and $T(\sigma_1)$ are not terminal nodes, so we can continue to construct our subtree, if the answer is "no".

Now if the answer is "yes", then we can find an even σ in the domain of T such that for any even σ_0, σ_1 above σ, $T(\sigma_0)$ and $T(\sigma_1)$ do not form an e-splitting. For any α an infinite path on $FS(T, \sigma)$, if φ_e^α is total, then we can compute it as follows: Given x, we search above σ for the first even σ' such that $\varphi_e^{T(\sigma')}(x)\downarrow$ and output that value. We can always find one because $\varphi_e^\alpha(x)\downarrow$. We cannot find an answer different from $\varphi_e^\alpha(x)$ because of the positive answer to our question.

If the answer is "no", then we know that for any even σ in the domain, we can find two even strings σ_0, σ_1 such that $T(\sigma_0)$ and $T(\sigma_1)$ form an e-splitting. So we can easily construct a splitting tree. In addition, we want the splitting tree to be recursively unbounded. We guarantee this by arranging for $f'(n) = 2n$ to be a witness function as in the following construction:

First put $T'(\emptyset) = T(\emptyset)$. By the negative answer to our question we can find two even strings σ_0, σ_1 (above $\emptyset$) such that $T(\sigma_0)$ and $T(\sigma_1)$ e-split. Now find t such that $\varphi_t = \varphi_0$ and $f(t)$ bounds $|\sigma_0|$ and $|\sigma_1|$. Extend σ_0 and σ_1 respectively to σ_0' and σ_1' with $|\sigma_0'| = |\sigma_1'| = f(t)$. Let $T'(0) = T(\sigma_0' * 0)$ and $T'(1) = T(\sigma_1' * 1)$.

On the 0 side, search above $\sigma_0' {*} 0$ for even strings $\sigma_{00} \supset \sigma_0' {*} 0 {*} 0$ and $\sigma_{01} \supset \sigma_0' {*} 0 {*} 1$ such that $T(\sigma_{00})|_e T(\sigma_{01})$ and let $T'(0i) = T(\sigma_{0i})$ for $i = 0, 1$. (We can find such two even strings because of our convention that $\varphi_e^\sigma(x)\downarrow$ only if for all $y < x$, $\varphi_e^\sigma(y)\downarrow$.)

On the 1 side, we wait for $\varphi_0(|T(\sigma_1' * 1)|) = \varphi_t(|T(\sigma_1' * 1)|)$ to converge. If it doesn't, $T'(1)$ is a leaf. If it converges, then we know that $T(\sigma_1' * 1 * i) \supset$

$T(\sigma_1' * 1) * (\varphi_0(|T(\sigma_1' * 1)|) + i + 1)$. Note that $\sigma_1' * 1 * 0$ and $\sigma_1' * 1 * 1$ are even, so we can search above them for even strings $\sigma_{10} \supset \sigma_1' * 1 * 0$, $\sigma_{11} \supset \sigma_1' * 1 * 1$ such that $T(\sigma_{10})|_e T(\sigma_{11})$. Then let $T'(1i) = T(\sigma_{1i})$ for $i = 0, 1$.

Inductively, suppose we have defined $T'(\sigma) = T(\sigma^*)$ for $|\sigma| = 2n$, and σ^* is even. We find two even strings σ_0, σ_1 extending σ^* such that $T(\sigma_0)|_e T(\sigma_1)$, and pick t such that $\varphi_t = \varphi_n$ and $f(t)$ bounds the lengths of σ_0 and σ_1. Then we extend σ_0 and σ_1 respectively to σ_0' and σ_1' with $|\sigma_0'| = |\sigma_1'| = f(t)$. Now we define $T'(\sigma * i) = T(\sigma_i' * i)$ for $i = 0, 1$. The rest of the construction here is similar to the one in the base case. It is easy to see that T' is special and recursively unbounded witnessed by $f'(n) = 2n$. This finishes the construction and the proof. $\square$

V.3 Tree Systems

Next we construct an **ANR** 2-minimal degree. We use a tree construction to find a string α of minimal degree. At the same time, we do another tree argument relativized to α to find a string β which is minimal over α.

Usually, such iterations are implemented with uniform trees (see [Ler83, Chapter VI]). Here we provide a different and more general approach.

The intuition is to build a "tree of trees" where "tree" refers to the construction of α and "trees" refers to the construction of β.

A *tree system* is a pair of functions (T, S) where T is a partial recursive tree from $\omega^{<\omega}$ to $\omega^{<\omega}$ and S is a recursive function defined on the range of T with values finite trees mapping from a subset of $2^{<\omega}$ to $\omega^{<\omega}$, such that if $\tau \subset \tau'$ are both in the range of T, then $S(\tau')$ is an extension of $S(\tau)$, i.e., $S(\tau')$ restricted to the domain of $S(\tau)$ is equal to $S(\tau)$.

In this setting T is called a tree and S is called a system. We use σ to denote strings in the domain of T, τ (and less frequently π) to denote strings in the range of T and the domain of S. We use R to denote finite trees in the range of S. We use μ to denote strings in the domain of such an R and ρ (and less frequently η, ξ or ζ) to denote strings in the range of such an R.

We write $lb(\rho)$ to denote the last bit of ρ, i.e. $\rho(|\rho| - 1)$.

In our proof we will construct tree systems with the following *Properties*:

1. For any σ, if $T(\sigma)\downarrow$ then for any $i \le |T(\sigma)| + 1$, $T(\sigma * i) \downarrow \supset T(\sigma) * i$, and $T(\sigma * i) \uparrow$ for $i > |T(\sigma)| + 1$.
2. For any node τ on T, the maximum length of leaves on $S(\tau)$ is $3|\tau| + 1$.
3. For any node τ on T, all leaves of $S(\tau)$ are of length $3i + 1$ for some i and all leaves are images of odd strings. We call these leaves of length $3|\tau| + 1$ the *top leaves* on $S(\tau)$, and the other leaves on $S(\tau)$ the *terminal leaves*. In addition, if $S(\tau)(\mu) = \rho$ is a terminal leaf, then $lb(\mu) = lb(\rho) = 1$.
4. For any infinite path $\alpha \in [T]$, $S(\alpha) = \cup_{\tau \subset \alpha} S(\tau)$ is a special tree (see Definition V.2.2).
5. For any $\tau = T(\sigma)$ and any even-node ρ on $S(\tau)$, there are $\eta_0, \eta_1 \in S(\tau)$ extending ρ which are top leaves of $S(\tau)$ with $lb(\eta_0) = 0$ and $lb(\eta_1) = 1$.

We call such tree systems *special.*

Property 1 assures that we can infinitely often code some information into α. Properties 2 and 3 provide a convenient specific framework. Property 4 is crucial: it allows us to "guess" whether something happens or not as in our hyperimmune minimal degree construction. Property 5 would follow from our construction but it is more convenient to make it explicit.

Before starting our main construction, we provide some technical lemmas about special tree systems.

Lemma V.3.1. *Given a special tree system (T, S), if T' is a subtree of T which also satisfies Property 1, and S' is the restriction of S to the range of T', then (T', S') is also a special tree system.*

Proof. Immediate. □

We give an analog of the "full subtree" in the usual tree construction.

Definition V.3.2. Given a tree system (T, S), $\tau = T(\sigma)$ and $\rho = S(\tau)(\mu)$, the *full subtree system* (T', S') of (T, S) above (σ, μ), is defined as follows: Let T' be the usual full subtree of T above σ. For any π in the range of T', define $S'(\pi)$ to be the full subtree of $S(\pi)$ above μ. We denote this T' by $FSTS(T, S, \sigma, \mu)$.

Intuitively, this is a "full subtree of full subtrees". Now we show that with one easy requirement, this full subtree system is special if the original tree system is special.

Lemma V.3.3. *Suppose (T, S) is a special tree system, $\tau = T(\sigma)$ and μ is an even string in the domain of $S(\tau)$, then $FSTS(T, S, \sigma, \mu)$ is also a special tree system.*

Proof. Properties 1, 3 and 5 are immediate. Property 4 follows from Lemma V.2.4 and finally Property 2 follows from Properties 3 and 4. □

V.4 An ANR 2-minimal Degree

Theorem V.4.1. *There is a 2-minimal* **ANR** *degree.*

Proof. We construct a sequence of special tree systems (T_i, S_i) with each one a *subtree system* of the previous one, i.e., T_{n+1} is a subtree of T_n and for any τ in the range of T_{n+1}, $S_{n+1}(\tau)$ is a subtree of $S_n(\tau)$.

In the end, the T_i's will approximate an infinite string α. That is to say, α is the only string in all $[T_i]$. In addition, $S_i(\alpha)$ will approximate β in the same way. Our plan is to make $\deg(\beta)$ an **ANR** degree minimal over the minimal degree $\deg(\alpha)$.

We take $\lambda(n, s)$ to be the standard approximation of the modulus function of K with the number of changes in the column $\{\lambda(n, s)\}_{s\in\omega}$ bounded by $n + 1$.

Notice that in our construction of a hyperimmune minimal degree we could replace the partial recursive φ_n used at level $f(n)$ by any partial recursive function specified uniformly in n. Relativizing this idea, when we construct a minimal cover β of α, we can code in some partial α-recursive function infinitely often.

Now we construct an α which is bounded by $f(n) = n+1$ and each value $\alpha(n)$ will be regarded as a guess at the true number of changes in the corresponding column $\{\lambda(n,s)\}_{s\in\omega}$. If we assume that infinitely often we can code the true number of changes into α, then α computes a partial recursive function $h(n)$ which infinitely often equals the modulus function of K: for each n, we first read $\alpha(n)$, and then run through the column $\{\lambda(n,s)\}_{s\in\omega}$ looking for $\alpha(n)$ many changes. If we can find such a position we output the value of $\lambda(n,s)$ at that place, and if we cannot, the computation diverges.

If β can infinitely often exceed the values of $h(n)$ where it is equal to $m_K(n)$, then $\deg(\beta)$ is **ANR**. One thing to worry about here is how to make sure that those places where β exceeds h are the positions where h coincides with the modulus function of K. The main difficulty in the proof is to find a framework within which we can carry out this idea.

In our construction we place one more requirement on our tree systems:

6. For any τ on T and ρ a terminal leaf of length $3i+1$ on $S(\tau)$, we have $lb(\rho) = 1$ and $|\{s : s < |\tau|, \lambda(i,s) \neq \lambda(i,s+1)\}| < \tau(i)$. This implies that, for any $\tau' \supset \tau$ on T such that ρ is no longer a leaf on $S(\tau')$, $|\{s : s < |\tau'|, \lambda(i,s) \neq \lambda(i,s+1)\}| \geq \tau(i)$.

Note that Lemma V.3.1 and V.3.3 are still true with Property 6 added.

Again note that α being nonrecursive and β being strictly above α are both automatic: β is **ANR** while α is not **ANR** by Theorem V.1.3, so $\deg(\beta)$ is not below $\deg(\alpha)$. In addition, $\deg(\alpha)$ is strictly above $\mathbf{0}$ because otherwise $\deg(\beta)$ would be an **ANR** minimal degree, contradicting Theorem V.1.3.

V.4.1 Requirements

First of all, we want α to be minimal.

R_e either φ_e^α is not total, or it is recursive, or $\alpha \leq_T \varphi_e^\alpha$.

In addition, we require β to be minimal over α.

P_e either φ_e^β is not total, or it is recursive in α, or $\beta \leq_T \varphi_e^\beta \oplus \alpha$.

We guarantee array nonrecursiveness by the following:

Q_n there exists at least n distinct x's such that $\beta(3x+2) > m_K(x)$.

Then β computes a function which is not dominated by m_K. Therefore it is **ANR** by definition.

The definition of forcing here is similar to that in section V.2: we say (T, S) *forces* a requirement if for every $\alpha \in [T]$ and every $\beta \in [S(\alpha)]$, the requirement holds for α and β. Finally $\alpha \leq_T \beta$ is guaranteed by the initial tree system defined below.

V.4.2 Initial tree system (T_0, S_0)

We define an initial tree system (T_0, S_0) in this subsection. For simplicity we write (T, S) instead of (T_0, S_0).

T is rather simple. It is the restriction of the identity function to a recursive domain defined as follows:

$\emptyset$ is in the domain; if σ is in the domain then $\sigma * 0, \sigma * 1, \ldots, \sigma * (|\sigma| + 1)$ are exactly the immediate successors of σ in the domain.

Given τ of length n in the range of T (and so in the domain of S), we define $S(\tau) = R$ as a finite tree as follows:

The domain of R is a subset of $2^{2|\tau|+1}$. First let $R(\emptyset) = \emptyset$. Inductively, for any μ of length $\leq 2|\tau|$ and $R(\mu) = \rho$, if μ is even, then put $R(\mu * 0) = \rho * 0$ and $R(\mu * 1) = \rho * 1$.

If $|\mu| = 2k + 1$ and $lb(\mu) = 0$, we put $R(\mu * 0) = \rho * \tau(k) * 0$ and $R(\mu * 1) = \rho * \tau(k) * 1$. If $lb(\mu) = 1$, then we search for the $\tau(k)$-th change in the column $\{\lambda(k, s)\}_{s \in \omega}$ up to $s = |\tau|$. If we cannot find that many changes, then both $R(\mu * 0)$ and $R(\mu * 1)$ diverge; if we can find such, then let x be the value of $\lambda(k, t)$ where t is the place we see the $\tau(k)$-th change, and let $R(\mu * 0) = \rho * \tau(k) * (x + 1)$ and $R(\mu * 1) = \rho * \tau(k) * (x + 2)$.

It is easy to see that R is a finite tree and the maximum length of its leaves is $3|\tau| + 1$. For every $\alpha \in [T]$ and every $\beta \in [S(\alpha)]$, it is immediate that $\forall x(\alpha(x) = \beta(3x + 1))$; hence $\alpha \leq_T \beta$.

Intuitively, along any path on the tree $S(\tau)$, position $3i + 1$ codes the (possible) number of changes, position $3i$ guesses whether it can be found (1) or not (0), and position $3i + 2$ codes a sufficiently large number, if position $3i$ guesses that such number of changes happens and it is actually found.

Infinitely often Property 1 can be used to find a full subtree of T (with the induced "subsystem" of S) to code the "correct number of changes along a column" into α. So in β we can code in a number greater than the corresponding value of the modulus function of K.

V.4.3 Force α to be minimal

Suppose we are given (T, S) and an index e, and we need to find a subtree system (T', S') of (T, S) to force R_e.

By Lemma V.3.1 we only need to find such a subtree T' of T which satisfies Property 1 and take S' to be the corresponding restriction.

As usual we ask whether the following holds (for simplicity we always let D be the domain of T):

$$\exists \sigma \in D \forall \sigma_0, \sigma_1 \supset \sigma, \sigma_0, \sigma_1 \in D(\neg(T(\sigma_0)|_e T(\sigma_1))).$$

If the answer is "yes" then we take $T' = FS(T, \sigma)$. It is routine to argue that φ_e^α is either not total or recursive for every $\alpha \in [T']$.

If the answer is "no", then we do the following construction. Start with $T'(\emptyset) = T(\emptyset) = \tau$. Now, in order to satisfy Property 1, we need to find $|\tau| + 2$ pairwise

e-splitting nodes on T, each extending one of $T(i) \supset \tau * i$ for $i = 0, 1, \ldots, |\tau| + 1$ respectively. In the inductive step we need a similar argument, i.e., once we have defined $T'(\sigma) = \tau = T(\sigma')$, then we need to find $|\tau| + 2$ pairwise e-splitting nodes on T, each extending one of $T(\sigma' * i) \supset \tau * i$ for $i = 0, 1, \ldots, |\tau| + 1$ respectively. We start with these nodes $T(\sigma' * i)$. Each time we pick one pair of them, and using the splitting property we can extend these two strings on T to make them e-split. We can iterate this process for each pair. Finally we end up with pairwise e-splitting extensions and let them be $T'(\sigma * i)$ for $i = 0, 1, \ldots, |\tau| + 1$.

This construction gives us an e-splitting subtree T' of T and it is easy to argue that $\alpha \leq_T \varphi_e^\alpha$.

V.4.4 Force β to be ANR

Given (T, S) we want to code a sufficiently large number into β, i.e., we need to find a new n such that $\beta(3n + 2) \geq m_K(n)$.

By Property 5 we know that there exist $\tau = T(\sigma)$ and $\rho = S(\tau)(\mu)$ a top leaf of $S(\tau)$ with $lb(\rho) = 1$. Then, by Property 1, we know that $\tau_i = T(\sigma * i) \supset \tau * i$, $i = 0, 1, \ldots, |\tau| + 1$ are exactly the successor nodes of τ. Intuitively, they are guessing that the $|\tau|$-th column of λ has i many changes, respectively for each i.

Then we take i to be the actual number of changes through that column, i.e., we let $i = |\{s : \lambda(|\tau|, s) \neq \lambda(|\tau|, s + 1)\}|$. That is to say, $T(\sigma * i) = \tau_i \supset \tau * i$ has the correct guess at the number of changes in this column. Also let t be the position in that column where we see the i-th change, i.e. t is the least such that $\lambda(|\tau|, t') = \lambda(|\tau|, t)$ for all $t' > t$.

Note that by Property 3, $\mu * 0$ is an even string. By the construction of the initial tree and Property 6, we know that for all $T(\sigma^*) = \tau^* \supset \tau_i$ with length $\geq t$, $S(\tau^*)(\mu * 0) \supset \rho * i * |m_K(|\tau|) + 1|$ or $\rho * i * |m_K(|\tau|) + 2|$. Pick any such σ^* and take $(T', S') = FSTS(T, S, \sigma^*, \mu * 0)$. By Lemma V.3.3, this tree system is special and it is easy to see that it forces β to have one more position n where $\beta(3n + 2)$ is greater than $m_K(n)$.

V.4.5 Force β to be minimal over α

In this subsection we will present the main splitting construction. Suppose we are given a special tree system (T, S) and an index e. We need to find a special subtree system (T', S') which forces P_e, i.e., either φ_e^β is not total, or it is recursive in α, or $\beta \leq_T \varphi_e^\beta \oplus \alpha$.

Notice that in the requirement above, if we could force $\beta \leq_T \varphi_e^\beta$, then we would make β a strong minimal cover over α, but this cannot happen because no **ANR** degree can be a strong minimal cover (for example, by [DJS96, Theorem 2.5]).

Now we ask the following key question (note that D is the domain of T and we let $E(\tau)$ be the set of all even strings in the domain of $S(\tau)$):

$$\begin{aligned}&\exists \sigma \in D \exists \mu \in E(T(\sigma))\\ &\forall \sigma' \supset \sigma, \sigma' \in D \forall \mu_0, \mu_1 \supset \mu, \mu_0, \mu_1 \in E(T(\sigma'))[\neg(S(T(\sigma'))(\mu_0)|_e S(T(\sigma'))(\mu_1))].\end{aligned}$$

That is, we ask whether there is a node $\tau = T(\sigma)$ and an even-node $\rho = S(\tau)(\mu)$ such that for any $\tau' = T(\sigma')$ extending τ on T, there is no e-splitting pair of even-nodes on $S(\tau')$ extending ρ.

If the answer is "yes", then we pick a witness pair (σ, μ), take $(T', S') = FSTS(T, S, \sigma, \mu)$ and claim that if φ_e^β is total, then it is recursive in α. To compute $\varphi_e^\beta(x)$, we simply search on the tree $S'(\alpha)$ for an even-node η which makes $\varphi_e^\eta(x)$ converge. We must find an answer because $\beta \in [S(\alpha)]$ and φ_e^β is total. We cannot find an answer different from $\varphi_e^\beta(x)$ by the positive answer to our key question.

Now if the answer is "no", then we know that

$$(\dagger) : \forall \sigma \in D \forall \mu \in E(T(\sigma))$$
$$\exists \sigma' \supset \sigma, \sigma' \in D \exists \mu_0, \mu_1 \supset \mu, \mu_0, \mu_1 \in E(T(\sigma'))[S(T(\sigma'))(\mu_0)|_e S(T(\sigma'))(\mu_1)].$$

We will use this property to construct a "splitting" subtree system (T', S'). Here by "splitting" we mean the following:

$$(*) : \text{ For all } \tau \text{ on } T', \text{ all leaves of } S'(\tau) \text{ pairwise } e\text{-split.}$$

First we argue that with property $(*)$ the requirement is satisfied. From α one can compute $S'(\alpha)$ and β is a path on $S'(\alpha)$. $(*)$ guarantees e-splitting so φ_e^β can pick a unique leaf which is an initial segment of β, on $S'(\tau)$ for every $\tau \subset \alpha$. This proves that $\beta \leq_T \varphi_e^\beta \oplus \alpha$.

To finish the proof we will provide such a splitting subtree system construction, i.e., we construct a subtree system (T', S') which is special and has property $(*)$.

To find $T'(\emptyset)$, first by $(\dagger)$ we search above $\emptyset$ for a σ such that there exist two e-splitting even-nodes ρ_0 and ρ_1 on $S(T(\sigma))$. Then by Property 5 we know that on the tree $S(T(\sigma))$, ρ_0 and ρ_1 have extensions η_0 and η_1 respectively such that $lb(\eta_0) = 0$, $lb(\eta_1) = 1$ and both are top leaves of $S(T(\sigma))$. Then define $T'(\emptyset) = T(\sigma)$ and $S'(T'(\emptyset))$ to be the subtree of $S(T(\sigma))$ defined by $S'(T'(\emptyset))(\emptyset) = S(T(\sigma))(\emptyset)$, $S'(T'(\emptyset))(0) = \eta_0$ and $S'(T'(\emptyset))(1) = \eta_1$.

Now suppose we have defined $T'(\sigma') = \tau = T(\sigma)$ and $S'(\tau)$ a subtree of $S(\tau)$. We need to define $T'(\sigma' * i) \supset T(\sigma * i)$ for each $i \in \{0, 1, \ldots, |\tau| + 1\}$ and define the corresponding $S'(T'(\sigma' * i))$ with pairwise splitting leaves. The good thing is that we only need e-splitting for leaves on each $S'(T'(\sigma' * i))$ separately but not for all leaves on every tree altogether. As we know the number i from α in the end, we only need to find e-splitting extensions separately for each i.

Suppose $\rho_0, \rho_1, \ldots, \rho_n$ are all leaves on $S'(\tau)$. Note that by induction, they pairwise e-split. Let $X = \{\rho_0, \rho_1, \ldots, \rho_k\}$ be the set of all top leaves and $Y = \{\rho_{k+1}, \rho_{k+2}, \ldots, \rho_n\}$ be the set of all terminal leaves on $S'(\tau)$. Some of the ρ_j's in X are terminal leaves on $S(T(\sigma * i))$ and we can put them aside (into Y) at this time. Now let $Z = \{\rho_0, \rho_1, \ldots, \rho_l\}$ be the set of all leaves on $S'(\tau)$ that are not terminal leaves on $S(T(\sigma * i))$.

The aim of the construction is to find $\tau_i \supset T(\sigma * i) \supset \tau * i$ on T such that for each $\rho_j \in Z$, $\rho_j = S'(\tau)(\mu_j)$ (note that $|\mu_j|$ is always odd by Property 3), one can find on $S(\tau_i)$ six appropriate nodes defined to be $S'(\tau_i)(\mu_j * 0)$, $S'(\tau_i)(\mu_j * 1)$, $S'(\tau_i)(\mu_j * 00)$, $S'(\tau_i)(\mu_j * 01)$, $S'(\tau_i)(\mu_j * 10)$ and $S'(\tau_i)(\mu_j * 11)$. Moreover, the latter four are top leaves of $S(\tau_i)$ and they have the desired e-splitting property.

For each $\rho_j \in Z$, $\rho_j = S'(\tau)(\mu_j) = S(T(\sigma))(\mu'_j)$ we know that $\rho'_j = S(T(\sigma * i))(\mu'_j * 0)$ is an even-node on $S(T(\sigma * i))$, hence also an even-node on $S(\pi)$ for any $\pi \supset T(\sigma * i)$.

Now start from ρ'_0. By (†) we know that there is a $\pi_0 \supset T(\sigma * i)$ and two even-nodes ξ_0^1, ξ_0^1 on $S(\pi_0)$ extending ρ'_0 that e-split.

Using the same idea, we apply (†) twice to get $\pi_2 \supset \pi_1 \supset \pi_0$ such that both ξ_0^0 and ξ_0^1 have two even-node extensions ξ_0^{00}, ξ_0^{01} and ξ_0^{10}, ξ_0^{11}, respectively.

Now on $S(\pi_2)$, some nodes in Y will become nonterminal and we need to move them into Z.

Pick the next node in Z, say ρ_1. ρ'_1 defined above is also an even-node in $S(\pi_2)$ and similarly by applying (†) three times one can get $\pi_5 \supset \pi_4 \supset \pi_3 \supset \pi_2$ and even-nodes ξ_1^0, ξ_1^1, ξ_1^{00}, ξ_1^{01} and ξ_1^{10} and ξ_1^{11} extending ρ'_1 (to be explicit, the subscript 1 refers to ρ'_1 and superscripts $0, 1, 00, 01, 10, 11$ refer to their positions in the relative 2-level tree structure extending ρ'_1).

Again some nodes in Y will appear nonterminal at this level and we need to move them into Z. There are only finitely many leaves on $S'(\tau)$ at the beginning. So we can iterate this process for all $\rho_j \in Z$ and finally end up with $\pi = \pi_{3t-1}$ for some t. On $S(\pi)$ each ρ'_j similarly has six extensions with appropriate e-splitting properties, and these nodes in Y are still terminal leaves. Now define $T'(\sigma * i) = \pi$, and define $R = S'(\pi)$ extending $S'(\tau)$ as follows: for each $\rho_j \in Z$, $\rho_j = S'(\tau)(\mu_j)$, define $S'(\pi)(\mu_j * 0) = \xi_j^0$ and $S'(\pi)(\mu_j * 1) = \xi_j^1$.

By Property 5 we can find a top leaf ζ_j^{mn} on $S(\pi)$ extending ξ_j^{mn} $(m, n \in \{0, 1\})$ such that $lb(\zeta_j^{mn}) = n$. Then we define $S'(\pi)(\mu_j * m * n) = \zeta_j^{mn}$ for $m, n \in \{0, 1\}$.

This finishes the splitting subtree system construction and it is not difficult to see that the subtree system (T', S') we constructed is a special tree system. □

V.5 Appendix

Here we prove a proposition mentioned in the Introduction.

Proposition V.5.1. *Suppose* $\mathbf{a}$ *is* $\mathbf{GL_2}$ *and* $\mathbf{b}$ *is a minimal cover of* $\mathbf{a}$*, then* $\mathbf{b}$ *is also* $\mathbf{GL_2}$ *if either of the following holds:*

1. $\mathbf{b} < \mathbf{a}'$*, or*
2. $\mathbf{b}$ *is hyperimmune-free relative to* $\mathbf{a}$*, i.e., every function recursive in* $\mathbf{b}$ *is dominated by a function recursive in* $\mathbf{a}$*.*

Proof. In the first case, by a relativized version of [JP78, Corollary 1], we know that if $\mathbf{b} < \mathbf{a}'$ then $\mathbf{b}$ is $\mathbf{L_2}$ relativized to $\mathbf{a}$. That means $\mathbf{b}'' = \mathbf{a}'' = (\mathbf{a} \vee \mathbf{0}')' \le (\mathbf{b} \vee \mathbf{0}')'$. Therefore $\mathbf{b}$ is $\mathbf{GL_2}$.

In the second case, it is well known ([JP78, Lemma 1]) that $\mathbf{a}$ being $\mathbf{GL_2}$ is equivalent to there being a function g recursive in $\mathbf{a} \vee \mathbf{0}'$ which dominates every function recursive in $\mathbf{a}$. Then this function g also dominates every function recursive in $\mathbf{b}$ and of course $g \le_T \mathbf{a} \vee \mathbf{0}' \le \mathbf{b} \vee \mathbf{0}'$. So $\mathbf{b}$ is also $\mathbf{GL_2}$. □

CHAPTER VI
2-MINIMAL NON-$\mathbf{GL}_2$ DEGREE

This chapter will appear as a paper in the Journal of Mathematical Logic.

VI.1 Introduction

We investigate two important subjects in the research of Turing degrees: one is the generalized high/low hierarchy, the other is the study of minimal degrees and minimal covers.

As we mentioned in Chapter I, in recursion theory, there are different notions of how close a degree is to the recursive degree $\mathbf{0}$, or how close a degree is to $\mathbf{0}'$, the degree of the complete set K (the halting problem). For example, the high/low hierarchy (see definition in §I.1.2) is such a measurement in terms of the number of Turing jump operators needed to collapse the degree with either $\mathbf{0}$ or $\mathbf{0}'$. Roughly speaking, the low degrees are close to the recursive degree $\mathbf{0}$ (in the sense that any degree below a low one is also low), and the high ones are close to $\mathbf{0}'$. The number of iterations provides an approximation of how close a degree is to either $\mathbf{0}$ or $\mathbf{0}'$, for example, the low$_1$ degrees are regarded as being closer to $\mathbf{0}$ than low$_2$ ones.

The high/low hierarchy only classifies the degrees below $\mathbf{0}'$, and we can generalize this notion to all Turing degrees and define the generalized high/low hierarchy. A lot of nice properties of the high/low hierarchy transfer to properties of the corresponding generalized high/low hierarchy. In particular, it is also a common technique to prove a theorem for the high/low hierarchy first, and then use the method of the proof to show an analogous version for the generalized high/low hierarchy (see, for example, [ASDWY09]).

Another notion of such measurement is minimality and iterated minimality (see definition in §I.1.1). For example, 2-minimal degrees are those degrees that are "two steps" away from $\mathbf{0}$. This provides another notion of how close a degree is to the recursive degree, i.e., the number of minimality iterations needed to reach the degree from $\mathbf{0}$.

The lowness notion and the minimality notion are both trying to characterize how close the degree is to $\mathbf{0}$, so we would like to see if they have some strong connections. The first breakthrough is the following theorem:

Theorem VI.1.1 (Jockusch and Posner [JP78, Theorem 1])**.** *Every minimal degree is generalized low$_2$ ($\mathbf{GL}_2$). In fact, every degree which is not $\mathbf{GL}_2$ ($\overline{\mathbf{GL}_2}$) computes a 1-generic set.*

Lerman iterated this theorem below $\mathbf{0}'$ and showed that iterations of minimality below $\mathbf{0}'$ cannot reach higher than $\mathbf{L_2}$ in the high/low hierarchy, therefore the degrees below $\mathbf{0}'$ that are not low$_2$ ($\overline{\mathbf{L_2}}$) cannot be n-minimal for any $n \in \omega$. In another words, they do not have the *finite maximal chain property*: A degree $\mathbf{c}$ has the *finite maximal chain property* if there is a chain $\mathbf{0} = \mathbf{c}_0 < \mathbf{c}_1 < \cdots < \mathbf{c}_n = \mathbf{c}$ such that each $\mathbf{c}_{i+1}$ is a minimal cover of $\mathbf{c}_i$ (such a chain is also called a *maximal chain*).

Lerman then asked whether this result also holds for $\overline{\mathbf{GL}_2}$ degrees ([Ler83, IV.3.7]), i.e., whether there is a $\overline{\mathbf{GL}_2}$ degree with the finite maximal chain property.

The class of $\overline{\mathbf{GL}_2}$ degrees has been studied over the years and these degrees have a lot of nice properties (see for example, [JP78], [ASDWY09] and [CSh12], see also chapter [2]). In particular, Theorem VI.1.1 actually shows that the degree structure below any $\overline{\mathbf{GL}_2}$ degree is complicated, since it is a classical result that one can embed any countable partial order below a 1-generic degree. In contrast, finite maximal chains are in some sense the least complicated partial orders. So it seems unlikely that a $\overline{\mathbf{GL}_2}$ degree can be the top of a finite maximal chain. However, this intuition turns out to be false.

On the other hand, Theorem VI.1.1 is strict with respect to the generalized high/low hierarchy. More precisely, we have:

Theorem VI.1.2 (Sasso [Sas74])**.** *There is a minimal degree which is not* $\mathbf{GL}_1$*.*

In the language of [LS88], these two results (Theorems VI.1.1 and VI.1.2) completely categorized the invariant classes of non-minimal degrees with respect to the generalized high/low hierarchy. Results of this sort can be used to get some bound on the quantifier complexity of definitions of various degree classes. So in [LS88], it was asked whether we can answer a similar question for 2-minimal degrees ([LS88, Question 6.4]), as it is "the first roadblock" in classifying $\exists\forall$ sentences with respect to the generalized high/low hierarchy. So the first question to consider here is whether there is a 2-minimal degree which is "significantly higher" than 1-minimal degrees, i.e., whether a 2-minimal degree can be $\overline{\mathbf{GL}_2}$.

In this chapter, we show that actually there is a 2-minimal degree which is $\overline{\mathbf{GL}_2}$. This answers Lerman's question and also provides a first step towards an answer to Question 6.4 in [LS88].

In the end, we modify the construction to get a 2-minimal degree which is $\mathbf{GH}_2$.

VI.2 Preliminary Ideas

Lerman's question we mentioned in the previous section was also studied in [Cai10] (see also Section V.1 and Proposition V.5.1). In Proposition V.5.1 we pointed out some necessary conditions if one wants to build a degree $\mathbf{b}$ minimal over $\mathbf{a}$ where $\mathbf{a}$ is $\mathbf{GL}_2$ and $\mathbf{b}$ is $\overline{\mathbf{GL}_2}$, since it is easy to see that such construction is necessary to get a $\overline{\mathbf{GL}_2}$ degree with the finite maximal chain property. We also showed that there is an *array nonrecursive* 2-minimal degree by building a maximal chain $\mathbf{0} < \mathbf{a} < \mathbf{b}$ with $\mathbf{b}$ being array nonrecursive. Array nonrecursive degrees are defined to generalize $\overline{\mathbf{GL}_2}$ degrees. They share a lot of nice properties with $\overline{\mathbf{GL}_2}$ degrees (see [DJS96] and [CSh12]), and have also been used in various places in the study of Turing degrees (see for example, [Ish99] and [Sh07]). However, one can actually show that the array nonrecursive 2-minimal degree constructed there is $\mathbf{GL}_2$.

In the setting above, by a relativized version of Theorem VI.1.1, we know that $\mathbf{b}$ is $\mathbf{GL}_2$ relativized to $\mathbf{a}$. Now if $\mathbf{a}$ is $\mathbf{GL}_1$, then $\mathbf{b}'' = (\mathbf{b}\vee\mathbf{a}')' = (\mathbf{b}\vee\mathbf{a}\vee\mathbf{0}')' = (\mathbf{b}\vee\mathbf{0}')'$ and so $\mathbf{b}$ is $\mathbf{GL}_2$. That is to say, in order to make $\mathbf{b}$ be $\overline{\mathbf{GL}_2}$, one has to make $\mathbf{a}$ be $\overline{\mathbf{GL}_1}$, i.e., not $\mathbf{GL}_1$. So the technique used in Theorem VI.1.2 might be very important for constructing a $\overline{\mathbf{GL}_2}$ degree with the finite maximal chain property.

In our construction, the framework (tree systems) comes from [Cai10], and the coding idea is from a new construction of a hyperimmune minimal degree ([Cai10], see also Section V.2). We also use Sasso's idea of narrow trees (in the proof of Theorem VI.1.2, see the review in Section VI.4) for both the "**a** part" and the "**b** part" of the proof.

We provide basic definitions and some new notions in Section VI.3 and discuss the requirements together with the basic ideas of the construction in Section VI.4. We then work with only one single $\overline{\mathbf{GL}_2}$ type of requirement in Section VI.5 to give readers a local picture of the construction, and we prove the full theorem in Section VI.6.

VI.3 Basic Definitions and Notions

VI.3.1 Trees

In this chapter, we use *trees* as partial functions $T : 2^{<\omega} \to 2^{<\omega}$.

Recall that a tree is *finite* if its domain is finite. For two finite trees R_0, R_1, we say that R_1 is a *tree extension* of R_0 if the restriction of R_1 to the domain of R_0 is the same as R_0. Intuitively, R_1 preserves the tree structure of R_0 and possibly adds some extensions.

Given two trees T and S, we say that S is a *subtree* of T if every node on S is a node on T. If $\tau = T(\sigma)$ is a node on T, then the *full subtree* T' of T above τ is defined as $T'(\xi) = T(\sigma * \xi)$ for every $\xi \in 2^{<\omega}$. We use $\mathbf{FS}(T, \tau)$ to denote this full subtree.

Given a tree T, the *narrow subtree* T' of T is defined as $T'(\sigma) = T(\emptyset \oplus \sigma)$ for every $\sigma \in 2^{<\omega}$. We use $\mathbf{Nar}(T)$ to denote the narrow subtree of T.

VI.3.2 Tree systems

A tree system is intuitively "a tree of trees". Similar to the case of trees, we might have different definitions of tree systems in different situations. Here we also want to embed recursiveness into the definition for simplicity.

A *tree system* is a pair (T, S) where T is a total recursive tree and S is a recursive function from the range of T to (indices of) finite trees such that for every $\tau_0 \subset \tau_1$ both in the range of T, $S(\tau_1)$ is a tree extension of $S(\tau_0)$. In this setting, T is a tree and S is called a *system*. Any node on any $S(\tau)$ is also called a *node* on the tree system.

From now on, we reserve the use of certain lower case Greek letters for strings in different worlds: σ will only denote strings in the domain of a tree T; τ will only denote strings in the range of T and domain of S; μ will only denote strings in the domain of $R = S(\tau)$ and ρ (with other letters like η, ξ) will only denote strings in the range of such R. So with a given tree system (T, S), when we say "for every τ", we mean "for every string τ in the range of T" (i.e., "for every node τ on T"), etc.

A tree system (T', S') is a *subtree system* of (T, S) if T' is a subtree of T and for every τ in the range of T', $S'(\tau)$ is a subtree of $S(\tau)$. A sequence of tree systems $\langle (T_i, S_i) \rangle_{i \in \omega}$ is *nested* if every (T_{i+1}, S_{i+1}) is a subtree system of (T_i, S_i).

Given a tree system (T, S), a node τ on T and a node ρ on $S(\tau)$, the *full subtree system* of (T, S) above (τ, ρ) is a tree system (T', S') where T' is the full subtree of T above τ and for each τ' in the range of T', $S'(\tau')$ is the full subtree of $S(\tau')$ above ρ. We use $\mathbf{FSTS}(T, S, \tau, \rho)$ to denote this full subtree system. It is an analog of full subtrees and will be used in our construction in many places.

For every path $A \in [T]$, $S(A) = \cup_{\tau \subset A} S(\tau)$ is a tree recursive in A. We call a path on $S(A)$ a *path* of the tree system.

A tree system (T, S) is *e-splitting* if every $S(\tau)$ is an e-splitting tree. Similarly we have the following computation lemma, though in our construction we will use a slightly different version.

Lemma VI.3.1. *For every e-splitting tree system* (T, S)*, if* $A \in [T]$ *and* $B \in [S(A)]$*, then* $B \leq_T \varphi_e^B \oplus A$.

Proof. Relativize Lemma I.3.1 to A. □

VI.3.3 Blocks

In our construction, we regard each $R = S(\tau)$ in a tree system (T, S) as a structure consisting of *blocks*. We use the letter $\mathbb{B}$ to denote blocks. Each block $\mathbb{B}$ consists of five or seven nodes: $R(\mu)$, $R(\mu * 0)$, $R(\mu * 1)$, $R(\mu * 0 * 0)$, $R(\mu * 0 * 1)$, and if $R(\mu * 1)$ is not a leaf, also $R(\mu * 1 * 0)$, $R(\mu * 1 * 1)$. Here μ is always even. In this setting, we call $R(\mu)$ the *root* of the block $\mathbb{B}$, $R(\mu * 0)$ the 0-*node* of $\mathbb{B}$, $R(\mu * 1)$ the 1-*node* of $\mathbb{B}$, etc. (see Figure VI.1). The two or four third level nodes are also called the *top-nodes* of the block.

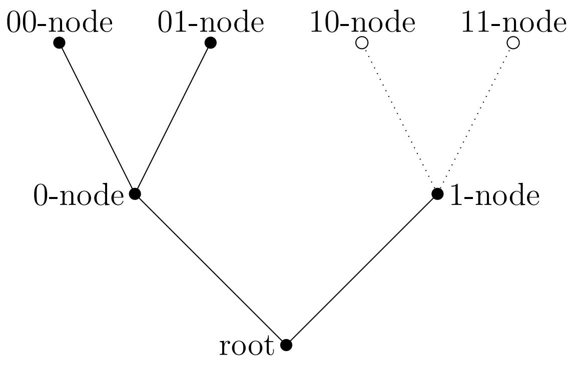

Figure VI.1: A block $\mathbb{B}$

A block is *full* if it has all four top-nodes. Note that a block which is not full might become full if we branch its 1-node in the construction later, in which case we say that we *fill* the block.

We connect the blocks in a tree $R = S(\tau)$ in a natural way: $\mathbb{B}_1$ immediately extends $\mathbb{B}_0$ if the root of $\mathbb{B}_1$ is a top-node of $\mathbb{B}_0$. The *level* of a block is defined naturally as the number of blocks below it.

A block is *e-splitting* if every pair of incompatible nodes in it e-splits.

In a block, the root is in some sense not important. It might be worth mentioning here that in our construction, sometimes we might change the root of a block to some node below it and keep every other node. In our construction, the crucial ingredients of a block are the 1-node and the 01-node. From this point of view, we are not changing the essence of a block if we only change its root.

VI.4 Review of Sasso's Proof and Basic Ideas on Narrow Trees

For a detailed version of Sasso's construction, see [Ler83, V.3.12].

In the construction of a minimal path A, if we want to make $A' >_T A \oplus 0'$, we simply need to satisfy $A' \neq \varphi_e^{A\oplus 0'}$ for every e. We first take a narrow subtree $\mathbf{Nar}(T)$ of the tree T with which we are currently working. Now "A is a path on $\mathbf{Nar}(T)$" is a $\Pi_1^0(A)$ statement, and so we can find an x such that A is a path on $\mathbf{Nar}(T)$ if and only if $x \notin A'$.

Then we ask whether we can force $\varphi_e^{A\oplus 0'}(x)$ to converge on $\mathbf{Nar}(T)$; if not, then the requirement is automatically satisfied by taking $\mathbf{Nar}(T)$ as the next tree in the construction. If so, say τ on $\mathbf{Nar}(T)$ makes $\varphi_e^{\tau\oplus 0'}(x) \downarrow= i$. If $i \neq 0, 1$, the requirement is also satisfied by taking $\mathbf{FS}(T, \tau)$. If $i = 1$, then the requirement is satisfied by taking $\mathbf{FS}(\mathbf{Nar}(T), \tau)$, since we have forced A to be on the narrow subtree. If $i = 0$, then we find an extension τ' of τ on T but not on $\mathbf{Nar}(T)$ (note that this is always possible by the definition of narrow trees), and by taking $\mathbf{FS}(T, \tau')$ we are also done, since we have forced A to be off the narrow subtree. In each case, we have forced $A'(x) \neq \varphi_e^{A\oplus 0'}(x)$, and so the final path A we construct is $\overline{\mathbf{GL}_1}$.

Now we want to build $A \leq_T B$ by a nested sequence of tree systems $\langle (T_i, S_i) \rangle$ in the sense that A is the only path on every T_i and B is the only path on every $S_i(A)$. We want to satisfy three requirements: A is minimal; B is minimal over A; and most importantly $B'' >_T (B \oplus 0')'$.

As we have already mentioned in the introduction, by relativizing Theorem VI.1.1, we know that if B is minimal over A, then $B'' \equiv_T (B \oplus A')'$. By the Limit Lemma, it suffices to satisfy the following requirements:

1. R_e: $\exists x \, \neg[(B \oplus A')'(x) = \lim_s \varphi_e^{B\oplus 0'}(x, s)]$.

Note that the above requirement says that there is an x such that either $\varphi_e^{B\oplus 0'}(x, s)$ is not defined for some s, or the whole sequence $\langle \varphi_e^{B\oplus 0'}(x, s) \rangle_{s\in\omega}$ does not have a limit, or the limit is not $(B \oplus A')'(x)$.

The left hand side of the equation will be controlled in a similar way as in Sasso's construction, except that here our "narrow subtree" also depends on A'. We will give a detailed definition of this new "narrow subtree" in §VI.5.2 and §VI.6.1.

For the right hand side, without loss of generality, we first assume that φ_e only takes values 0 or 1. Then we guess that the limit does not exist and take full subtree systems along our construction to force the value to change infinitely many times. Once we discover that we can no longer force the value to change, then we know that it is time to do some type of "narrow subtree construction" to satisfy R_e. The major difficulty is to find a plan such that one can implement either of the two types of subtree constructions (to keep B either on or off the "narrow subtree") at any subsequent stage after we define the "narrow subtree" for R_e.

The other two requirements are handled in more or less standard ways:

2. P_e: either φ_e^A is not total, or recursive, or $A \leq_T \varphi_e^A$.
3. Q_e: either φ_e^B is not total, or recursive in A, or $B \leq_T \varphi_e^B \oplus A$.

An interesting feature of the construction is that requirements Q_e have interactions with requirements R_e and a finite injury argument is used.

The requirement $A \leq_T B$ will be directly satisfied by our initial tree system and the requirements that A is not recursive and B is not recursive in A are automatically satisfied since all minimal degrees, together with the recursive degree, are $\mathbf{GL}_2$.

VI.5 One R Requirement

In this section, we are only handling a single requirement R_e with other types of requirements P_i, Q_i for $i \in \omega$. First we give the initial tree (T_0, S_0) and see how A' comes into play in the construction. Discussions in this section will provide an introduction and a "local picture" about the full formal construction in the next section. Sometimes we might be a bit vague about some notions and claims, and may use terms before their formal definitions. The reader can refer to Section VI.6 for formal definitions, full constructions and detailed verifications.

VI.5.1 Initial tree system

In the initial tree system, T_0 is simply the identity function on $2^{<\omega}$, i.e., the full binary tree. For every $\tau \in 2^{<\omega}$, $R = S_0(\tau)$ is defined to be a tree $2^{2|\tau|} \to 2^{<\omega}$ as follows: First put $R(\emptyset) = \emptyset$. Once we have defined $R(\mu) = \rho$ for $|\mu| = 2e < 2|\tau|$, we let $R(\mu * 0) = \rho * (\tau(e)) * 0$ and $R(\mu * 1) = \rho * (\tau(e)) * 1$. On the 0 side, we simply let $R(\mu * 0 * 0) = \rho * (\tau(e)) * 0 * 0$ and $R(\mu * 0 * 1) = \rho * (\tau(e)) * 0 * 1$. On the 1 side, we check whether $\varphi_e^\tau(e)$ converges, if not, then $R(\mu * 1)$ is a leaf of R; if it converges, then let $R(\mu * 1 * 0) = \rho * (\tau(e)) * 1 * 0$ and $R(\mu * 1 * 1) = \rho * (\tau(e)) * 1 * 1$. From this point of view, we are actually defining the tree system in blocks.

In this setting, the 1-node of a block guesses that $\varphi_e^A(e)$ converges, and it eventually branches on $S_0(A)$ if and only if such guess is correct. We say that this block is *associated with* index e. Note that by the Padding Lemma, for each index e, one can uniformly find an infinite recursive set $I_e = \{e_0, e_1, e_2, \dots\}$ such that for each i, $\varphi_e^{(\cdot)}(e) = \varphi_{e_i}^{(\cdot)}(e_i)$ as Turing functionals. Here we abuse notation and say that this block is also *associated with* each e_i as well, but it is still true that if a block is associated with e then its 1-node branches along path A if and only if $\varphi_e^A(e)$ converges.

Now for every index e, we can recursively find infinitely many blocks associated with e above any even-node. We will make (a modified version of) this property true for every tree system we construct. Each tree system (T, S) has at least one recursive *aiding functional* $f^A(e, \rho, \tau)$ which, given any path $A \in [T]$ which extends τ as oracle, any index e and any even-node ρ on $S(\tau)$ (more precisely, an "accessible" one, see the detailed construction later and property (7) in Definition VI.6.1), returns a $\tau' \supset \tau$ which is an initial segment of A and a block above ρ which is associated with e on $S(\tau')$. For example, our initial tree system easily has a recursive aiding functional. Later we will divide all blocks into two (or more) groups, and we will have one aiding functional for each group.

The following is easy to see, but it is convenient to mention it here.

Fact VI.5.1. *If (T, S) has a recursive aiding functional f and ρ is an (unterminated) even-node on $S(\tau)$, then* $\mathbf{FSTS}(T, S, \tau, \rho)$ *also has a recursive aiding functional g.*

Note that in the construction we might *terminate* some even-nodes so that they will not continue to branch, and we do not require to have aiding functionals above these nodes.

We will make this notion of aiding functionals explicit in Definition VI.6.1 for the full construction.

Whenever we say "we find a block associated with index e above a node ρ (at τ)", we mean that we use the aiding functional to get such a block using the leftmost path above τ as A. This process is recursive.

VI.5.2 Pruned subtree

Suppose we are given a tree system (T, S) (with a recursive aiding functional f) and we want to handle the single requirement R_e. As we mentioned before, we are going to define a new notion of "narrow subtree" for B which depends on A'. This is also the reason why we try to "code" A' into the initial tree system, and we will keep "coding" A' into every tree system we construct in a similar way.

To avoid confusion, we use a new notion, *pruned subtree*, for the B and S part of the construction and keep the notion of narrow subtree for the A and T part.

The key point in Sasso's narrow subtree notion is a density property: for every node τ on the narrow subtree $\mathbf{Nar}(T)$, there is an extension τ' of τ on the original tree T but off $\mathbf{Nar}(T)$, i.e., the nodes that are off $\mathbf{Nar}(T)$ form a *dense* set with respect to T. So in order to give a definition of pruned subtrees, we need to specify a dense subset of nodes.

We first define a new subtree system $(\bar{T}, \bar{S})$ and separate all blocks in this new tree system into two groups of blocks P and N, which stand for, respectively, "positive" and "negative" blocks for the requirement R_e. The positive blocks will be used later as coding positions for A' for R_e. The negative blocks are not used as coding positions for this requirement R_e, and we need them because in the next section in proving the full theorem we will have to reserve these blocks for other R type of requirements. In order to introduce a lemma (Lemma VI.5.3) which is needed there, we use a "P and N" version of the construction already here.

At stage 0, we find a τ and a block $\mathbb{B}$ associated with 0 on $S(\tau)$; then we define τ to be the root of our new tree $\bar{T}$ and define $\bar{S}(\tau)$ to have this single block (here we mean a two level tree structure whose range is exactly the nodes on this block $\mathbb{B}$). We put this block into N.

At the next stage, first let τ' and τ'' be two extensions of τ on T. On the τ' side, we find a $\tau^* \supset \tau'$ such that we can extend the top-nodes of the block $\mathbb{B}$ to some other blocks also associated with 0 on $S(\tau^*)$ (if we see that the 1-node of $\mathbb{B}$ branches, then we also fill this block and find appropriate blocks above the 10 and 11-nodes of $\mathbb{B}$). We then define τ^* to be $\bar{T}(0)$ and define $\bar{S}(\tau^*)$ to have these two levels of blocks (see Figure VI.2). Then we put all the blocks we find in the second stage into P (note that they are still associated with 0). Here is an easy lemma which summarizes this construction. (The proof is immediate from the existence of an aiding functional.)

Lemma VI.5.2. *Let e be any index. If ρ is an even-node on $S(\tau)$, then we can find $\tau' \supset \tau$ (in practice, we find such τ' along the leftmost path of T) and a block associated with e above ρ along the leftmost path on $S(\tau')$.*

Note that here we have actually changed the root of these second level blocks (and inductively also every block except the bottom one) in the construction. Later we will call these blocks in the old tree system *old* blocks and these in the new tree system *new* ones. In some sense, the blocks are "new" but the nodes are "old".

We do the same for the τ'' side and this finishes the construction at the second stage. Then above all current top-nodes of these second level blocks, we can find blocks associated with 1 and put them into N, and at the next level we have P-blocks associated with 1, and so on. At the same time we check whether the 1-nodes in all these blocks branch in the old tree system: if any one branches, we also extend it in the new system with the old 10-node and 11-node, and continue the construction above them in the same way.

Following this, we build the new subtree system $(\bar{T}, \bar{S})$ with alternating levels of (new) blocks in N and P, with both having a full representation of blocks associated with each e. Blocks in P (resp. N) are called P-blocks (resp. N-blocks). It is easy to see that both P and N have their own recursive aiding functionals in $(\bar{T}, \bar{S})$. (See also the full construction of pruned subtrees in §VI.6.1.)

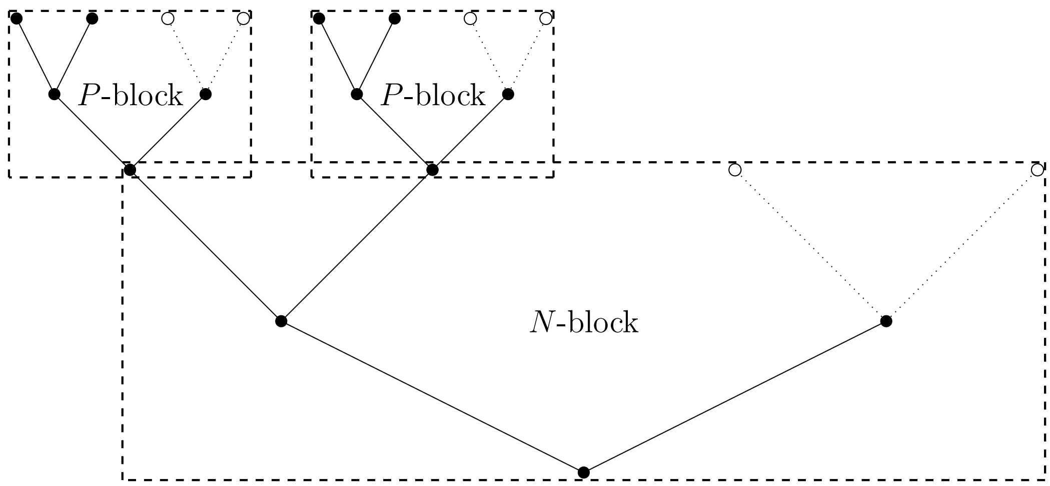

Figure VI.2: P-N-block structure on $\bar{S}(\tau^*)$

We define a dense set of nodes D_e for each A: for every P-block $\mathbb{B}$ on $\bar{S}(A)$, note that it is associated with an index e; if $\varphi_e^A(e)$ converges, then do nothing; if it diverges, then put the 01-node of $\mathbb{B}$ into D_e. The density comes from the simple fact that there are infinitely many indices e such that $\varphi_e^A(e)$ diverges for every A. More importantly, $(\bar{T}, \bar{S})$ has the feature that, for every oracle functional $\varphi_x^{(\cdot)}$, we have on each $\bar{S}(A)$, a P-block associated with x and a 01-node which is in D_e if the corresponding computation diverges with oracle A. We shall preserve this property for every subsequent tree system and so we can implement certain types of constructions at any time we need (see §VI.5.5).

All nodes in D_e are called *witnesses* (i.e., they witness the fact that B is off the pruned subtree which we will define later). In a P-block $\mathbb{B}$ on $\bar{S}(\tau)$, if the associated computation does not converge at τ, then we call the 01-node of $\mathbb{B}$ a *potential witness* at τ. So a potential witness is a node which looks like a witness at this τ but may change to a nonwitness later at some $\tau' \supset \tau$, in which case the corresponding computation converges at that τ'.

A rough idea is that, if a P-block is full, then all its top-nodes are allowed to be on the pruned subtree, otherwise only the 00-node is allowed.

The *pruned subtree* of $\bar{S}(A)$ is the tree formed by removing all nodes in D_e and their extensions. We denote this subtree by $\mathbf{Pru}(\bar{S}, A)$. It is easy to see that this pruned subtree is recursive in A'. So the statement "B is a path on $\mathbf{Pru}(\bar{S}, A)$" is $\Pi_1^0(B \oplus A')$. Unlike Sasso's narrow tree construction, this pruned subtree will not appear in the real construction and we will start with $(\bar{T}, \bar{S})$ at the next stage. In fact, it cannot appear since it is recursive in A' but not in A.

We let x_e be such that $x_e \notin (B \oplus A')'$ if and only if B is a path on $\mathbf{Pru}(\bar{S}, A)$, and we are going to force $(B \oplus A')'(x_e)$ to be different from $\lim_s \varphi_e^{B \oplus 0'}(x_e, s)$. We also call the sequence $\langle \varphi_e^{B \oplus 0'}(x_e, s) \rangle_{s \in \omega}$ (or with B replaced by some ρ) an *x_e-sequence*.

In the end, if $\lim_s \varphi_e^{B \oplus 0'}(x, s)$ does not exist, then we will keep B on the pruned subtree of $\bar{S}(A)$. Slightly differently from the common usage, we say that the limit exists if we only see finitely many changes in the x_e-sequence, and so it is possible that some entries diverge (if this happens, the requirement is automatically satisfied). If the limit is 1, we also keep B on the pruned subtree. If the limit is 0, we make B go off the pruned subtree. We always keep the root of any subsequent tree system (i.e., $S(T(\emptyset))(\emptyset)$) on the pruned subtree unless the last possibility happens.

VI.5.3 Satisfy P_i

Suppose we have a tree system (T', S') at some stage and we want to satisfy P_i.

To make A minimal, we only have to change T' without changing any $S'(\tau)$. So the P-N structure is preserved and we only need to take the restriction of S' to the new domain as our new system.

This is handled exactly in the same way as in the standard Spector minimal degree construction. We ask whether the following is true:

$$\exists \tau \forall \tau_0, \tau_1 \supset \tau \ [\neg(\tau_0 |_i \tau_1)].$$

If so, we take $T^* = \mathbf{FS}(T', \tau)$ and it is routine to show that φ_i^A is either partial or recursive. Otherwise we take the standard i-splitting subtree T^* of T' and claim that $A \leq_T \varphi_i^A$ (Lemma I.3.1). In both cases, we take S^* to be the restriction of S' to the range of T^* and pass (T^*, S^*) to the next stage. It is easy to find new P-aiding functional and N-aiding functional for the new tree system.

VI.5.4 Satisfy Q_i

Suppose we have (T', S') and we want to satisfy Q_i, and suppose that the requirement R_e is still active.

The basic idea is as follows: we cannot ask the i-splitting question for every node ρ, since some nodes are leaves, and some might lead us off the pruned subtree permanently. We only want to consider some "good" nodes at this time, good enough to be the root of a new tree system.

As we mentioned before, along our construction, we sometimes may terminate some of the nodes on $S(\tau)$ and then these nodes cannot be extended on $S(\tau')$ for any $\tau' \supset \tau$. We say τ *terminates* these nodes. See the construction of splitting trees below (see also property (1) in Definition VI.6.1 for restrictions on termination).

An even-node ρ is *accessible* at τ if there is no potential witness below ρ on $S'(\tau)$, and ρ has not yet been terminated by any $\tau' \subset \tau$. Whenever we call a node accessible, it is always assumed that the node is an even-node. Intuitively, accessible nodes are these that we can use without worrying about whether we might permanently get off the pruned subtree. We use $Acc(\tau)$ to denote the set of accessible nodes at τ.

We will guarantee that in any splitting construction, we never terminate an accessible node ρ at τ unless it was terminated by a previous node $\tau' \subsetneq \tau$ (at which the node ρ was not accessible). So if ρ is accessible at τ, then it is accessible at every $\tau' \supset \tau$. Note that there is no circular dependency between accessibility and termination: we have the old tree system where accessibility and termination have been defined, and in constructing the new tree system we may terminate some nodes (by using the old notion of accessibility on the old tree system) and after the construction we can define the new accessible nodes for the new tree system.

We will also ensure that, in a P-block, if the root is accessible, then all top-nodes, except possibly the 01-node, are accessible. In a N-block, if the root is accessible, then all top-nodes are also accessible.

Now whenever we say "find a block associated with e above ρ", we always mean that we find a block which is along the leftmost path of $S'(\tau')$ above this ρ along the left most path of T' above the τ with which we are currently working. So if this ρ is accessible, the root of the block we find by this process is also accessible. It is worth mentioning here that in Fact VI.5.1, if f can find such block along the leftmost path, then so can g.

To satisfy our requirement Q_j, we ask the following question:

$$\exists \tau \exists \rho \in Acc(\tau) \forall \tau' \supset \tau \forall \rho_0, \rho_1 \supset \rho; \rho_0, \rho_1 \in Acc(\tau') \; [\neg(\rho_0 |_i \rho_1)].$$

Basically, the question is asking, in the "accessible part" of the tree system, whether there is a pair (τ, ρ) such that there are no i-splittings above it.

If so, we take $\mathbf{FSTS}(T', S', \tau, \rho)$ and claim that φ_i^B is either partial or recursive in A, if R_e never acts after this stage. In this case, we will know that B is finally on the pruned subtree of $S(A)$. Now if φ_i^B is total, we can simply search on $S'(A)$ above this node ρ for an accessible node where $\varphi_i^{(\cdot)}(x)$ converges. This process will stop and should give us the correct answer, since every node $\rho' \subset B$ is eventually accessible at some $\tau' \subset A$. Therefore φ_i^B is recursive in A.

If the answer is no, then we know that:

$$(\dagger):\ \forall\tau\forall\rho \in Acc(\tau)\exists\tau' \supset \tau\exists\rho_0, \rho_1 \supset \rho; \rho_0, \rho_1 \in Acc(\tau')\ (\rho_0|_i\rho_1).$$

Now we do an i-splitting construction to get a new tree system (T^*, S^*). As we assumed, the root of the tree system ρ is always accessible (at the root of T'). So we define $T^*(\emptyset) = T'(\emptyset)$ and $S^*(T^*(\emptyset))$ to be a singleton tree consisting of ρ.

Then we take the two immediate successors τ', τ'' of $T'(\emptyset)$ on T'. Using ($\dagger$) we find a $\tau_1 \supset \tau'$ and two accessible i-splitting extensions ρ_0, ρ_1 of ρ at τ_1. Extend τ_1 to τ_2 where we can extend ρ_0 and ρ_1 from the leftmost path to N-blocks $\mathbb{B}_0$ and $\mathbb{B}_1$ respectively, both associated with 0. Let η_j be the j-node in $\mathbb{B}_j$ extending ρ_j for $j = 0, 1$. Note that η_0 is not an even-node and so we cannot apply ($\dagger$) directly, but since we are taking the leftmost path, the 00-node $\xi \supset \eta_0$ in $\mathbb{B}_0$ is still accessible at τ_2 (see Definition VI.6.1). Now apply ($\dagger$) again with τ_2 and ξ to get η_0 extended by two i-splitting accessible nodes (at $\tau_3 \supset \tau_2$) and use the same technique to extend these two nodes along their leftmost extensions respectively to N-blocks associated with 0 (at some $\tau_4 \supset \tau_3$), and take the corresponding 00-node and 01-node in each. Let these be η_{00} and η_{01}. Now at $S'(\tau_4)$, we check whether η_1 is a leaf or not. If it is not a leaf, then we know that the 10-node in $\mathbb{B}_1$ is accessible and do the same thing extending η_1, i.e., we can find accessible i-splitting nodes η_{10} and η_{11} (both in old N-blocks associated with 0) extending η_1 in the same way. If it is a leaf, then we do nothing. So now we are at some τ with five or seven nodes (the root, η_0, η_1, η_{00}, η_{01}, and possibly η_{10}, η_{11}). We define τ to be the left immediate successor of $T^*(\emptyset)$ on T^* and define $S^*(\tau)$ to have one single N-block (associated with 0) containing these five or seven nodes.

Here is a lemma which summarizes this construction.

Lemma VI.5.3. *Let X be either P or N and let x be any index. If ($\dagger$) holds, then for any ρ accessible at τ, we can find a $\tau' \supset \tau$ where there are (four or six) appropriate nodes above ρ which are originally in corresponding positions in old X-blocks associated with x (for example, the new 01-node we pick is originally a 01-node in an old block); and these nodes form a new i-splitting X-block associated with x. Additionally, all these nodes are originally in old X-blocks whose roots are accessible at τ'.*

It is crucial that the new 01-node is an old 01-node in a block associated with the same index. When we try to satisfy the requirement R_e and pick a 01-node on a new tree system (see Plan II in Section VI.5.5), this property guarantees that this 01-node is also a 01-node the old tree system where we defined pruned trees and x_e-sequences.

Then we do the same thing for the τ'' side, extend it to some other τ with another i-splitting N-block defined. This finishes stage 1 of the construction. In general, in our splitting tree system construction, blocks are new, but the nodes are old ones in the old blocks. In this case, we *do not* say that we are *redefining* the blocks, as it could be confused with another process in the injury argument in the next section.

At the next stage, we see that η_{00} is accessible at τ as above, and we can do almost the same construction, except that we now look for P-blocks associated with 0 instead of N-blocks. In the end, we have one P-block extending each top-node in the first level N-blocks. Also we check whether the 1-node η_1, which is previously a leaf, branches, if so we need to do a similar construction to branch it by two nodes and fill the first level N-block.

The trouble comes at the next stage. A 01-node in a P-block might be a potential witness and hence inaccessible. So we can no longer apply (†) to it to get i-splitting extensions. In this case, we simply copy what is in S' block by block to S^* above this 01-node at subsequent stages, until we find out that it is accessible.

The most difficult part of the construction is when we find out that a potential witness η (a 01-node of a P-block) is not a witness and so some inaccessible nodes above it becomes accessible. At that time we have so far seen some levels of N and P blocks above this η that we have copied down from the previous system S' (and maybe we have already terminated some of the nodes in previous splitting tree constructions). Suppose we are at τ in the construction and η is not a potential witness at τ. On $S'(\tau)$ above η as we have copied down so far, we terminate all 1-nodes above η which haven't branched, and for every 01-node ρ in any P-block which is still a potential witness at τ, we also terminate all extensions of such ρ. That is to say, we are left with a finite number of accessible (at τ) top-nodes in the highest level blocks on $S'(\tau)$ above η. For these finitely many, we can apply (†) some finite number of times to find an extension τ' of τ where we can extend each top-node to a pair of a 0-node and a 1-node in appropriate N-blocks (we have destroyed the P-N-block structure from η, so we start with the next N-block as extending the P-block where η lives) where the roots of all blocks are accessible at τ', and all these extensions pairwise i-split (see Lemma VI.6.7 for the detailed construction). Then we continue the construction as before to finish the blocks.

So this new tree system is not strictly i-splitting but somewhat delayed in its splittings, and we can control how this delay happens. More importantly, if Q_i is not injured in later constructions and B is on the pruned subtree $\mathbf{Pru}(\bar{S}, A)$, then we can compute B from $\varphi_i^B \oplus A$ as follows. To find B, we simply go through $S^*(A)$: At first the tree is perfectly i-splitting so we will not have trouble finding initial segments of B using φ_i^B; If we ever meet a potential witness η, then, as we know that B is on the pruned subtree $\mathbf{Pru}(\bar{S}, A)$, this η will be accessible at some $\tau \subset A$. So we go along the construction of (T^*, S^*) and find the first time when we see such a τ on T^*. At that time, as the construction proceeded, we terminated a lot of nodes above η and only extended the nodes that were left, and we have made all of their extensions i-split at the next level. So we can effectively find the next initial segment of B.

It is not difficult to see that this new tree system also has aiding functionals. Note that we only consider aiding functionals for accessible nodes, and above an accessible η which was previously a potential witness as above, we need to go beyond the terminated nodes and find a block such that the nodes are never terminated (see Definition VI.6.1).

If we are dealing with requirement Q_j with R_e inactive, i.e., we believe that we have satisfied R_e permanently, the construction is very similar and much easier. We will ask the key question without worrying about accessibility but only evenness. There are no P-blocks to handle, and so we only construct appropriate N-blocks level by level in the case that we get a negative answer. The tree system we construct will be i-splitting and the final verification directly follows from Lemma

VI.3.1.

VI.5.5 Satisfy R_e

We also want to (partially) satisfy the requirement R_e at subsequent stages by forcing infinitely many changes in the x_e-sequence $\langle \varphi_e^{B\oplus 0'}(x_e, s)\rangle_{s\in\omega}$. Similarly we only want to go to accessible nodes, and so we ask the following (let λ be the root of the given tree system (T', S')):

$$\exists\tau\exists\rho \in Acc(\tau)[\varphi_e^{\rho\oplus 0'} \text{ has more changes in the } x_e\text{-sequence than } \varphi_e^{\lambda\oplus 0'} \text{ has}].$$

If so we take $\mathbf{FSTS}(T', S', \tau, \rho)$ and proceed to the next step. We make at least one more change in the x_e-sequence, and if this happens infinitely many times, then the limit does not exist. In this case, we satisfy the requirement by infinitely many attempts, and the requirement R_e is always active. In the end B will be on the pruned subtree $\mathbf{Pru}(\bar{S}, A)$.

If we ever get a negative answer, then we know that we cannot have more changes in the x_e-sequence, at least for the nodes in the accessible part of the tree system. This case is divided into two subcases, depending on the final value of the x_e-sequence we have seen so far in the accessible part.

(Plan I) If the final value is 1, then we want to keep B on the pruned subtree. We can satisfy this by simply removing all P-blocks. More precisely, let $\mathbb{B}'$ be the N-block above a P-block $\mathbb{B}$ extending its 00-node, i.e., the 00-node of $\mathbb{B}$ is the root of $\mathbb{B}'$. We first remove $\mathbb{B}$. Then we define a new N-block: the root is the root of $\mathbb{B}$, and every other node is the node in the same position in $\mathbb{B}'$. From another point of view, though it is not quite accurate, we "terminate" all (extensions of) 1-nodes and 01-nodes of every P-block (See Lemma VI.6.5). It is easy to see that B is now forced to be on the pruned subtree and there will be no more P-blocks. The requirement R_e is permanently satisfied and becomes inactive.

(Plan II) If the final value is 0, then we want to take B off the pruned subtree. First we take a narrow subtree $\mathbf{Nar}(T')$ of T' and let S'' be the restriction of S' to the range of $\mathbf{Nar}(T')$. As we saw before in Sasso's construction, there is an x such that $x \notin A'$ if and only if A is on $\mathbf{Nar}(T')$. Now we find a τ on $\mathbf{Nar}(T')$ and a leftmost P-block $\mathbb{B}$ associated with x on $S'(\tau)$ (in fact, any P-block associated with x with an accessible root works here). Let ρ be the 01-node of $\mathbb{B}$. We take $\mathbf{FSTS}(\mathbf{Nar}(T'), S'', \tau, \rho)$ and remove all P-blocks above ρ as in Plan I above. The new subtree system (T^*, S^*) has no P-blocks and we only need to verify that the requirement R_e is permanently satisfied (hence we set R_e to be inactive).

All the nodes in the new subtree system have only one common potential witness ρ below them (from the viewpoint of (T', S')). Suppose that an even-node $\eta \supset \rho$ on $S^*(\tau')$ has at least one more change in the x_e-sequence $\langle \varphi_e^{\eta\oplus 0'}(x_e, s)\rangle_{s\in\omega}$. We can then extend τ' to τ^* on T which is off $\mathbf{Nar}(T')$ and τ^* is long enough to see that $\varphi_x^{\tau^*}(x)$ converges. On $S'(\tau^*)$, ρ is no longer a potential witness, and so η becomes accessible at τ^*, since there are no potential witnesses between ρ and η in the original tree system (T', S'). This of course contradicts the negative answer to our key question above, which asserts that there is no such accessible η with more changes in the x_e-sequence. So the limit is forced to be 0. Noticing that by induction this ρ is a 01-node in a block associated with x on the tree system $(\bar{T}, \bar{S})$

when we defined pruned subtrees in Section VI.5.2, it is easy to see that we have forced B off the pruned subtree, since $\varphi_x^A(x)$ diverges for any $A \in [\mathbf{Nar}(T')]$.

One useful intuition about the above argument is that we can take A off the narrow subtree, and consequently drive B back on the pruned subtree.

Now this construction will injure all requirements Q_i that have been satisfied since we defined the pruned subtree for $(\bar{T}, \bar{S})$: if we took a full subtree system for Q_i, we are now in the inaccessible part of it and so the key question does not govern the situation; if we took a splitting subtree system, then we just copied the previous tree system in the construction when we see that ρ is a potential witness and not accessible, and so we do not guarantee splitting. Now the solution is, for all these requirements Q_i that we have injured, we just try to satisfy them again. This is how an injury argument comes into the proof of the full theorem in the next section. For Plan I, it seems here that it will not injure other requirements, but it actually causes injury in the full construction (interestingly, it ought to injure some other requirements so that they will not injure it later in the construction).

VI.6 Proving the Full Theorem

Now we add all requirements R_e into the list. Note that no requirement P_e will be injured, since our sequence of tree systems is nested and each T_{i+1} is a subtree of T_i. In addition, the satisfaction of P_e will not injure other requirements, basically because our new system is just a restriction of the old one; so no $S(A)$ will change in the construction, and it will not injure any satisfied Q_e or R_e requirements (see also Lemma VI.6.2). Therefore it suffices to consider all P_e requirements separately (in fact, they are handled in the same way as in §VI.5.3 and we omit the construction here) and only analyze the interaction between R_e and Q_e requirements.

Now we put all of them into a priority list:

$$Q_0, R_0, Q_1, R_1, \ldots, Q_e, R_e, \ldots$$

So each requirement must respect all requirements before it in this list. We will keep in mind several ideas: (1) For each R_e, we will define a new group of blocks P^e from the current N-blocks and define corresponding e-pruned subtrees, immediately after we satisfy Q_e. (2) Whenever we try to satisfy a requirement Q_e, we will cancel all current active or inactive (i.e., claimed to be satisfied) requirements R_i for all $i \geq e$ we have considered, and redefine their P^i-block groups and pruned subtrees later. (3) Whenever we redefine P^e-blocks for some e, we might also injure the satisfaction of every Q_i $(i > e)$ since in their construction they have to respect these old-version P^e-blocks (see construction below for details). (4) In addition, whenever we (claim to) permanently satisfy R_e by following Plan I or Plan II as in §VI.5.5, we might injure every Q_i satisfied since we last defined the P^e-block group. Note that it is automatic that for any Q_i injured in this way, $i > e$ since otherwise we would redefine a P^e-block group after that.

One thing might be important to note again: we say that we *define* or *redefine* P^e-blocks at some stage if and only if (T, S), the tree system at the beginning of the stage, does not have P^e-blocks and (T', S'), the tree system at the end of the stage, does have P^e-blocks. In our construction, we will often *reconstruct* the P^e-blocks in the sense that both (T, S) and (T', S') have P^e-blocks, and these blocks might be different on the two trees, but the new nodes in new P^e-blocks are originally nodes in old P^e-blocks (in fact, in the same position in the blocks).

Finally the injury argument works as follows: Q_0 will not be injured by any other requirements so it is permanently satisfied at the first step. Then for R_0, note that the P^0-blocks are defined right after we satisfy Q_0 and are never redefined. If R_0 is always active, then it will not injure other requirements, and so in particular, Q_1 is not injured. If R_0 is permanently satisfied at stage s, we then satisfy Q_1 after that and Q_1 will not be injured again. It is easy to see that the construction works by a typical finite injury argument (see §VI.6.5).

Our forcing notion actually consists of (1) a tree system (T, S), (2) a finite set of nodes on each $S(\tau)$ terminated by τ and (3) also on each $S(\tau)$, the information about the groups of blocks and the index (or indices) associated with each block (and hence the aiding functionals). We will regard (2) and (3) as being recursively embedded into the tree system. For the initial tree system (T_0, S_0), no τ terminates any node and every block is an N-block associated with the index we specified in the description.

VI.6.1 Pruned subtrees

We will define e-pruned subtrees and P^e-blocks immediately after each time we satisfy the requirement Q_e. Suppose that we are given R_e and a tree system (T, S) where each $S(\tau)$ consists of, level by level, N-blocks, P^{e_0}-blocks, P^{e_1}-blocks,. . . , and P^{e_t}-blocks, and each e_i is less than e. In addition, we have defined the notions of e_i-pruned subtrees, e_i-witnesses and e_i-potential witnesses for each e_i.

As in §VI.5.2 we separate a group of P^e-blocks from N-blocks. Intuitively, the set of N-blocks works like a *cornucopia* such that we can take out a full collection (P^e) of blocks associated with each index, and we are still left with such a full collection in the new N. We note again that, each time we pick a block above a node, we always go through the leftmost path to find one in order to avoid e_i-potential witnesses for R_{e_i} requirements and interactions between different R_e requirements.

There is another thing to keep in mind. At this time, (like the believability conditions in any priority tree argument) we believe that all requirements R_{e_0} to R_{e_t} are infinite, i.e., we will not satisfy them in a permanent way by Plan I or Plan II later. Hence now we also believe that in the end B is on the e_i-pruned subtree for each e_i.

We first construct a new subtree system $(\bar{T}, \bar{S})$. We start from the root, pick an N-block associated with 0, then extend these top-nodes to an P^{e_0}-block associated with 0 and so on (in the same way as in §VI.5.2). At the $t+3$-rd level, we then extend the top-nodes to an N-block associated with 0 again, and rename it as a P^e-block. The only exception is that when we try to extend an e_i-potential witness, we will copy whatever is in S without change in the same way as in the splitting subtree system construction in §VI.5.4. So above an e_i-witness, there are no P^e-blocks defined, and above an e_i-potential witness, we define P^e-blocks only if it becomes a nonwitness. When we see that an e_i-potential witness η becomes a nonwitness, we terminate the unbranched 1-nodes and e_j-potential witnesses above η as in §VI.5.4 and we are left with some top-nodes. Then we extend these top-nodes by picking appropriate blocks above them and define P^e-blocks there.

For example, along the leftmost path, the blocks we build are exactly (let $X(i)$

denote an X-block associated with i):

$$(*):\ N(0), P^{e_0}(0), \ldots, P^{e_t}(0), P^e(0), N(1), P^{e_0}(1), \ldots, P^e(1), N(2), \ldots$$

Let us describe this construction in detail at the inductive step: whenever we put τ into $\bar{T}$ and define $\bar{S}(\tau)$, $\bar{S}(\tau)$ is going to have the following finite sets:

1. a set X of top-nodes that are leaves on $\bar{S}(\tau)$ with no e_i-potential witnesses (at τ) below any one of them, which need immediate handling;
2. a set Y of 1-nodes in different blocks that have not yet branched, for which we shall wait to see what happens;
3. for each e_i-potential witness η_j (such that no other node below it is an e_j-potential witness for some e_j at τ), a subtree $Z(\eta_j)$ of $\bar{S}(\tau)$ above η_j, where we continue copying from S until we see that this η_j is not an e_i-witness.

Now on T, τ has two immediate successors τ' and τ''. We first pick τ'. Applying an easy modification of Lemma VI.5.2 (strictly speaking, we apply property (7) in the next subsection) a finite number of times, we can find an extension $\tau^* \supset \tau'$ where each leaf in X gets extended to appropriate blocks (here appropriate refers to the correct N or P^{e_i}-block associated with the correct number, according to the list $(*)$ above, and note that for a P^e-block in the list, we first find an N-block in S and rename it as a P^e-block in $\bar{S}$). Each time we remove the leaf from X. After we have finished doing this for all leaves in X at τ^*, we then check whether any 1-node ρ in Y has branched at τ^*, if so we can simply fill the new block with the 10-node and 11-node in the old block extending ρ.

Now for each $Z(\eta_j)$, we check whether η_j turns out to be a nonwitness or not (at τ^*): if not, we shall copy all blocks from $S(\tau^*)$ above η_j to the new system to extend $Z(\eta_j)$; if so, we first copy everything from $S(\tau^*)$ as in the other case, and we terminate the following nodes above η_j:

1. any node that has been previously terminated by τ^*;
2. any 1-node that has not branched at τ^*;
3. any even-node that has an e_i-potential witness (at τ^*) below it for some e_i.

So in the end we are left with a finite list of unterminated even-nodes which are leaves above η_j on $S(\tau^*)$. For convenience we call them *key-nodes*. We have destroyed the block structure of $S(\tau^*)$ from η_i to these key-nodes and begin our new block-search accordingly at the next stage.

Finally we define τ^* to be the left immediate successor of τ on the new tree $\bar{T}$, and define $\bar{S}(\tau^*)$ to be the new tree we construct as above. Note that our new X consists of new leaves that do not have any potential witness below them, including all these 10-nodes and 11-nodes we found for 1-nodes in Y, and all the key-nodes from the termination process.

On the τ'' side, we essentially do the same construction with τ' replaced by τ''. This finishes the inductive construction of $(\bar{T}, \bar{S})$. Now on each $\bar{S}(A)$, a 01-node ρ on a P^e-block associated with j is called an *e-witness* if $\varphi_j^A(j)$ diverges;

ρ is called an *e-potential witness at* τ if $\varphi_j^\tau(j)$ diverges. The *e-pruned subtree* of $\bar{S}(A)$, $Pru_e(\bar{S}, A)$, is defined as the subtree of $\bar{S}(A)$ with all e-witnesses and their extensions removed.

Later, whenever we say e-pruned subtree, we always refer to the version of $Pru_e(\bar{S}, A)$ at the last time it was defined.

On each tree system we construct, there will be different groups of blocks $P^{e_0}, P^{e_1}, \ldots, P^{e_t}$, and there will be e_i-potential witnesses for each e_i. For simplicity, if it does not cause any confusion, we will call a node ρ a *potential witness* (resp. *witness*) at τ if it is an e_i-potential witness (resp. e_i-witness) at τ for some e_i. We carefully avoid interactions between different R_e-requirements and so each ρ can be an e_i-potential witness for only one e_i.

We then take x_e to be such that $x_e \notin (B \oplus A')'$ if and only if B is on the e-pruned subtree of $\bar{S}(A)$. Note that whenever we redefine the P^e-blocks we also redefine all other notions and change the value of this x_e. In the end we will prove that x_e eventually stops changing.

We will refer the construction here as a "pruned subtree construction", although it is not quite accurate, as $(\bar{T}, \bar{S})$ is not a "pruned subtree system" of (T, S).

VI.6.2 Special tree systems

We are going to construct a nested sequence of tree systems and we want to make sure that, for example, whenever we take a full subtree system above a certain node, the set of paths is not empty and the subtree system also has some nice properties needed to continue the construction. So we introduce the notions of *accessible nodes* and *special tree systems* to simplify the arguments and make our proof smoother.

We give a list of *properties* that all of our tree systems will satisfy. Some of the properties are redundant, i.e., they are direct consequences of other properties, but it is more convenient to put them in the list and they provide a better picture of special tree systems.

In a tree system (T, S), each $S(\tau)$ consists of $N, P^{e_0}, P^{e_1}, \ldots$ and P^{e_t} blocks level by level. From §VI.6.1 we have the notion of e_i-potential witnesses for each e_i. Note that this notion applies not only to the nodes on the tree system but also to other strings. However, we will guarantee the following:

0. For every e_i, for every τ and for every pair of successive nodes $\rho \subset \rho'$ on $S(\tau)$, no string in (ρ, ρ') is an e_i-potential witness at τ. In fact, only 01-nodes of P^{e_i}-blocks can be e_i-potential witnesses.

So whenever we say an e_i-potential witness, we always refer to a node on the tree system.

A node τ on T might permanently terminate some of the leaves on $S(\tau)$ (i.e., no extension τ' of τ can extend these leaves). Note that if τ terminates ρ, then for τ' extending τ, we also say that τ' terminates ρ. We guarantee that this kind of termination process only happens in the pruned subtree construction in §VI.6.1 or in the splitting subtree system construction in §VI.6.3. In particular, we will need a property that:

1. If an even-node ρ (on $S(\tau)$) has at most one potential witness below it (at τ) and it is not terminated at τ, then it is not terminated at any $\tau' \supset \tau$.

For example, in the primitive version of the splitting subtree system construction in §VI.5.4, an even-node is terminated only if it has two potential witnesses $\rho_0 \subsetneq \rho_1$ below it at some τ, and later ρ_0 becomes a nonwitness before ρ_1 does.

An even-node ρ is *accessible* at τ if for each e_i, ρ has no e_i-potential witness below it at τ, and it is not terminated by any $\tau' \subset \tau$. Whenever we call a node accessible, it is always assumed that it is an even-node.

It is immediate from this definition and properties (0) and (1) that:

2. If ρ is accessible at τ, then it is accessible at every $\tau' \supset \tau$.
3. If the root of an N-block is accessible at τ, then both its 00-node and 01-node are accessible at τ. In addition, if its 1-node is branching, then both its 10-node and 11-node are accessible at any $\tau' \supset \tau$ where it branches.
4. If the root of a P^{e_i}-block is accessible at τ, then the statement in property (3) holds except for its 01-node. In addition, its 01-node is not terminated by any $\tau' \supset \tau$.

We also guarantee that:

5. If ρ is an accessible leaf at τ, then it is not a leaf at any $\tau' \supsetneq \tau$, i.e., such ρ is always branching along every path extending τ.
6. The root of the tree system is always accessible at the root of the tree T.

Finally we need recursive aiding functionals for each tree system.

7. For each group X (X is either N or P^{e_i} for some e_i), there is a recursive oracle functional f such that given any path $A \in [T]$ (as oracle), an accessible node ρ at $\tau \subset A$ and an index e, returns a $\tau' \subset A$ and an X-block associated with e on $S(\tau')$ above ρ along the leftmost path of $S(A)$ (hence the root of this block is accessible at τ').

In practice, we only use f as a recursive function with the leftmost (hence recursive) path on T extending τ. Note that if termination happens, we have destroyed the block structure from the potential witness to the key-nodes left after termination, but since ρ is guaranteed to be accessible, the termination has happened and f returns a block from the leftmost path above the leftmost key-node.

By properties (1) to (4), the block we find by f is *proper*: a block is *proper* if its nodes will not be terminated by any τ on T (with respect to the current tree system (T, S)).

8. The aiding functional f in (7) always finds a proper block.

Definition VI.6.1. A tree system (T, S) is *special* if it satisfies properties (0) to (8) above.

Note that our initial tree system (T_0, S_0) is special and every block is an N-block. Now we provide some technical lemmas for different subtree system constructions.

Lemma VI.6.2. *If (T, S) is special and T' is a subtree of T which is also total, and S' is the restriction of S to the range of T', then (T', S') is also special.*

Proof. Property (6) follows from property (2). The other properties are immediate. $\square$

Lemma VI.6.3. *If (T, S) is special and ρ is an accessible node at τ, then* $\mathbf{FSTS}(T, S, \tau, \rho)$ *is also special.*

Proof. Immediate. $\square$

Lemma VI.6.2 is used for requirements P_e and Lemma VI.6.3 is used for all full subtree system constructions. In addition, we need a lemma for the pruned subtree construction in §VI.6.1.

Lemma VI.6.4. *In § VI.6.1, if (T, S) is special, then the subtree system $(\bar{T}, \bar{S})$ we construct is also special.*

Proof. Property (0) is still true since every time we extend nodes, either we stay on the leftmost path so the root of the block we find is still accessible, or we simply copy the previous tree system. Property (1) holds for $(\bar{T}, \bar{S})$ directly by the construction: none of the nodes on $\bar{T}$ can terminate this type of ρ. For property (7), the construction of the new subtree system guarantees that we can simply go through the leftmost path of $\bar{S}(A)$ from any accessible node looking for the block we want. The other properties are immediate. $\square$

Sometimes in satisfying R_e for some e, we need to *remove* all P^i-blocks for $i \geq e$ in the same way as in Plan I or Plan II in §VI.5.5. Basically, first fix an r; for each P^r-block $\mathbb{B}$, we shall link its root and its 00-node together, making the root of $\mathbb{B}$ the new root of the block above $\mathbb{B}$ extending the 00-node, and discard its 1-node and 01-node, together with anything above them. At each step, we remove all P^r-blocks for one r, and we can iterate this process some finite number of times and get a tree system without any P^r-blocks for any $r \geq e$. We need to know that doing this will not change specialness:

Lemma VI.6.5. *If (T, S) is special and (T', S') is the new tree system we get after removing all P^e-blocks, then (T', S') is also special. (Hence any finite iteration of this process preserves specialness.) In addition, in § VI.6.1, if ρ is a* 01*-node of a P^e-block on $\bar{S}(\tau)$, then the tree system formed by removing all P^e-blocks from* $\mathbf{FSTS}(\bar{T}, \bar{S}, \tau, \rho)$ *is also special.*

Proof. Property (0) follows from property (4). Property (1) might be confusing: Strictly speaking, we are not terminating 1-nodes and 01-nodes in the "removed" P^e-blocks. Our new tree system does not reach any of these nodes and it does not make sense to "terminate" them. All the terminations related to e-potential witnesses are simply removed, so property (1) is still true. The other properties are immediate.

For the second claim, since we only define P^e-blocks above accessible nodes of (T, S), this ρ is actually a 01-node of an old N-block whose root is accessible at this τ in (T, S). Now the tree system $\mathbf{FSTS}(\bar{T}, \bar{S}, \tau, \rho)$ is simply a result of an alternative version (i.e., we start from an index e instead of 0) of our pruned subtree construction in §VI.6.1 for $\mathbf{FSTS}(T, S, \tau, \rho)$, which is special by Lemma VI.6.3. Then by removing all P_e-blocks, we make the root ρ accessible on the new tree system, and other verifications follow the same idea as in the proof of Lemma VI.6.4 and the first claim here. □

In the following construction, at every stage, we always start with a special tree system (T, S) from the previous stage and one requirement P_e (see §VI.5.3), Q_e or R_e which is the highest unsatisfied one in our priority list, and we construct a special subtree system (T', S') which satisfies or partially satisfies the requirement.

VI.6.3 Satisfy Q_e

Suppose we are going to satisfy Q_e with a given special tree system (T, S). We know that, from our discussion, each $S(\tau)$ only consists of levels of $N, P^{e_0}, \ldots$ and P^{e_t}-blocks, and each e_i is less than e.

The idea is similar to the one in §VI.5.4. Now our requirement Q_e must respect all these R_{e_i} requirements whose blocks are still in the tree system.

We ask almost the same key question as in §VI.5.4:

$$\exists\tau\exists\rho \in Acc(\tau)\forall\tau' \supset \tau\forall\rho_0, \rho_1 \supset \rho; \rho_0, \rho_1 \in Acc(\tau')\ [\neg(\rho_0|_e\rho_1)].$$

We take the full subtree system above (τ, ρ) if the answer is yes. Lemma VI.6.3 guarantees that the new subtree system is also special. If the answer is no, we know that:

$$(\dagger):\ \forall\tau\forall\rho \in Acc(\tau)\exists\tau' \supset \tau\exists\rho_0, \rho_1 \supset \rho; \rho_0, \rho_1 \in Acc(\tau')\ (\rho_0|_e\rho_1).$$

So we do a similar e-splitting subtree system construction, creating $N, P^{e_0}, \ldots$ and P^{e_t} e-splitting blocks level by level for accessible nodes, and copy from the previous tree system above 01-nodes that are potential witnesses. We also handle these potential witnesses in the same way, i.e., whenever we find out that they are no longer witnesses, above these nodes, we terminate every 1-node that is not branching and terminate every 01-node which is still a potential witness. Then we are left with some finite number of top-nodes (key-nodes) all of which are accessible at that step, and continue the construction by finding pairwise e-splitting accessible nodes above them.

To simplify our description of the full construction, we provide a framework for creating new e-splitting blocks similar to Lemma VI.5.3. There are a few notational changes, and so we rewrite it here:

Lemma VI.6.6. *Let X be N or P^{e_i} for some e_i and let x be any index. If $(\dagger)$ holds, then for any ρ accessible at τ, we can find a $\tau' \supset \tau$ where there are appropriate nodes above ρ which are originally in corresponding positions in old proper X-blocks associated with x; and these nodes form a new e-splitting X-block associated with x. Additionally, all these nodes are originally in old blocks whose roots are accessible at τ'.*

Proof. See §VI.5.4. □

In addition, as we mentioned there, we also need a lemma for pairwise e-splittings above multiple nodes.

Lemma VI.6.7. *If* (†) *holds, then for any* τ *and any finite set* F *of pairwise incompatible accessible nodes at* τ*, we can find an extension* $\tau' \supset \tau$ *where there are a pair of accessible extensions for each* $\eta \in F$ *and all these extensions pairwise* e*-split.*

Proof. A pair of nodes η_0, η_1 extending ρ is a 1*-level* e*-splitting* if they e-split. A group Z of nodes above ρ is a $(k+1)$*-level* e*-splitting* if there are two nodes $\eta_0, \eta_1 \in Z$ which form a 1-level e-splitting above ρ, $Z_0 = \{\xi \in Z : \xi \supsetneq \eta_0\}$ is a k-level e-splitting above η_0, and similarly $Z_1 = \{\xi \in Z : \xi \supsetneq \eta_1\}$ is a k-level e-splitting above η_1. In other words, if there is an n-level e-splitting above ρ, then we can find an e-splitting binary tree with n many levels above ρ.

We prove this lemma by establishing the following claim: If $|F| = k$ and each $\rho \in F$ has a k-level e-splitting above it on $S(\tau')$, then we can find one pair of nodes at the top level of each k-level e-splitting above each $\rho \in F$ and all of these nodes pairwise e-split.

First we show how this claim is used: by (†), it is easy to find such $\tau' \supset \tau$ where we can find k-level e-splittings above each $\rho \in F$ and all nodes we find are accessible at τ'. Then by the claim we are done.

We prove the claim by induction on k. For the base case $k = 1$, it is trivial. Now assume that the claim holds for k, and we have $k+1$ nodes in $F = \{\rho_0, \rho_1, \dots, \rho_k\}$. In each $(k+1)$-level e-splitting above ρ, call a node *first-class* if it is in the top level of the $k+1$ level tree structure, and call a node *second-class* if it is one level below the first-class ones, i.e., k-levels above a $\rho \in F$.

Among all second-class nodes in all these $(k+1)$-level e-splittings, we first pick one with the longest φ_e-image. Without loss of generality, let η_0 above ρ_0 be this second-class node.

For the remaining nodes $\rho_1, \rho_2, \dots, \rho_k \in F$, we first ignore all the first-class nodes above them and look at the remaining k-level e-splitting structures. By the induction hypotheses, we know that we can find pairwise e-splitting extensions ξ_{i0}, ξ_{i1} above each ρ_i for $i = 1, 2, \dots, k$ and all these ξ's are second-class nodes.

Since η_0 is a second-class node with the longest φ_e-image, and ξ_{i0}, ξ_{i1} e-split, we know that at least one of them e-splits with η_0. Let η_i be one of ξ_{i0}, ξ_{i1} which e-splits with η_0, for each $i = 1, 2, \dots, k$.

Now above each η_i for $i = 0, 1, \dots, k$, we can find two e-splitting first-class nodes and it is easy to see that they pairwise e-split. This finishes the inductive proof of the claim. □

With these two lemmas in hand, let us describe the inductive construction in detail. Suppose we have put τ into T' and defined $S'(\tau)$ as a subtree of $S(\tau)$. As in §VI.6.1, we have three types of nodes in hand:

1. X, which consists of all accessible leaves on $S'(\tau)$ and which will be handled first;
2. Y, which consists of all 1-nodes whose corresponding root is accessible (at τ), for which we will keep waiting to see if they branch in the old system S;

3. $Z(\eta_i)$ for each potential witness η_i (with no other potential witness below), which is the subtree of $S'(\tau)$ above η_i and also the subtree of $S(\tau)$ above η_i, since we simply copied the tree there.

For each $Z(\eta_i)$, we continue copying until we figure out that η_i is not a witness. At that time we terminate some nodes and do an e-splitting construction for multiple nodes.

As before, there are two immediate successors τ' and τ'' of τ on T and we first pick τ'. We start with the leaves in X and apply Lemma VI.6.6 a finite number of times to extend them to appropriate e-splitting blocks. Each time, we remove the corresponding leaf node from X, and we end up with $\tau_0 \supset \tau'$.

Now at τ_0, some of the 1-nodes in Y may have branched and we use the same idea as in Lemma VI.6.6 applying (†) to extend each of these 1-nodes to e-splitting accessible 10-node and 11-node in appropriate blocks to fill the new block where the 1-node lives.

Also possibly some η_i which is a potential witness at τ might turn out to be a nonwitness at the current τ_t, i.e., the corresponding computation converges at τ_t. So we need to handle the subtree $Z(\eta_i)$: as usual, we copy everything from $S(\tau_t)$ above η_i, then terminate these three types of nodes above η_i:

1. any node that has been previously terminated by τ_t;
2. any 1-node that has not branched at τ_t;
3. any even-node that has an e_i-potential witness (at τ_t) below it for some e_i.

In the end, we have a finite list of key-nodes and it is easy to see that all of them are accessible at τ_t. Now we apply Lemma VI.6.7 to extend τ_t to τ_{t+1} at which each key-node is extended by two accessible nodes such that all of them pairwise e-split. Then we can apply Lemma VI.6.6 again to these extensions and get appropriate blocks. Also note that we have destroyed the block structure from η_i to these key-nodes, and so we shall start searching for blocks that work as the next ones in the list $(*)$ in §VI.6.1, which extend the block where η_i lives as 01-nodes.

At each step, we check whether we should do these two types of constructions (Y or $Z(\eta_i)$) and possibly extend the current τ to a higher one. It is nevertheless a finite list and this process eventually stops (say at τ^*). At that time, there might be some remaining unbranched 1-nodes in Y and we do not have to take care of them. There might be also some remaining η_i's which are still potential witnesses at τ^*, and we just copy $S(\tau^*)$ above η_i to the new system $S'(\tau^*)$.

We define τ^* to be the left immediate successor of τ in the new tree T and define $S'(\tau^*)$ to be the tree constructed as above. Then we do the same thing for the τ'' side. This ends one inductive step in the construction.

If the original tree system (T, S) is special, then it is easy to see that the new one (T', S') is also special: Property (0) still holds since we either copy the previous tree system, or make sure that the node to which we extend is in an old block whose root is accessible at the corresponding τ. Property (1) is true by our termination process. Property (7) holds naturally by our construction, as we carefully pick nodes in appropriate blocks at each splitting step. The other properties are immediate.

We say that Q_e *acts* at this stage. In §VI.6.5, we will verify that the requirement Q_e is (permanently) satisfied in both cases when it acts for the last time.

VI.6.4 Satisfy R_e

After we define P^e-blocks for R_e, we split the requirement R_e into infinitely many subrequirements, each asking whether we can force the values in the corresponding x_e-sequence $\langle\varphi_e^{B\oplus 0'}(x_e, s)\rangle_{s\in\omega}$ to change at least once more, and we insert these subrequirements into the priority list in any reasonable way.

Note that at this time on (T, S) with $N, P^{e_0}, \dots$ and P^{e_t}-blocks, it is not necessarily the case that these e_i's are less than e. In fact, if R_e is always active, then it has to take care of P^{e_i}-blocks for arbitrarily large e_i.

Now suppose we need to handle one of such subrequirements. We ask the same question as in §VI.5.5, i.e., whether we can find one or more change in the x_e-sequence in the accessible part of the tree system. If the answer is yes, then we take the full subtree system as before and force the values in the x_e-sequence to change at least once more. If the answer is no, then we do one of the following constructions, claim that we have satisfied the requirement R_e and declare R_e to be inactive (and then we remove all other subrequirements for the same R_e from the priority list).

(Plan I) If the final value is 1, and so we want to make B be on the e-pruned subtree, then we remove all P^e-blocks, or equivalently remove all extensions of 1-nodes and 01-nodes in all P^e-blocks. In addition, we remove all P^{e_i}-blocks for any $e_i > e$, cancel all Q and R-requirements of lower priority and reconsider them again. The reason to do so is that later we might apply Plan II for an R^{e_i} requirement and this would lead us to an inaccessible node on (T, S). So we need to avoid a lower priority requirement injuring a higher one and simply try to satisfy all lower priority requirements again.

The new tree system is special by Lemma VI.6.5.

(Plan II) If we want to make B go off the e-pruned subtree, then we first take $\mathbf{Nar}(T)$, the narrow subtree of T and let S' be the restriction of S to the range of $\mathbf{Nar}(T)$. Then we take a τ on $\mathbf{Nar}(T)$ where $S(\tau)$ has a leftmost P^e-block $\mathbb{B}$ associated with x where $x \notin A'$ if and only if A is on $\mathbf{Nar}(T)$. Let ρ be the 01-node of $\mathbb{B}$. We then take $\mathbf{FSTS}(\mathbf{Nar}(T), S', \tau, \rho)$ and remove all P^{e_i} blocks above ρ for all $e_i \geq e$ (in fact, as we will see later, we only need to remove P^e-blocks). In this process we might injure some requirements Q_i for $i > e$ that have been satisfied since the last time we redefined P^e-blocks, but for simplicity we just reconsider all requirements of lower priority as in Plan I.

First by Lemma VI.6.2, $(\mathbf{Nar}(T), S')$ is special, but we cannot directly apply Lemma VI.6.3 since ρ is not accessible at τ. In fact, ρ is never accessible at any extension of τ on $\mathbf{Nar}(T)$. Note that in this case, all of our constructions since the last time we defined P^e-blocks (say, at stage s) simply copied the tree system from the previous tree above (τ, ρ). So the system part of $\mathbf{FSTS}(\mathbf{Nar}(T), S', \tau, \rho)$ is as it was right at the end of stage s. The final tree system we get after removing all P^e-blocks is special by Lemma VI.6.5 and Lemma VI.6.2.

In both Plan I and Plan II, we say that R_e *acts*. As we have mentioned, we also cancel all lower priority satisfied Q or R type requirements or active R type subrequirements, and put them back to the priority list to their original positions.

VI.6.5 Final verifications

As a common finite injury argument, we need show that each Q_e will be eventually satisfied and never injured again, and each R_e is either always active after some stage, or it is satisfied by either Plan I or Plan II at some stage and never injured later by requirements of higher priority. First we show that every requirement eventually stops acting.

Note that Q_0 is trivially satisfied at the first step and never gets injured since it has the highest priority. Now by induction, assume that the above claim holds for every Q_i and R_{i-1} ($i = 0, 1, 2, \ldots, k$). First we show that the claim holds for R_k: find a stage s_0 by which these higher priority requirements have acted for the last time. Hence P^k-blocks never gets redefined after this stage s_0, and every requirement acting after s_0 should respect R_k. Now if R_k is not always active after s_0, we must have followed either Plan I or Plan II to (permanently) satisfy this requirement. It never gets injured since all requirements of higher priority have stopped acting.

If R_k is always active after stage s_0, then Q_{k+1} is satisfied and is never injured after s_0. If R_k acts at some stage $s_1 > s_0$ then by construction we try to satisfy Q_{k+1} after that and it then never gets injured again. This finishes the proof of the claim. It follows that the notions of e-pruned subtrees, e-witnesses, e-potential witnesses and the value of x_e are eventually fixed. We always refer to their final versions in the following arguments.

It is easy to see that B is off the (final version of an) e-pruned subtree only in the case that we follow Plan II for the requirement R_e in the end, because otherwise, either (1) we follow Plan I and then B is forced to be on the e-pruned subtree, or (2) R_e is always active after some stage, all requirements after that must respect R_e and all roots of tree systems after that stage are accessible with respect to e.

Lemma VI.6.8. *Every Q_e is satisfied.*

For every Q_e, we consider the stage s when it acted for the last time (and so it is never injured after that). The tree system (T, S) at the beginning of stage s only has N-blocks and P^{e_i}-blocks for those $e_i < e$ such that R_{e_i} is always active after stage s. This means that B is on the e_i-pruned subtree for each e_i and every node $\rho \subset B$ is eventually accessible at some $\tau \subset A$ (with respect to (T, S)).

If we got a positive answer to the key question at stage s, then we took the full subtree system of (T, S) above some (τ', ρ'). Now assuming that φ_e^B is total, we can compute φ_e^B recursively in A as follows: for any x, we search in $\mathbf{FSTS}(T, S, \tau', \rho')$ for an accessible node ρ^* at some $\tau^* \subset A$ such that $\varphi_e^{\rho^*}(x)$ converges, and output the value $\varphi_e^{\rho^*}(x)$. We must find such a convergent computation because $\varphi_e^B(x)$ converges with some use $\rho \subset B$ and we can find a $\tau \subset A$ where ρ is accessible. We cannot find different answers by the positive answer to the key question.

If we got a negative answer, then we have constructed an e-splitting subtree system (T', S'). We claim that we can compute B from $\varphi_e^B \oplus A$. We start to retrieve the construction of (T', S'), easily follow the e-splitting structure of $S'(A)$ and get initial segments of B as long as we are in the accessible part. Once we reach a 01-node ρ of a P^{e_i}-block and ρ is not accessible at the current $\tau \subset A$, we know that this ρ cannot be a witness since B is on the e_i-pruned subtree. Therefore we wait until the construction reaches some $\tau' \supset \tau$ and ρ becomes accessible at $\tau' \subset A$. At that time, according to our construction, we terminated a lot of nodes above ρ and had in hand some top-nodes which are all accessible at τ'. They do not necessarily e-split, but in the construction, we made all of their immediate successors pairwise e-split, and hence we can find the right initial segment of B using φ_e^B. Now we are back to the accessible world and we can continue the computation.

This finishes the proof that all requirements Q_e are satisfied and B is a minimal cover of A.

Lemma VI.6.9. *Every R_e is satisfied.*

For each R_e, the P^e-blocks are never redefined and x_e is never redefined after some stage. If it is always active after that stage, then it is easy to see that we have forced the x_e-sequence $\langle \varphi_e^{B\oplus 0'}(x_e, s)\rangle_{s\in\omega}$ to have infinitely many changes, and so the limit does not exist. Therefore R_e is eventually satisfied in this case.

Suppose R_e acted by following either Plan I or Plan II and is never injured after that stage. Note that on the tree system (T', S') at the end of that stage, we only had N and P^{e_i}-blocks for $e_i < e$ in the system, and by assumption these R_{e_i} will never act (i.e. they are always active) after that stage. So B is on the e_i-pruned subtree for every such e_i and every node $\rho \subset B$ is eventually accessible at some $\tau \subset A$, as before.

A node ρ is A-*accessible* if it is accessible at some $\tau \subset A$ with respect to (T', S'). In later constructions, we keep the root of any tree system A-accessible, since all these R_{e_i} requirements are always active. In particular, every $\rho \subset B$ is A-accessible.

So if we followed Plan I, then by the negative answer to the key question, we have forced every A-accessible node on (T', S') to have a limit value 1 in the x_e-sequence. It is easy to see that we have forced B to be on the e-pruned subtree $Pru_e(\bar{S}, A)$, so we have satisfied the requirement R_e.

If we followed Plan II at that stage, the argument is slightly harder than, but quite similar to, the one we used in §VI.5.5. Let (T, S) be the tree system at the beginning of the stage (note again that (T', S') is the tree system at the end of that stage). Suppose in the construction we took ρ as the 01-node in a block associated with x where $x \notin A'$ if and only if A is on $\mathbf{Nar}(T)$. It suffices to prove that no A-accessible node on (T', S') can change the limit value of the x_e-sequence from 0: Suppose not. Let $\eta \supset \rho$ be an accessible node (with respect to (T', S')) at some $\tau \subset A$ and η has one or more changes in the x_e-sequence than the root of (T, S) has. Now on (T, S), η has only one potential witness ρ below it at τ, and so by property (1) of Definition VI.6.1 it will not be terminated by any extension of τ. One can then take a $\tau' \supset \tau$ on T such that τ' is off $\mathbf{Nar}(T)$ and long enough to see that $x \in A'$ (i.e., $\varphi_x^{\tau'}(x)$ converges). Then η is accessible at τ' (with respect to (T, S)), which contradicts the negative answer to the key question we asked.

So in this case, we have forced A-accessible nodes to have limit value 0 in the x_e-sequence and it is not difficult to see that B is off the e-pruned subtree $Pru_e(\bar{S}, A)$ (since ρ as above is also a 01-node in a block associated with x in $(\bar{T}, \bar{S})$ where we defined the final version of e-pruned subtrees as in Section VI.6.1). Hence we have also satisfied the requirement R_e in this case. By our discussion in Section VI.4, B is $\overline{\mathbf{GL}_2}$. This finishes the proof of the main theorem.

Theorem VI.6.10. *There is a 2-minimal degree which is not $\mathbf{GL}_2$.*

VI.7 2-minimal $\mathbf{GH}_2$ Degree

In fact, by a small modification in our construction, we can make B be $\mathbf{GH}_2$. Therefore we are just one step away from the highest possible class $\mathbf{GH}_1$. To make $(B \oplus 0')'' \leq_T B''$, it suffices to decide whether $\varphi_e^{B\oplus 0'}$ is total by B''. So in our construction we can try to force $\varphi_e^{B\oplus 0'}$ to be total step by step, in the same way as handling the x_e-sequence: if we can do this infinitely often, then we keep B on the e-pruned subtree, otherwise when we discovered that we can no longer force $\varphi_e^{B\oplus 0'}$ to be total, we then take B off the e-pruned subtree by Plan II. There is a similar finite injury argument. In the end, by using $(B \oplus A')'$ we can decide whether B is on or off the pruned subtrees, and so we can use B'' to run the whole construction and figure out the true outcomes for the finite injury argument as well as all the coding indices for $(B \oplus 0')''$.

In Chapter VII we will modify this construction and finally get a 2-minimal degree which is $\mathbf{GH}_1$.

CHAPTER VII
2-MINIMAL $\mathbf{GH}_1$ DEGREE

VII.1 Introduction

In this chapter we combine the ideas in Chapter VI and several new ideas to build a 2-minimal degree which is $\mathbf{GH}_1$. First we briefly describe the framework for our construction.

We use a sequence of tree systems $\langle T_i, S_i \rangle_{i\in\omega}$ (not necessarily nested) to approximate two sets A and B as before, i.e., we want to make A minimal and B minimal over A. In addition, we want to guarantee that B is $\mathbf{GH}_1$, i.e., $(B \oplus 0')' \leq_T B'$. So in the end we need to argue that we can compute $(B \oplus 0')'$ from B'.

At each stage, we also construct an *admissible* path through a priority tree (as in various injury arguments of the constructions of r.e. sets). We will use α, β, γ only for nodes on the priority tree, and keep other lower case Greek letters for the nodes on the tree T or system S (and we use the same convention as in Chapter VI that τ denotes the nodes on the tree and ρ denotes the nodes on the system). We use "admissible" instead of the commonly used terminology "accessible" since we will need the notion of accessible nodes in the tree system construction. Similar to other priority tree constructions, in the end we will argue that there is a left-most path (true path) visited infinitely often (in fact, cofinitely often since we only have a finite injury argument). The minimality requirements will be satisfied along the true path, however, the $\mathbf{GH}_1$ requirements will be satisfied at nodes possibly off the true path, but will be translated to satisfaction conditions along the true path (see details in Section VII.4).

Our convention is that, the priority tree grows downwards, i.e., the highest priority node is at the top; whereas the tree systems used in the construction grow upwards, i.e., the root is at the bottom.

Recall in Chapter VI, we define a notion of accessible nodes with respect to the groups of blocks we have on the tree system, and in particular, the notion of accessible nodes is changing from one tree system to another. In this chapter, we will fix one common notion of accessibility (which is very similar to the one we used in Chapter VI) and keep it in all tree systems we are going to construct.

VII.2 Initial Tree System, Accessibility and List of Requirements

The initial tree system is exactly the same as the one we used in Chapter VI, i.e., the tree T_0 is the identity function or say the full binary tree, and the system part $S_0(A)$ tries to code A' in the 1-nodes of blocks. A node on the system is an *even-node* if it is the image of an even length string, i.e., it is a root or a top-node of a block (see Figure VI.1).

We say an even-node $\rho \supset \xi$ at $S(\tau)$ is *accessible above ξ at τ* if there is no even-node $\rho' \in (\xi, \rho]$ which is a 01-node in a block associated with x where $\varphi^\tau_x(x)$ diverges. This definition is consistent with the definition in Chapter VI that we regard every block here as a P-block there. Moreover, the definition here addresses the idea of *relative* accessibility, e.g., ξ or ρ might not be accessible at τ but there is nothing between ξ and ρ which makes ρ inaccessible, then we call ρ (relatively) accessible above ξ at τ. So in Chapter VI, the accessible nodes are accessible above the root. Here we use the same notion that a node is *accessible* if it is accessible above the root of the tree system. Note that this notion depends on the root of the tree system. We use $Acc(\tau)$ to denote the accessible nodes at τ.

We have the following list of requirements to satisfy:

- P_e: either φ_e^A is not total, or recursive, or $A \leq_T \varphi_e^A$.
- Q_e^0: either there is an x such that $\varphi_e^B(x)$ diverges, or φ_e^B is total.
- Q_e^1: if φ_e^B is total, then either it is recursive in A, or $B \leq_T \varphi_e^B \oplus A$.

Note that we split the B-minimality requirement Q_e into two requirements: Q_e^0 forces totality and Q_e^1 forces splitting.

In addition, we want to make B be $\mathbf{GH}_1$, and we do this by forcing the jump of $B \oplus 0'$.

- R_e: $\varphi_e^{B\oplus 0'}(e)$ diverges, or it converges at the root λ of the tree system, i.e., $\varphi_e^{\lambda\oplus 0'}(e)\downarrow$.

At each stage s, we have an admissible path through a fixed priority tree. Each node α is labeled by one of the requirements above and also labeled by (T_α, S_α), the tree system before we try to satisfy the requitement labeled in α.

Each node has two outcomes: For P_e, it is either a nonsplitting outcome (left), or a splitting outcome (right); Q_e^0 has nontotal (left) or total (right) outcomes, and Q_e^1 has nonsplitting (left) and splitting (right) outcomes as well; R_e has convergence (left) and divergence (right) outcomes. For convenience we arrange a priority list where each Q_e^0 is immediately followed by Q_e^1 along the right (total) outcome, and there are no Q_e^1 nodes along the left (nontotal) outcome.

At each stage we ask a key question to the lowest admissible node. If it is a P_e, Q_e^0 or Q_e^1 node, we choose the outcome according to the answer to the key question, pass a new tree system to the next node and continue to the next stage (see Section VII.3). If it is an R_e node, and if the answer (outcome) is convergence, then we take a full subtree system to force convergence and proceed to the next stage without doing anything special. If the outcome is divergence, we need to implement a so-called "narrow subtree system construction" (which is very similar to Plan II in §VI.6.4) to get a new tree system; in addition we need to go through the admissible path and check whether we have chance of switching left at any of the Q_e^0 or Q_e^1 nodes. If there is a node where we can switch left, then we do so and we will never switch back again (see details in Section VII.4).

It is not difficult to see that there is a true path through the priority tree and we will argue that the minimality requirements are satisfied along the true path; moreover we will argue that each $\varphi_e^{B\oplus 0'}(e)$ is forced at the first node we try to force it (no matter whether the node is on or off the priority tree) and the $\mathbf{GH}_1$ requirement are satisfied by a slightly tricky "reconstruction".

VII.3 Minimality Requirements

VII.3.1 Make A minimal

The P_e requirements are handled in the same way as in §VI.5.3 and we go to the outcome corresponding to the answer we get for the key question.

VII.3.2 Force totality for φ_e^B

Given Q_e^0 at α with a tree system (T_α, S_α), we ask the following key question:

$$\exists x \exists \tau \exists \rho \in Acc(\tau) \forall \tau' \supset \tau \forall \rho' \supset \rho; \rho' \in Acc(\tau')[\varphi_e^{\rho'}(x) \uparrow].$$

That is, we ask whether there is a ρ accessible at τ such that there is no accessible node above it makes $\varphi_e^{(\cdot)}(x)$ converge. If the answer is yes, then we go to the nontotal (left) outcome of α, take the full subtree system $\mathbf{FSTS}(T_\alpha, S_\alpha, \tau, \rho)$ and label it to the next node following the nontotal (left) outcome of α.

If the answer is no, then we can construct a tree forcing totality, as the negative answer tells us that at least in the accessible part of the tree system, we can always find nodes to force convergence of any $\varphi_e^{(\cdot)}(x)$. The problem is that, we are only guaranteed to find such convergence in the accessible part of the tree system, and above a potential witness, there might be no such node until it becomes a nonwitness later.

The construction here is slightly different from the ones in Chapter VI. There we simply copy everything above a potential witness, but here we regard it as a forcing totality (or forcing splitting) requirement and try to search for convergence (splittings) until we find such or the node becomes accessible (nonwitness).

Basically what we do is similar to a partial totality forcing tree construction (as in Sacks minimal degree construction, see [Sac61]) where we try to find an extension of the potential witness such that it converges at the next value x for which we want to force convergence. If we find such a node then we take the two appropriate successors (in the sense of in the correct block associated with the index we want) of that node as our next level nodes; if not then the tree above the potential witness is empty. This process of possibly partial search continues until the potential witness becomes a witness, and if it happens we will do a similar termination as in §VI.5.4 and extend the remaining nodes to force new convergence.

Adding some slight complication we are building block structures above the potential witness, so we can have some node which is a potential witness above a potential witness, etc. The basic rule is that, when we see a potential witness ρ becomes a witness, then we terminate the remaining potential witnesses above ρ and keeping tracking the status of the potential witnesses below ρ while we do the searching.

While building new blocks we need to find appropriate old blocks extending the node which already forces convergence of some x, and now it is not always the case that we can find such appropriate old blocks since the system part might be partial. One can extend the current node to the node forcing convergence first and wait there for the appropriate blocks above it to appear, or an alternative solution is to use a slightly different key question asking for an extension in an old block associated with a fixed index e. The first plan seems to destroy some of the binary tree structure, but if we opt to keep the binary tree structure, we can also add in some extra information on tree systems and modify the key question asking for nodes with certain properties to resolve the problem. The second plan works here but makes a similar problem in multiple splitting harder to handle. So for simplicity we will phrase our construction in terms of the first plan which allows us to change a little bit of the tree structure.

We omit the details of this construction here since we will do a very similar one in the following section.

VII.3.3 Force splitting for φ_e^B

multiple splitting

This is almost handled in a similar way as the totality requirements, and the only major difference is that for the multiple splitting construction (Theorem VI.6.7), we cannot wait for all these key-nodes to have a certain number of levels of splittings before we find their extensions. Alternatively we need to do pairwise splitting step by step by the following lemma. In the end, we say that we finish the multiple splitting construction. All requirements have to wait for (and copy from) the constructions of higher priority ones to finish their multiple splitting or convergence (at a node) before they try to find splitting or convergence for their own purpose or before they can use any of the nodes appearing in the multiple splitting construction.

Lemma VII.3.1. *Suppose we have a tree system which already forces totality (in the accessible part). Given two accessible nodes ρ_0 and ρ_1 (above ξ) at τ, then either of the four cases happen along any extension of τ:*

1. *some potential witness $\xi' \subset \xi$ becomes a nonwitness;*
2. *ρ_0 does not have accessible e-splitting extensions (above ξ);*
3. *ρ_1 does not have accessible extensions (above ξ) which have φ_e-image longer than some l, i.e., totality is not forced above ρ_1.*
4. *we can find accessible extensions $\rho_0' \supset \rho_0$, $\rho_1' \supset \rho_1$ such that $\rho_0'|_e\rho_1'$.*

Proof. Suppose no potential witness $\xi' \subset \xi$ becomes a nonwitness later in our search, and suppose we can find two e-splitting accessible extensions ρ_0' and ρ_0'' of ρ_0. Also suppose that we can always find accessible extensions of ρ_0 with long enough φ_e-image, then we can extend ρ_1 along the left-most path to get a node ρ_1' with φ_e-image longer than the φ_e-images of both ρ_0' and ρ_0''. At least one of them e-splits with ρ_1'. □

When we try to do multiple splitting, we start with a finite list of key nodes $\rho_0, \rho_1, \dots, \rho_k$ (after some terminations process similar to the one in §VI.5.4), and we pick the first pair ρ_0, ρ_1. Use the lemma above to wait for (1) either some potential witness below ξ turns out to be a nonwitness, in which case we terminate the multiple splitting construction and do a "bigger" multiple splitting; (2) or wait for e-splitting accessible pairs above ρ_0 to show up (if they don't show up then the multiple splitting construction stops here); (3) or if an e-splitting pair shows up, then extend ρ_1 along the left-most path to some node with longer φ_e-image (if this extension is not successful then the multiple splitting also stops here, but then we have a node forcing nontotality); (4) finally use lemma to find two extensions of ρ_0 and ρ_1 which e-split with each other, and for convenience rename them as ρ_0 and ρ_1 respectively. If we can finish this process then we pick the next pair and start to find their e-splitting extensions, and if we can finish doing this for all pairs and end up with pairwise e-splitting extensions of all these key-nodes, then we finish the multiple splitting construction and continue the normal splitting construction building blocks above these nodes.

construction details

Suppose we are given a requirement Q_e^1 at node α. We know that the previous node is a Q_e^0 node and we believe that we can force totality. Now we ask the following question:

$$\exists\tau\exists\rho \in Acc(\tau)\forall\tau' \supset \tau\forall\rho_0, \rho_1 \in Acc(\tau'); \rho_0, \rho_1 \supset \rho\neg[\rho_0|_e\rho_1].$$

It asks whether there is an accessible ρ which forces nonsplitting in the accessible part of the tree system. If the answer is yes, we take the full subtree system of (T_α, S_α) above such a pair (τ, ρ) and proceed to the next node along the left outcome of α.

If the answer is no, then we know the following:

$$(*):\ \forall\tau\forall\rho \in Acc(\tau)\exists\tau' \supset \tau\exists\rho_0, \rho_1 \in Acc(\tau'); \rho_0, \rho_1 \supset \rho[\rho_0|_e\rho_1].$$

That is, starting from an accessible node we can always find e-splitting accessible extensions. This allows us to build a splitting subtree. However, in the inaccessible part (above a potential witness) we might not have chance to find splittings, and so we simply wait for splittings to appear (as we described in the forcing totality case).

For convenience say we are building (T, S) a splitting subtree of (T_α, S_α) and at stage s we have already built a subtree T of T_α up to level s, and for each τ on T up to level s we have $S(\tau)$ a subtree of $S_\alpha(\tau)$.

Suppose we are given one τ and the corresponding system $S(\tau)$. We want to find two extensions of τ and their corresponding systems. These two extensions are extending respectively the left and right immediate successors of τ on T_α and the constructions for finding them and defining the system part are essentially the same. We only describe the construction on one side.

By induction hypothesis, $S(\tau)$ is a finite tree with the following information: It consists of blocks level by level, and each block is associated with some index. The roots of these blocks are all accessible above the root at τ. Some of the blocks are full and others are not, i.e., the corresponding computation still diverges. In addition, some nodes are potential witnesses and above these nodes, the tree looks like a "miniature" of some tree structure similar to $S(\tau)$: in the sense that above a potential witness ρ, we also have the tree system with blocks and the roots of these blocks are accessible above ρ; we also see potential witnesses and above these potential witnesses we again see a miniature of this kind of tree structure. One can think of a *matryoshka doll*: a smaller doll which lives inside a bigger one looks similar to the bigger one but it is not exactly the same as the bigger one, and there are possibly even smaller ones inside the small doll. This chain of dolls (potential witnesses and miniatures) will eventually end. Interestingly, we always know that the biggest doll exists, but we don't know that the other dolls exist until we open the previous level dolls and see them, and the same happens here for $S(\tau)$: the first level miniature is always total, but the remaining ones might be partial until we go above the corresponding potential witnesses and find out that they are total.

In addition, at each level of miniatures, some of the nodes may be previously potential witnesses (and had its own miniature), but they are no longer witnesses at τ. Then we have done several levels of multiple totality or multiple splitting constructions for these nodes (by some of the requirements above α and also R_e^1 at α), and such construction destroys the corresponding miniature, but only up to that level, in other words, such multiple totality or splitting construction can also happen inside a miniature. Some of such multiple constructions have finished, some have not. In the construction we need to make sure that the previous multiple constructions have finished before we can start new multiple splitting construction for our requirement Q_e^1, at any level of miniatures.

By induction hypothesis, this $S(\tau)$ is e-splitting at each level of miniatures, and we need to continue building this e-splitting structure. At the first level of miniature, the construction is the same as Lemma VI.6.6, i.e., we use $(*)$ to search for extensions of τ where we can find appropriate blocks extending the current ones. At other levels of miniatures we simply do a partial tree construction: that is, we wait for splittings to appear but not search for them.

Some of the potential witnesses might become a nonwitness and so we have done some termination process and started some multiple constructions. If some higher priority requirement has not finished its multiple splitting, then Q_e^1 is only going to copy the tree without any modification. If every higher priority requirement has finished its multiple totality or splitting construction, then Q_e^1 starts its own multiple splitting construction right from where higher priority requirements finish their construction. The construction proceeds as what we described before. The most outer-layer level multiple splitting is total, i.e., we search for nodes to finish it; and all other level multiple splitting constructions are partial, i.e., we wait for splittings or appropriate nodes to appear.

If we haven't done a termination process, then we first need to do some termination and get a list of key-nodes to do multiple splitting. Starting from the potential witness ρ (which now becomes a nonwitness), we already have some partial tree structure and in particular some even-nodes are waiting for splitting extensions to appear. We first extend all these even-nodes by a full tree of $S_\alpha(\tau')$ (suppose we are now at τ'), i.e., copy whatever is in $S_\alpha(\tau')$ above these nodes to the new system. Then we terminate all the following nodes (see also §VI.6.3): all 1-nodes that have not yet branched; all nodes that still have potential witnesses between ρ and these nodes; and all previously terminated nodes. In the end we are left with a finite list of *key-nodes* that are accessible above ρ at τ'. Then we can start our multiple splitting process as above.

This finishes the splitting construction at the inductive step. In the end, we go to the right outcome of α with the splitting subtree system (T, S).

VII.4 Force the Jump of $B \oplus 0'$

Given R_e at a node β with a tree system (T_β, S_β), we ask:

$$\exists\tau\exists\rho \in Acc(\tau)[\varphi_e^{\rho\oplus 0'}(e)\downarrow].$$

If the answer is yes, we take the full subtree system above this (τ, ρ) and continue to the left (convergence) outcome of β. If the answer is no, then we go to the right (divergence) outcome and do the following *narrow subtree system construction* (which is very similar to the Plan II in Chapter VI). Let S'_β be the restriction of S_β on the range of $\mathbf{Nar}(T_\beta)$. Find a block (at some τ) along the left-most path associated with index x where $x \in A'$ if and only if $A \in [\mathbf{Nar}(T_\beta)]$. Take the 01-node ρ of this block. Then take the full subtree of $(\mathbf{Nar}(T_\beta), S'_\beta)$ above (τ, ρ). For convenience we call this new tree system (T'_β, S'_β). In this case we claim that we have forced $\varphi_e^{(\cdot)\oplus 0'}(e)$ to diverge in the accessible part of the tree system (we denote this claim by C_e). We will show, in the verification, that C_e is true for the new subtree system we have and in fact true for all tree systems we will construct later. Interestingly, the satisfaction of C_e will be automatically passed along the construction if we go down or go to the left on the priority tree.

We need to do some additional queries here. It is easy to see that from the point of view of all previous requirements, the nodes on the new subtree system are not accessible (since they are above ρ and $\varphi_x^{(\cdot)}(x)$ diverges) and all the key questions we asked may fail to work. For some of them we can argue that the forced divergence or nonsplitting is still true, but for some others we need to check whether there is a chance of switching to the left of the priority tree. In general we opt to switch to left. The reason we can switch to left is as follows: note that ρ is inaccessible, and so for splitting constructions above it, we only search for splitting extensions but are not guaranteed that we can find such. Therefore it is possible above some node we cannot find splitting extensions and so by taking the full subtree above that we can force nonsplitting and hence go to the left.

We need the following lemma:

Lemma VII.4.1. *For every accessible ρ' at τ' on the new subtree system in the R_e-divergence construction above, and for every tree T labeled on the nodes above β, there exists $\tau'' \supset \tau$ on T such that ρ' is accessible at τ''. In short, all accessible nodes on the new tree system was previously accessible at a different node.*

The proof is essentially the same as the argument in §VI.5.5. Since the new tree we get is a narrow subtree, it is always possible to find an extension τ'' to get off the tree. Then it is easy to see, from the choice of the associated index x, ρ becomes a nonwitness at τ''. Therefore ρ' becomes accessible.

So we go along the admissible path and check each node one by one. If it is a $P_{e'}$ node, then we don't have to do anything, since the tree part is always nested. If it is an $R_{e'}$ node, again we don't have to do anything: if we previously forced convergence then it is of course preserved; if we previously forced divergence, then no accessible node here on the new tree system can make convergence since all of them are accessible on the old one (at a different node).

If we have a $Q^0_{e'}$ or $Q^1_{e'}$ node, and if we previously went to left (nontotal or nonsplitting) then again by the lemma above we do not have to anything, as the nontotality or nonsplitting is still forced on the accessible part of the new tree system. If we previously went to the right node, then we need to check whether above this inaccessible ρ we have a chance of forcing nontotality or nonsplitting. Take $Q^1_{e'}$ (node α) as an example (the question for $Q^0_{e'}$ is a natural analog and so omitted here). Recall that (T_α, S_α) is the tree system labeled at α, i.e., the tree system before we do the construction at α, and (T'_β, S'_β) is the new tree system we get from the narrow subtree system construction. Since we are dealing with two or more tree systems here, it is better to make it clear which tree system we are talking about when we address accessibility. We use $Acc_S(\tau, \rho)$ to denote the set of accessible nodes above ρ at τ with respect to the system S, and so it is a subset of nodes on $S(\tau)$. If we omit ρ then we mean the set of accessible nodes above the root of $S(\tau)$. We ask the following:

$$\exists \tau \text{ on } T'_\beta \exists \rho \in Acc_{S'_\beta}(\tau) \forall \tau' \supset \tau \text{ on } T'_\beta \forall \rho_0, \rho_1 \supset \rho; \rho_0, \rho_1 \in Acc_{S_\alpha}(\tau', \rho) \neg[\rho_0 |_{e'} \rho_1].$$

The intuition is as follows. We look at the current tree T'_β and we ask whether there is an accessible node ρ on the current system S'_β such that the world above ρ with respect to the new tree T'_β and the old system S_α has no e'-splittings.

If the answer is yes, then we can switch to the left at this $Q^1_{e'}$ node α by taking the full subtree system of (T'_β, S_α) above (τ, ρ) as in the key question. This forces nonsplitting in the accessible part of the new tree system. We will check that all previously satisfied P or R requirements will be automatically carried over to the left outcome, and for the Q requirement below the right outcome of α, we can satisfy them again below the left outcome.

If the answer is no, then we know that the e-splitting structure above the root is still working on the tree system (T'_β, S'_β) in the accessible part, as accessible nodes above ρ still have accessible e'-splittings. So we do nothing here and continue to check the next Q node.

We continue on this checking procedure and end up with either switching to the left or staying at the same node β. If we finally end at the same node β, then we can do the following construction to make the system part more "structured".

Note that the answers from the checking procedure guarantee that, for every accessible node on the system S'_β, it has accessible extensions, but it is not guaranteed that such accessible extensions exist along every path through T'_β. Now we can build a new subtree system to force extensions in the accessible part, by simply thinning out the tree and make sure that accessible nodes always have accessible extensions along every path. Briefly, at the inductive step, given τ, we find two extensions τ' and τ'' where all accessible nodes on $S'_\beta(\tau)$ have accessible extensions. Finally the tree part is a subtree of T'_β and the system part is a restriction. In the language of dolls, we are simply making the most outer-layer "doll" total.

This construction is not crucial and might not be necessary, if the next construction involves system level constructions, then it will automatically do this construction for us. If we choose not to do it, then if the next construction is a tree level construction (such as a splitting tree construction), then we need to go over the checking procedure again to see whether we have chances of switching to the left at some Q node again.

This finishes the construction at an R node.

In the whole construction, we can make sure that the only case we allow a witness ρ to be the root the new system is in this R-divergence construction. All other constructions use accessible nodes as the root. So in the end, each witness along B corresponds to one R-divergence narrow subtree system construction. This allows us to read this coded information and retrieve the construction by only using $B \oplus A'$.

VII.5 Final Verifications

It is not difficult to see that there is a true path via a finite injury argument. The P type of requirements are satisfied when we try to satisfy them at the first time, since the tree parts of tree systems we build are nested.

Given a requirement Q_e, we first check the unique Q_e^0-node on the true path (say α): if the true outcome at α is to the left, then we have forced divergence of a fixed $\varphi_e^{(\cdot)}(x)$ in the accessible part. By Lemma VII.4.1 and our construction we know that all roots of the tree systems extending α's left outcome are accessible on the tree system (T_α, S_α), so the divergence has been forced and the requirement Q_e is satisfied. In this case there are no Q_e^1 nodes on the true path.

If the true outcome at α is to the right, then we know that totality has been forced, i.e., we are working on a tree system which forces totality along the path B we build. Then we check the true outcome of Q_e^1 node α' following α. If the true outcome is to the left, then we know that (T_γ, S_γ) forces nonsplitting in the accessible part where γ is the node immediately extending the left outcome of α'. We can argue that in the end, if φ_e^B is total, then it is recursive in A: to compute $\varphi_e^B(x)$ we search on $S_\gamma(A)$ an accessible node ρ at some $\tau \subset A$ (accessible with respect to (T_γ, S_γ)) such that $\varphi_e^\rho(x)$ converges. There must be such a node since totality has been forced, and we claim that this value is the same as $\varphi_e^B(x)$. Suppose not, then we can find an initial segment τ' of A extending τ long enough to have nodes ρ and some $\rho' \subset B$ on $S_\gamma(\tau')$ where the corresponding values of $\varphi_e^{(\cdot)}(x)$ are different. By Lemma VII.4.1 and our construction, each ρ' an initial segment of B is accessible at a node $\tau'' \supset \tau'$ (τ'' is usually not an initial segment of A); ρ is accessible at τ and so at τ''; this contradicts that nonsplitting has been forced in the accessible part of (T_γ, S_γ).

If the true outcome at α' is to the right, then we know that the e-splitting structure always works along B and we can compute B from $A \oplus \varphi_e^B$ quite easily by following the splitting. The system is e-splitting at first, then when we see a potential witness, either it is a witness or it will become a nonwitness later, but in either case we can search above it for e-splitting pairs. Either we find such an e-splitting pair, or we see that the potential witness becomes a witness and so do multiple splitting. We can always follow the splitting structure to find initial segments of B by $A \oplus \varphi_e^B$.

So the minimality requirements are all satisfied and we still need to show that B is $\mathbf{GH}_1$, i.e., $(B \oplus 0')' \leq_T B'$. In fact we can show that $(B \oplus 0')' \leq_T B \oplus A'$, and this implies that B is in $\mathbf{GL}_1(A)$.

Note that our construction is a $0''$ construction and here we can compute the whole construction by $B \oplus A'$. The key fact here is that with B and A' we can tell which nodes along B are witnesses: they are exactly the ones that are 01-nodes on blocks that are associated with index e such that $\varphi_e^A(e)$ diverges. In the construction, these witnesses are exactly the ones coming from the narrow subtree system construction in Section VII.4 chosen as codings of whether $A \in [\mathbf{Nar}(T)]$ for some T. In addition, from the index x associated to the block of a 01-node witness we can effectively retrieve the correct $0''$ construction, since the index x codes the information of $\mathbf{Nar}(T)$ as being the iterations of (P requirements) full subtree constructions, splitting subtree constructions; (Q requirements) full subtree system constructions, totality subtree system constructions, splitting subtree system constructions; and (R requirements) narrow subtree system constructions, possibly with some recognizable padding added (as we used the Padding Lemma in the construction). This means that $B \oplus A'$ can retrieve the original $0''$ construction.

Now it suffices to show that each requirement R_e is satisfied when we first try to satisfy it at some β (not necessarily on the true path) as in Section VII.4. If we go to the convergence outcome of β then the requirement is of course satisfied. If we claim to force divergence then we generate a claim C_e which says that "$\varphi_e^{\rho\oplus 0'}(e)$ diverges for any accessible ρ on the current tree system". We will prove by induction that this C_e is carried over to all following tree system we construct. The base case is that C_e is true on (T'_β, S'_β) (the tree system we get by the narrow subtree system construction in Section VII.4, see Lemma VII.4.1).

In the construction we only go down or go to the left of the current node. If we go down, in most cases when the new root is accessible on the old tree system, the accessible nodes on the new tree systems are still accessible on the old one. The only exception is the narrow subtree system construction, when the new root is not accessible, but Lemma VII.4.1 is then used to show that accessible nodes on the new tree system are accessible on the old one.

If we go to the left, then it must be the case when we do a narrow subtree system construction for an R node $\tilde{\beta}$ and go left at the checking process as in the construction. We can use the following lemma to finish the proof:

Lemma VII.5.1. *Suppose we switch to the left outcome at a Q node α because of the narrow subtree system construction at an R node β, also suppose the full subtree system we get after switching left at α is (T'_α, S'_α), then all nodes accessible on (T'_α, S'_α) are accessible on (T_β, S_β).*

Proof. Let (T'_β, S'_β) be the tree system we get from the narrow subtree system construction. Let us assume that α is a Q_e^1 node (and the proof for Q_e^0 is essentially the same). In the construction, we asked the following question at α:

$$\exists \tau \text{ on } T'_\beta \exists \rho \in Acc_{S'_\beta}(\tau) \forall \tau' \supset \tau \text{ on } T'_\beta \forall \rho_0, \rho_1 \supset \rho; \rho_0, \rho_1 \in Acc_{S_\alpha}(\tau', \rho) \neg[\rho_0 |_e \rho_1].$$

Since we switched to the left at α, we must had a yes answer, i.e., there is a τ and a ρ accessible at τ on S'_β such that there is no e-splitting accessible pair above ρ on S_α at any $\tau' \supset \tau$ on T'_β.

This implies that, in the e-splitting subtree system construction along the right outcome of α, we cannot find any e-splittings for $\tau' \supset \tau$ on T'_β, i.e., the system part extending ρ is empty.

Note that the new subtree system (T'_α, S'_α) along the left outcome of α is the full subtree system of (T'_β, S_α) above this (τ, ρ). Now given an accessible node ξ

at τ' on (T'_α, S'_α), we can find an extension τ'' of τ' on T_β and τ'' is off T'_β (this is possible because T'_β is a subtree of the narrow subtree of T_β). Then at τ'' we see that ρ becomes a nonwitness and so accessible, so according to the splitting tree construction (Section VII.3) we start a multiple splitting construction above ρ. It is currently empty above ρ, and so we simply copy everything from S_α and do termination and multiply splitting. In particular, ξ is accessible at τ'' on S_α and so it will not be terminated in the construction. All following constructions will follow the multiple splitting first and so ξ is still accessible at τ'' on S_β. $\square$

By the above lemma, switching left does not change the accessibility of nodes. So the claim C_e will be carried over if we switch to the left at some α for the narrow subtree system construction at $\tilde{\beta}$. Combining the arguments above, we know that $(B \oplus 0')'(e)$ is forced the first time when we try to satisfy R_e, so $(B \oplus 0')' \leq_T B \oplus A' \leq_T B'$, i.e., B is $\mathbf{GH}_1$.

CHAPTER VIII
ITERATED FPF AND MINIMALITY (JOINT WITH GREENBERG)

VIII.1 Introduction

We present here a preliminary version of some joint research with Greenberg. Our main goal is to construct an initial segment of the Turing degrees, $\mathbf{0} = \mathbf{a}_0 < \mathbf{a}_1 < \cdots < \mathbf{a}_n < \ldots$, an ω-chain of degrees where each $\mathbf{a}_{i+1}$ is a strong minimal cover of $\mathbf{a}_i$, and each $\mathbf{a}_{i+1}$ is relatively **FPF** over $\mathbf{a}_i$, i.e., there is a function recursive in $\mathbf{a}_{i+1}$ which is DNR relative to $\mathbf{a}_i$.

The motivation comes from reverse mathematics. There we want to study the relative provability strength of sentences and systems. For example, does the existence of a **FPF** degree imply that there are incomparable Turing degrees? If we can find such an ω-chain as planned, then the answer is no (see [Sim10]). In this chapter we use our established framework of tree systems to handle "two steps", i.e., a maximal chain $\mathbf{0} < \mathbf{a} < \mathbf{b}$ where $\mathbf{a}$ is **FPF** and $\mathbf{b}$ is relatively **FPF** over $\mathbf{a}$.

The "bushy" tree structure was introduced in [KLxx] to construct a **FPF** minimal degree, i.e., the "first step". Here we first give a simplification of their bushy tree construction and use the similar idea to do our "second step".

VIII.2 Kumabe-Lewis Construction: A Simplification

VIII.2.1 Basic set-up

A *tree* is a partial function from $\omega^{<\omega}$ to $\omega^{<\omega}$. In this chapter we only consider *finite-branching* trees, i.e., for each τ on the tree, there are only finitely many i's such that $\tau * i$ is on the tree. We will only use recursive trees with recursive domain.

A tree T is *f-bushy* for some function f if for each nonterminal node τ of length n on the tree, τ has at least $f(n)$ many immediate successors and all of its immediate successors are of the form $\tau * i$ for some i, i.e., one bit extensions of τ.

A tree T *admits* (g, h) if T is g-bushy and there is no h-bushy finite subtree S of T such that every S-terminal node τ is also a T-terminal node. Intuitively, if T admits some pair (g, h) and $g(x) > h(x)$ for every x, then T is bushy while the terminal nodes are not very bushy.

We use the following notions: f^+ denotes the function $2f$ and f^- denotes the function $f/2$; we will also use notions like f^{++} or f^{+2}, or f^{+l} [$f^{+l}(n) = 2^{l(n)} f(n)$]. Given functions g and h with $g > h^{+l}$, we will use a function which is *in the middle* of g and h, i.e. something like $\sqrt{gh}$, or we can also use h^{+l^-}. That is, we pick a function which is "far away" from both g and h.

In the construction, we build a nested sequence of trees $\langle T_i \rangle$ such that each T_i admits (g_i, h_i). For each i, g_{i+1} is dominated by g_i, and h_{i+1} dominates h_i. In addition, each $g_i \geq h_i^{u_i}$ where $u_i(n)$ is a recursive function related to the number of nodes of length n. So at each step, we increase h and decrease g, but still guarantee that they are far apart from each other.

We can guarantee that, before the values of $g(l)$ and $h(l)$ collapse, the root of the tree has already grown up to level l, so we will not encounter the case that $g(l) < h(l)$. To note, the functions g_{i+1} and h_{i+1} will depend on the previous pair (g_i, h_i) and the answers to the questions we ask in the construction.

VIII.2.2 Initial tree

Given a recursive function g_0, we define the initial tree T_0 as follows: First put the empty string $\emptyset$ into the domain and define $T_0(\emptyset) = \emptyset$; at stage s, for each unterminated unbranched σ in the domain, we put $\sigma * 0, \sigma * 1, \ldots, \sigma * (g_0(|\sigma| - 1))$ in the domain and define T_0 as the identity function on these strings; in addition, we check whether some $\varphi_e(e)$ converges at stage s for some $e < s$: if so we terminate all nodes on the tree which are not DNR witnessed by such $\varphi_e(e)$ (i.e., all τ such that $\tau(e) = \varphi_e(e)$).

In this definition, we use a recursive function g_0 which we will specify later. Intuitively, g_0 has to grow fast enough such that after n steps in the construction, $g_n(n)$ is still sufficiently greater than $h_n(n)$. Note that h_0 is the constant 2 function here since each $\varphi_e(e)$ can only have one value.

VIII.2.3 Force totality

Given T which admits some (g, h), we want to force the e-th minimality requirement, i.e., for each $A \in [T]$, φ_e^A is either nontotal, or recursive, or computes A.

Fix k in the middle of (g, h). We call a node τ on T *accessible* if there is no finite subtree S above τ which is k-bushy and every S-terminal node is terminal on T. For convenience, we will call S *terminated* if every S-terminal node is terminal on T. Given a subtree S, a *top-node* on S is a terminal node on S which is not a terminal node on T. So terminated subtrees do not have top-nodes.

The following lemma will be useful in our constructions.

Lemma VIII.2.1. *Given a finite g-bushy tree S and color all the terminal nodes on S by two colors, then there is a g^--bushy subtree S' of S which is homogenous, i.e., all terminal node on S' have the same color.*

Proof. We color the other nodes of S inductively by the majority color of its immediate successors. That is, given a nonterminal node τ, by bushiness it has at least $g(|\tau|)$ many immediate successors. By induction, every immediate successor has been colored by either of the two colors, and then we can color τ by the majority color of its immediate successors. Then by picking the nodes with the same color as the root of S we get a subtree S' which is g^--bushy. □

This lemma is widely used in the following way: in the construction we may terminate some nodes and we want to make sure that we are not terminating too many nodes. What we do is to make sure that the previous terminations do not terminate some h-bushy many nodes (by induction), and in the new termination we are not terminating h-bushy many nodes; then by the above lemma we know that combining them together we are not terminating nodes that are h^+-bushy.

Now we ask whether there is an x and an accessible node τ on T such that there is no k-bushy finite subtree S above τ such that every top-node τ' on S has $\varphi_e^{\tau'}(x)\downarrow$.

If the answer is yes, then we take the full subtree of T above such τ and terminate all the nodes τ' which has $\varphi_e^{\tau'}(x)\downarrow$. It is easy to see that we are not terminating k-bushy many nodes for this reason and so combining with the old terminal nodes we are not terminating k^+-bushy subtrees. In this case, we have satisfied this requirement and process to the next step with this new tree admitting (g, k^+).

If the answer is no, then we do the following construction to force the totality of φ_e. Note that the negative answer says that for every τ, either it is not accessible, i.e., we can find a k-bushy terminated subtree above it, or we can find a k-bushy subtree above it where every top-node τ' has $\varphi_e^{\tau'}(x)\downarrow$. So we proceed our construction starting from the root searching for k-bushy subtree of either type: given any τ on the new tree which has not been branched or terminated, we search above τ on the old tree T for a k-bushy finite subtree S such that either S is terminated, or all top-nodes on S have convergent $\varphi_e^{(\cdot)}(x)$ where $x = |\varphi_e^{\tau}|$. It is easy to see that our new tree admits (k, h) since all terminated nodes are old (i.e., they are already terminated on the old tree). In addition, every infinite path A on the new tree has total φ_e^A, i.e., we have forced totality. For convenience we replace the old tree T with the new one and still call it T.

VIII.2.4 Force splitting

Following the construction above, we have a tree T admitting (k, h) and it forces totality. We want to force splitting as in minimal degree constructions, but as discussed in [KLxx], it is not possible to construct such a minimal degree using direct splitting trees since it will automatically produce non-**FPF** degrees. So what we shall expect for is a "delayed splitting" tree. Our notion of trees is slightly different from that in [KLxx], but our delayed splitting construction is very similar to their construction. In particular, the key ideas are the same: given two incomparable nodes τ_0 and τ_1, we cannot expect that they e-split, but we can wait for a level l such that all extensions of τ_0 at level l pairwise e-split with all extensions of τ_1 at level l, i.e., if $\tau_0' \supset \tau_0$, $\tau_1' \supset \tau_0$ both are long enough (above level l), then they e-split.

Let k' be in the middle of (k, h). Note that now the notion of accessible nodes has changed from the old tree: a node is accessible if there is no k'-bushy terminated subtree above it. We ask the following: is there an accessible node τ such that there exist no pair of k'-bushy subtrees S_0, S_1 above τ where all top-nodes on S_0 pairwise e-split with top-nodes on S_1 (we denote this by $S_0|_e S_1$).

If the answer is yes, we pick the full subtree above such τ, and do a termination process as in [KLxx] to make sure that all remaining nodes give the same answer for each $\varphi_e^{(\cdot)}(x)$. The tree is still k-bushy and the terminated nodes are not k'^+-pushy, i.e., the new tree admits (k, k'^+). The key reason that we do not terminate too many (bushy) nodes is that, the terminated nodes produce a different $\varphi_e^{(\cdot)}$ value compared to the nonterminated ones (which is of course bushy enough). So if at some step we discover that we terminate too many (bushy many) nodes, then we can get two bushy e-splitting subtrees, which contradict the yes answer.

If the answer is no, we do a similar delayed splitting tree construction as in [KLxx, Section 6]. The splitting tree we get admits (k^{l-2u}, h) where $u(l)$ is the number of nodes of length l. See also our delayed tree system construction in §VIII.3.4.

In summary, each construction will reduce the "distance" of the admissible pair (g, h) by some amount (bounded by $2u$), and on our initial tree, we only need to guarantee that g is fast-growing enough that for each l, by l times of the construction as above (in the worst case, i.e., the splitting subtree construction), one still has $g_l(l) \geq 2h_l(l)$, then we can process the plan without a problem.

To note, in order to make sure that at each step we extend the root of the tree, we need to take a one-bit extension of the root if we haven't extended it. One can always find a suitable one-bit extension such that the corresponding full subtree admits the same pair, since otherwise the old tree would not admit the pair (g, h).

VIII.3 Work with Tree Systems: 2-minimal

VIII.3.1 Tree systems

Our notion of tree systems mainly follows that in Chapter VI, i.e., a tree system (T, S) has two parts: the tree part T is a tree and the system part S maps the nodes on the tree T to finite trees with some "coherence" property. However, since we need to have bushy trees and systems, it is much more convenient to use finitely branching trees instead of binary branching trees.

The bushiness notion is similarly defined for the system part. Note that in the system part $S(\tau)$, the even positions code information from τ, so the bushiness is defined correspondingly: $S(\tau)$ is p-bushy if every nonterminal node ρ of length $2n$ has at least $p(n)$ many immediate successors, each of the form $\rho * \tau(n) * i$. One can also think of it as a bushy tree in the first construction after removing all even position codings. A tree system (T, S) is (g, p)-bushy if T is g-bushy and each $S(\tau)$ is p-bushy.

We call (Γ, Δ) a finite tree system above (τ, ρ) if Γ is a finite tree above τ and for each terminal node τ' on Γ, $\Delta(\tau')$ is a finite tree above ρ. Such a finite tree system is (h, q)-bushy if Γ is h-bushy above τ and each $\Delta(\tau')$ is q-bushy above ρ. Given a tree system (T, S) and a finite subtree system (Γ, Δ), note that a *top-node* of Γ is a terminal node on Γ which is not a terminal node on T; similarly a *top-node* on the finite tree system (Γ, Δ) is a top-node on some $\Delta(\tau)$ where τ is a top-node on Γ. In another words, top-nodes are these top level nodes which haven't been terminated (and which may or may not be terminated by later constructions along different paths). We call (Γ, Δ) *terminated* if it does not have top-nodes, i.e., every (Γ, Δ)-terminal node is also a (T, S)-terminal node.

One say that a tree system (T, S) *admits* (g, h, p, q) if and only if it is (g, p)-bushy and there is no terminated (h, q)-bushy finite subtree above the root. We call such (g, h, p, q) an *admitting quadruple* for the tree system. In a similar way, we will guarantee that in the construction, (T_i, S_i) admits a quadruple (g_i, h_i, p_i, q_i) that satisfies some properties which allow us to continue the construction. We can even guarantee that for each tree system (T_i, S_i), T_i admits (g_i, h_i) and each $S_i(A)$ admits (p_i, q_i) for each $A \in [T_i]$ (note that this directly implies that (T_i, S_i) admits (g_i, h_i, p_i, q_i)).

The following lemma is essential in our argument.

Lemma VIII.3.1. *Suppose* (Γ, Δ) *is* (g, h)*-bushy above* (τ, ρ)*, and we color the terminal nodes of the tree system by two colors, then there is a subtree system* (Γ', Δ') *which is* (g^-, h^-)*-bushy above* (τ, ρ) *and which is homogenous, i.e., every terminal node has the same color.*

Proof. Given a terminal node τ on Γ, $\Delta(\tau)$ is h-bushy, and we can apply Lemma VIII.2.1 to get a homogenous h^--bushy subtree $\Delta'(\tau)$ of $\Delta(\tau)$. We then color τ by the color of the nodes on the homogenous subtree we get. This gives a coloring of the terminal nodes on Γ, then we can apply Lemma VIII.2.1 on Γ to get a homogenous g^--bushy subtree Γ'. It is easy to see that (Γ', Δ') is the subtree system we want. $\square$

This lemma will be used mainly in the following case: if we terminate nodes for two reasons and for each reason we are not terminating a (g, h)-bushy tree system, then combining them together we are not terminating a (g^+, h^+)-bushy tree system.

VIII.3.2 Initial tree system

As in the previous section, we pick our g_0 and p_0 to be fast-growing enough in the construction of the initial tree system (T_0, S_0) to make it (g_0, h_0)-bushy. On the tree, we terminate a τ on the tree if it is not DNR. On the system, note that each ρ on $S_0(\tau)$ is $\tau \oplus \rho'$ for some ρ', and we want to make sure that ρ' is DNR in τ, so we terminate a ρ on the system at τ if such ρ' is not DNR in τ. It is easy to see that h_0 and q_0 are both constant 2.

VIII.3.3 Force totality

We will need the following accessibility notion. Given a tree system (T, S) which admits (g, h, p, q) and let (k, l) be in the middle of (g, p) and (h, q), and given a ρ on some $S(\tau)$, we say that (τ, ρ) is *accessible* if there is no (k, l)-bushy terminated finite subtree system above (τ, ρ). For example, this implies that τ is DNR, there is no k-bushy subtree above τ which is terminated, and there is no k-bushy subtree Γ above τ such that ρ is terminated at each terminal node on Γ, etc. We always pick the root of the new tree and the root of the new system which form an accessible pair in the old tree system.

To force totality, we ask the following: whether there is an accessible pair (τ, ρ) and an x such that for any k-bushy tree Γ above it, there is always a τ' a top node on Γ such that the ρ''s on $S(\tau')$ that the set of nodes that $\varphi_e^{\rho'}(x) \downarrow$ does not contain a subset which is l-bushy above ρ, or in other words, we ask whether there is an accessible (τ, ρ) and an x such that there is no (k, l)-bushy tree system above (τ, ρ) where every top-node converges at x.

If the answer is yes, then we take the full subtree above such (τ, ρ) and terminate (in addition to those we have already terminated) all ρ' such that $\varphi_e^{\rho'}(x)$ converges. It is easy to see that we are not terminating a (k, l)-bushy tree system for this reason, so the new admitting quadruple (g', h', p', q') is (g, k^+, p, l^+).

If the answer is no, then we construct a tree system forcing totality. It is easy to see that by the negative answer to the question, for each accessible pair (τ, ρ) and each x we can always find a (k, l)-bushy system above it which makes $\varphi_e(x)$ converge. By iterating this (see next paragraph) we can construct a subtree system which forces totality. All terminations are old (i.e., we do not make new terminations). The new admitting quadruple is (k, h, l, q).

The construction is as follows: we start from the root of the tree and the root of the system: this pair is accessible, so we can find a (k, l)-bushy tree system (Γ, Δ) above it where every top-node converges at 0. Let (Γ, Δ) be an initial part of our new tree system. At the next step, say $\tau_0, \ldots, \tau_n$ are the top-nodes of Γ and at each τ_i, we have $\rho_{i0}, \ldots, \rho_{im}$ as top-nodes on $\Delta(\tau_i)$. Next we try to find extensions above (τ_0, ρ_{00}): if it is accessible, then we can find a (k, l)-bushy tree system as its extension where every top-node converges at 1; if not, then by the definition of accessibility we can find, also a (k, l)-bushy tree system but that every terminal node there is terminated on the old tree system, i.e., a terminated finite tree system, and we also take it as our extension for (τ_0, τ_{00}). Then we have to find extensions for (τ_0', τ_{01}) where τ_0' is the first top-node on the tree part of the tree system we find extending (τ_0, τ_{00}), either forcing convergence at 1 or forcing termination. Following that we try to handle (τ_0'', τ_{01}) where τ_0'' is the second top-node following τ_0'. Nevertheless such recursive search will stop as either one of these two types of subtree systems has to appear, and so by iterating this procedure finitely many times we can find a tree system extending (Γ, Δ) such that every top-node converges at 1. It is easy to see how to handle the whole construction inductively and the new tree system is (k, l)-bushy and also forces totality. All terminations are from the old tree, so the new admitting quadruple is (k, h, l, q).

VIII.3.4 Force splitting

Following the second case above (and for convenience we still use (T, S) to denote the new tree system), we try to force splitting. Let (k', l') be in the middle of (k, l) and (h, q); Note that our accessibility notion has changed to the one with respect to (k', l') as we are in a new tree system.

To make B a strong minimal cover of A (as compared to only a minimal cover), we need a (delayed) full splitting subtree system (T', S') of (T, S), i.e., for any two incomparable nodes ρ_0, ρ_1 on the whole tree system, we can find a level l where all extensions of ρ_0 pairwise e-split with all extensions of ρ_1 at that level l.

We ask whether there is an accessible (τ, ρ) such that for all k'-bushy subtree Γ above τ, there is a top node τ' on Γ such that there is no pair of l'-bushy subtrees Π_0, Π_1 of $S(\tau')$ above ρ such that $\Pi_0|_e\Pi_1$, i.e., the top-nodes of Π_0 pairwise e-split with the top-nodes of Π_1.

If the answer is yes, then we first take the full subtree system above such (τ, ρ), and along each $S(A)$ we do something similar as in [KLxx] or in §VIII.2.4 to terminate those nodes in the minority part with respect to the value being forced (by the totality forcing tree). In the end, we cannot terminate a (k', l')-bushy subtree system for the new reason because it would contradict the yes answer of our question (and so the new tree system admits (k, k'^{+}, l, l'^{+})). See argument in §VIII.3.4.

If the answer is no, then we know that for any accessible node (τ, ρ) we can always find a finite k'-bushy tree Γ above τ such that for every top node τ' of Γ we can find two pairwise e-splitting l'-bushy Π_0, Π_1 on $S(\tau')$. This allows us to do a (delayed) splitting tree construction. Similarly if we find out that the node is not accessible, then we extend it with the (k', l')-bushy system where every top node is terminated (to force termination). By the construction we will reduce the bushiness of the tree by $2w^2$ levels and the system by $2w$ levels, where $w(n)$ is the number of nodes of length $2n$ on the system. So finally our new splitting subtree system admits $(k'^{-2w^2}, h, l'^{-2w}, q)$. See construction in §VIII.3.4.

So in a similar way, we can choose the bushy functions g_0 and p_0 to be sufficiently large so that we can always continue the construction after finitely many steps of reducing bushiness.

force $\varphi_e^B \leq_T A$

In the case of getting a yes answer to our key question, we try to terminate some of the nodes on (T, S) to force $\varphi_e^B \leq_T A$, i.e., on each $S(A)$, the values of $\varphi_e^\rho(x)$ for all nonterminal ρ's are the same. This is done simply by relativizing [KLxx, Section 5] or §VIII.2.4 along each path on T. At the inductive step, suppose that we have defined a new $S'(\tau)$ for some τ where every top-node agree on $\varphi_e^{(\cdot)}(x)$ for $x < t$, and on our forcing totality tree system, each of our next level τ' extending τ has $S(\tau')$ where top-nodes converge at the next bit t, so we simply terminate the minority part of these top-nodes, i.e., keep the part which has a l^--bushy subset.

In the end we need to show that we are not terminating a (k', l')-bushy tree system for the new reason. That is, we need to argue that these nodes τ where we terminate a l'-bushy system on $S(\tau)$ do not have a k'-bushy subset.

The argument here is the same as §VIII.2.4. Note that if τ is the first place (from the root) where we terminate l'-bushy system, then by induction, we know that at the previous level we are not terminating such a l'-bushy subtree system, therefore the remaining top-nodes are still bushy (at least l^--bushy). So on such $S(\tau)$ we are able to find two l'-bushy subsets Π_0, Π_1 which form a pairwise splitting pair. This contradicts the fact that such τ's cannot be k'-bushy by the positive answer to our question.

forcing $B \leq_T \varphi_e^B$

If we have a no answer to our question, then we know that for every accessible pair (τ, ρ) we can always find a finite k'-bushy tree Γ above τ such that for every top node τ' of Γ there are two pairwise e-splitting l'-bushy Π_0, Π_1 on $S(\tau')$. We use the following lemma to iterate the delayed splitting process. The proof is essentially the same as the proof of [KLxx, Lemma 6.2] (construct a string ψ_i bit by bit and argue by cases).

Lemma VIII.3.2. *Let (τ_1, ρ_1) and (τ_2, ρ_2) be given. Above (τ_1, ρ_1), we have a (u_1, v_1)-bushy tree system (Γ_1, Δ_1), and above each top node τ' on Γ_1, there is a u_1-bushy tree Γ such that for every top-node τ'' on Γ and every top-node ρ' on $\Delta_1(\tau')$, there are two v_1-bushy pairwise e-splitting pairs Π_0, Π_1 above ρ' on $S(\tau'')$. Let Π be the collection of all top-nodes on such Π_0 and Π_1. Above (τ_2, ρ_2) we have a (u_2, v_2)-bushy subtree system Γ_2, Δ_2 such that every top node on the system has* *longer φ_e-image than any node in Π. Within these given nodes, we can find subtree systems above (τ_1, ρ_1) and (τ_2, ρ_2) which are (u_1^{--}, v_1^{--})-bushy and (u_2^{--}, v_2^{--})-bushy respectively, and such that all top-nodes on one tree system pairwise e-split with all top-nodes on the other tree system.*

In this lemma, note that all these bushy tree systems can have terminal nodes (on (T, S)) and these terminal nodes do not have to satisfy the properties about their φ_e-images, in the conclusion we get tree systems with possible terminal nodes, and similarly these terminal nodes may not have e-splitting properties.

Now it is clear that we can iterate this lemma with the "no" answer of our question to construct a delayed splitting subtree system. At the inductive step, we have already constructed the tree system up to some finite part (Γ, Δ). Let Π be all the top-nodes on (Γ, Δ). Then to get the next level system, we use the above lemma to make sure that for any two strings ρ_0, ρ_1 in Π, all extensions of ρ_0 e-split with all extension of ρ_1 at the next level tree system.

At the inductive step, above some nodes τ on Γ and ρ on $S(\tau)$ we may have already set up some extensions, and we are going to shrink these extensions step by step to finally reach the ones we want. We pick ρ_1 at τ_1 and ρ_2 at τ_2, two top-nodes (Γ, Δ) which we haven't forced splitting. Note that we may have already picked some of their extensions (Γ_1, Δ_1) and (Γ_2, Δ_2) as in the lemma. For the top-nodes of (Γ_1, Δ_1) we apply the negative answer to our key question to get e-splitting pairs (if it is accessible) or terminated bushy subtree systems (if it is not accessible). For (Γ_2, Δ_2), we extend it to some tree system where the top-nodes have long enough φ_e-images by totality forcing. Then apply the lemma we can get two tree systems extending (τ_1, ρ_1) and (τ_2, ρ_2) respectively by shrinking the bushiness by a factor of 4 and make sure that all extensions of ρ_1 and all extensions of ρ_2 pairwise e-split. We do not define them as extensions on the new subtree system until we finally finish this whole process making every pair of top-nodes delayed e-splitting.

(If ρ_1 and ρ_2 are on the same $S(\tau)$, i.e., τ_1 and τ_2 are the same in the above lemma, then we use [KLxx, Lemma 6.2] and the construction there instead to force delayed splitting on the tree.)

Each iteration reduces the bushy number from f to f^{--}, and in the end the tree part bushiness is reduced by $2w^2$ levels and the system part by $2w$ levels, where $w(n)$ is the number of nodes on the system of length $2n$. All terminations in the construction are old ones, and so the admitting quadruple is reduced to $(k'^{-2w^2}, h, l'^{-2w}, q)$.

CHAPTER IX
THE N-R.E. DEGREES: UNDECIDABILITY AND Σ_1 SUBSTRUCTURES (JOINT WITH SHORE AND SLAMAN)

IX.1 Introduction

Turing reducibility (introduced in [Tur39]) captures the intuitive notion of one set $A \subseteq \mathbb{N}$ being computable from another B, We write $A \leq_T B$, A is *Turing reducible* to or *computable from* B to mean that there is a Turing machine (program) Φ that can compute A if given access to an "oracle" for B in the sense that the computing machine is augmented by a procedure that allows it to ask for any number n it computes if $n \in B$ and to receive (on a tape) the correct answer. This reducibility naturally induces a partial order $\leq_T$ on the set $\mathcal{D}$ of equivalence classes (called *Turing degrees* or simply *degrees*) $\mathbf{a} = \{B | A \leq_T B \ \&\ B \leq_T A\}$. The structure of $\mathcal{D}$ then captures that of relative complexity of computation of sets and functions (on $\mathbb{N}$). The study of this relation on all sets (functions), and on many important subclasses of sets has been a major occupation of recursion (computability) theory ever since its introduction.

In addition to the full structure, $\mathcal{D}$, the most important substructures studied have been those of the recursively enumerable degrees, $\mathcal{R}$, and $\mathcal{D}(\leq \mathbf{0}')$, the degrees below the halting problem, $K = \{e | \Phi_e(e) \text{ converges}\}$ whose degree is denoted by $\mathbf{0}'$. The recursively enumerable sets are those which can be enumerated (listed) by a recursive (computable) function. They can also be seen as those sets A for which there is a very simple approximation procedure, a recursive function $f(x,s)$ to the characteristic function $A(x)$ of A such that $\forall x(f(x,0) = 0 \ \&\ \lim f(x,s) = A(x))$ that changes its mind about membership in A at most once, i.e. there is at most one s such that $f(x,s) \neq f(x,s{+}1)$. Shoenfield's Limit Lemma ([Shn59]) says that the sets (or functions) computable from the halting problem $0'$ are precisely those with some convergent recursive approximation, i.e. the sets A such that there is a recursive function $f(x,s)$ such that $\forall x(f(x,0) = 0 \ \&\ \lim f(x,s) = A(x))$. So, while for each x there are only finitely many changes, the number of such changes over all x may be unbounded.

In this chapter we study a natural hierarchy of intermediate classes of sets and degrees. The n-r.e. sets are those for which there is a recursive approximation $f(x,s)$ as above for which there are at most n changes of value at each x. The corresponding degree structures are denoted $\mathcal{D}_n$, the degrees of the n-r.e. sets. (So $\mathcal{D}_1 = \mathcal{R}$ the r.e. degrees.) This hierarchy was introduced by Putnam ([Put65]) and Gold ([Gol65]). It was extended into the transfinite by Ershov ([Ers68a], [Ers68b] and [Ers70]) who proved that the sets in the transfinite hierarchy he defined are precisely those computable from $0'$.

The early work on degree theories began with the investigation of local algebraic or order-theoretic properties of the structures. This work continues in full force to this day. In the past three decades or so, a more global approach has emerged as well. Here one studies issues such as the decidability or, more generally, the complexity of the theories of degree structures as well as related questions about definability in, and possible automorphisms of, these structures.

For the first couple of decades, a major motivating idea was that (at least some of) these structures should be simple and characterizable by basic algebraic properties. Shoenfield's conjecture ([Shn65]) would have been such a complete characterization of $\mathcal{R}$ analogous to that of the rationals as the countable dense linear order without endpoints. Even after the conjecture had been refuted by Lachlan ([Lac66]) and Yates ([Yat66]), Sacks ([Sac66]) still conjectured that the r.e. degrees were decidable. More recent results have produced a dramatically different prevailing paradigm for $\mathcal{D}$, $\mathcal{D}(\leq_T \mathbf{0}')$ and $\mathcal{R}$ as well as many degree structures for other notions of reducibility. Rather than seeing the complexity of the structures as an obstacle to characterization, it suggests that a sufficiently strong proof of complexity would completely characterize each structure. Instead of expecting the structures to be decidable and homogeneous with many automorphisms (like the rationals), one looks to prove that the theories are as complicated as possible, there are definable degrees and that the structure has few automorphisms.

Typical results include the following:

Theorem IX.1.1. *$\mathcal{D}$, $\mathcal{D}(\leq_T \mathbf{0}')$ and $\mathcal{R}$ are each undecidable by Lachlan ([Lac68]); Epstein ([Eps79]) and Lerman ([Ler83]); and Harrington and Shelah ([HaS82]), respectively.*

Theorem IX.1.2. *The theories of $\mathcal{D}$, $\mathcal{D}(\leq_T \mathbf{0}')$ and $\mathcal{R}$ are as complicated as possible, i.e. recursively isomorphic to true second order arithmetic for $\mathcal{D}$ and to true first order arithmetic for $\mathcal{D}(\leq_T \mathbf{0}')$ and $\mathcal{R}$ by Simpson ([Sim77]); Shore ([Sh81]); and Harrington and Slaman and then Slaman and Woodin (both unpublished) (see Nies, Shore and Slaman ([NSS98]) for a proof and stronger results), respectively.*

Theorem IX.1.3. *All relations invariant under the double jump that are definable in arithmetic are definable in $\mathcal{D}$, $\mathcal{D}(\leq_T \mathbf{0}')$ and $\mathcal{R}$ where for $\mathcal{D}$ we mean second order arithmetic and for the others first order by Slaman and Woodin ([SW01]) (see [Sla91] for an announcement and [Sh07] for a quite different proof that applies to various substructures of $\mathcal{D}$ as well), essentially Shore [Sh88] (but see also [NSS98], Theorem 3.11 and the remarks following it) and Nies, Shore and Slaman ([NSS98]), respectively. (The converse holds by the definability of these degree structures in arithmetic.)*

A survey paper for this area is [Sh06].

We take the first steps on this road for the structures $\mathcal{D}_n$ by proving that they are all undecidable. We conjecture that our work can be extended along the lines of [NSS98] to show that their theories are also all recursively isomorphic to that of true arithmetic. Perhaps one can even prove definability results as done there for $\mathcal{R}$. Basic survey papers on the structure of the $\mathcal{D}_n$ are [Ars09], [Ars10] and [SYY09].

Another important theme in the study of these degree structures has been delimiting the similarities and explicating the differences among them. While it is relatively easy to distinguish among $\mathcal{D}$, $\mathcal{D}(\leq_T \mathbf{0}')$ and $\mathcal{R}$ in many way the issue becomes particularly compelling when we turn to the $\mathcal{D}_n$. It is easy to imagine, and was proved early on, that moving from $\mathcal{R}$ to all sets or even to the unlimited approximations characterizing those below $0'$ introduces many differences. For the $\mathcal{D}_n$, however, the question is what does the ability to change precisely one more time buy us in terms of additional degrees, algebraic structure and complexity.

Of course, the first question is are the $\mathcal{D}_n$ actually distinct. Indeed, there are, for each n, $(n+1)$-r.e. degrees which are not n-r.e. ([Coo71] with the stronger result that they can be found not even n-REA in [JSh84]). While the one quantifier theory of all the degree structures from $\mathcal{R}$ to $\mathcal{D}$ are the same since one can embed all finite (even countable) partial orderings into $\mathcal{R}$ (and so all the rest as well), there were many early results establishing elementary differences between $\mathcal{R}$ and the other $\mathcal{D}_n$ with cupping, density and lattice embedding properties playing the featured role (as in, for example, [Ars85], [CLLSS91] and [Dow89], respectively). Differences between any of the other $\mathcal{D}_n$, however, seemed hard to find. [Dow89] even conjectured that they might all be elementarily equivalent, i.e. all sentences (in the first order language with $\leq$) true in any $\mathcal{D}_n$ for $n \geq 2$ is true in all of them. This conjecture was not refuted until quite recently. Arslanov, Kalimullin and Lempp ([AKL10]) provide an elementary difference between $\mathcal{D}_2$ and $\mathcal{D}_3$. In fact, the sentence they exhibit on which the structures differ is at the smallest possible level: two quantifiers ($\forall\exists$). They conjecture (as one would now expect) that the $\mathcal{D}_n$ are pairwise not elementarily equivalent. They also conjecture that this level of difference ($\forall\exists$) is as small as possible in the strong sense that every $\exists\forall$ sentence true in any $\mathcal{D}_n$ is true in every $\mathcal{D}_m$ for $m \geq n$.

Now an $\exists\forall$ sentence is true if there are choices (parameters substitutable) for the existentially quantified variables such that the resulting universal sentence is true of these parameters. The strongest way that their conjecture could be true is for the same parameters to work in both structures. This view brings to mind a much earlier question raised about other pairs of our degree structures. Are any Σ_1 substructures of any others. ($\mathcal{M}$ is a Σ_1 *substructure* of $\mathcal{N}$, $\mathcal{M} \preceq_1 \mathcal{N}$, if for any Σ_1 formula $\exists\bar{y}\varphi(\bar{x},\bar{y})$ where φ is quantifier free and any choice of elements $\bar{a}$ from $\mathcal{M}$, $\mathcal{M} \vDash \exists\bar{y}\varphi(\bar{a},\bar{y}) \Leftrightarrow \mathcal{N} \vDash \exists\bar{y}\varphi(\bar{a},\bar{y})$.)

Slaman ([Sla83]) proved early on that this fails at the extreme ends: $\mathcal{R} \not\preceq_1 \mathcal{D}(\leq \mathbf{0}')$ (and so, *a fortiori*, $\mathcal{D}_n \not\preceq_1 \mathcal{D}(\leq \mathbf{0}')$ for any $n \geq 1$. Slaman and then others raised the natural question of whether it could be that $\mathcal{D}_n \preceq_1 \mathcal{D}_m$ for any $n < m$. Yang and Yu ([YY06]) provided a negative answer for $n = 1$ and $m = 2$ (and so for any $m \geq 2$). We complete the picture by showing that $\mathcal{D}_n \not\preceq_1 \mathcal{D}_m$ for any $n < m$. (We have just heard that Arslanov and Jamaleev are preparing a different proof for the case $n = 2$.)

Turning now to our proofs, we begin with undecidability. As usual (see for essentially our situation §2 of [NSS98] or for a more general model theoretic treatment [Hod93, §5.3]), we have a formula $\varphi_D(x,\bar{p})$ which, for each choice of parameters $\bar{p}$, defines a subset D of our structure $\mathcal{D}_n$ and another formula $\varphi_R(x,y,\bar{p})$ which defines a binary relation R on D. To prove undecidability it suffices to show that, as the parameters vary over $\mathcal{D}_n$, a sufficiently rich class of structures (D,R) are coded in this way. In our case, we code partial orders. As the (r.e.) set of theorems of the theory of partial orders is recursively inseparable from the (r.e.) set of sentences (of the language of partial orders) that are false in some finite partial order ([Ta62]), it suffices to code any collection of relations containing all finite partial orders. The point here is that if $\mathcal{D}_n$ were decidable then the set of sentences true in every partial order coded by φ_D and φ_R as the parameters $\bar{p}$ range over all elements of $\mathcal{D}_n$ would be recursive. Of course, it contains the theorems of the theory of partial orders and, if we code all finite ones, is disjoint from the set of sentences with finite counterexamples. As it turns out, it is no more difficult to prove that one can code all recursive partial orders than all finite ones. This is what we do explicitly in our proof of Theorem IX.1.4.

Theorem IX.1.4. *Given a recursive partial order $(\omega, \leq_*)$ and an $n \geq 1$, there exist uniformly n-r.e. sets G_i for each $i \in \omega$, an n-r.e. set L and r.e. sets P and Q such that:*

1. *Each $\mathbf{g}_i$ is a maximal n-r.e. degree below $\mathbf{a}$ such that $\mathbf{q} \not\leq \mathbf{g}_i \vee \mathbf{p}$ where $A = \bigoplus_i G_i$.*

2. $\mathbf{g}_i \leq \mathbf{g}_j \vee \mathbf{l}$ *if and only if* $i \leq_* j$.

Thus the required formulas φ_D and φ_R defining our domains and order relations have parameters $\mathbf{a}$, $\mathbf{p}$ and $\mathbf{q}$. The first says that $\mathbf{x}$ is a maximal degree below $\mathbf{a}$ such that $\mathbf{q} \not\leq \mathbf{x} \vee \mathbf{p}$. The second says that $\mathbf{x} \leq \mathbf{y} \vee \mathbf{l}$. so we have the desired result.

Theorem IX.1.5. *The theories of $\mathcal{D}_n$ are undecidable for every n.*

If instead of recursive inseparability, we wanted to rely only on the undecidability of the theory of partial orders, we should code all partial orders recursive in $0'$ as every sentence which is not a theorem (of the theory) has a counterexample recursive in $0'$ by the effective version of the completeness theorem.

One can with only minor modifications not affecting the structure of our proof handle partial orders recursive in $0'$. We precisely describe the modifications needed in IX.6.5. With some additional work and a serious reorganization of the priority tree, one can get all partial orders recursive in $0''$. One puts in a new type of node which guesses in a Δ_3 procedure at each bit of information about this partial order and bases later work on these guesses. The added complexity is considerable without much gain for applications. It seems that one can even get any Σ_3 partial order by a slightly more complicated procedure. We briefly describe this procedure in IX.6.5 as well.

It is worth remarking that our proof works for $n = 1$ as well as all larger n. Indeed, it can be significantly simplified for $n = 1$ by omitting all items that consider the possibility that the G_i and W_i (the list of n-r.e. sets recursive in A) are not r.e. This gives a considerably simplified proof of the undecidability of $\mathcal{R}$ along the lines suggested in Harrington and Shelah but with a simpler statement using fewer parameters and a significantly easier construction. We do not believe any proof even for $\mathcal{R}$ along these lines has been published before.

We next turn to Σ_1 substructures.

Theorem IX.1.6. $\mathcal{D}_n \not\preceq_1 \mathcal{D}_m$ *for* $n < m$.

The technical result needed here is the following generalization of Theorem 1.12 in [YY06] who do the case $n = 1$:

Theorem IX.1.7. *For any $n \geq 1$, there are r.e. degrees $\mathbf{g}, \mathbf{p}, \mathbf{q}$, an n-r.e. degree $\mathbf{a}$ and an $n+1$-r.e. degree $\mathbf{d}$ such that:*

1. *For every n-r.e. degree $\mathbf{w} \leq \mathbf{a}$, either $\mathbf{q} \leq \mathbf{w} \vee \mathbf{p}$, or $\mathbf{w} \leq \mathbf{g}$.*
2. $\mathbf{d} \leq \mathbf{a}$, $\mathbf{q} \not\leq \mathbf{d} \vee \mathbf{p}$, *and* $\mathbf{d} \not\leq \mathbf{g}$.

This theorem shows directly that $\mathcal{R}_n$ is not a Σ_1 elementary substructure of $\mathcal{R}_{n+1}$ in the language with $\vee$ as well as $\leq$: In $\mathcal{R}_n$, no $\mathbf{w}$ below $\mathbf{a}$ has the property that $\mathbf{q} \not\leq \mathbf{w} \vee \mathbf{p}$ and $\mathbf{w} \not\leq \mathbf{g}$ while in $\mathcal{R}_{n+1}$, $\mathbf{d} \leq_T \mathbf{a}$ has both properties. We can elimnate $\vee$ by rephrasing the property of $\mathbf{w}$ as $\exists \mathbf{z}(\mathbf{w}, \mathbf{p} \leq \mathbf{z}\ \&\ \mathbf{q} \not\leq \mathbf{z})\ \&\ \mathbf{w} \not\leq \mathbf{g}$ which is Σ_1 in just $\leq$ and so the existence of a $\mathbf{w}$ with this property is true in $\mathcal{D}_{n+1}$ (i.e. $\mathbf{d}$) but false in $\mathcal{D}_n$. Of course, as $\mathbf{d}$ is in $\mathcal{D}_m$ for every $m \geq n+1$, $\mathcal{D}_n \not\preceq_1 \mathcal{D}_m$ as well.

Much of the construction and verification is the same for Theorems IX.1.4 and IX.1.7. We treat the first theorem as primary. In §IX.2 where we cover basic notions and conventions common to both, we use curly brackets { } to indicate changes (usually alphabetic only at this stage) for the second theorem. The rest of the chapter is divided into two parts, one for each of the theorems. Each part describes first the requirements (§IX.3, IX.7), then the priority tree (§IX.4, IX.8), the construction (§IX.5, IX.9) and finally the verifications that the construction succeeds (§IX.6, IX.10). We describe everything in full detail for the first theorem and then for the second describe only the changes needed. In our descriptions of the constructions, material enclosed in square brackets [ ] is meant to convey intuition or describe aspects of the construction that will only be verified later. It is not part of the formal definition of the construction procedures.

As might be expected from the types of requirements, both constructions are $0'''$ arguments even for the case $n = 1$. As these constructions go, however, ours are at the simpler end: the priority tree is finitely branching, there is no backtracking and only one type of requirement is injured along the true path. The key idea for carrying the arguments from the r.e. case ($n = 1$) to the n-r.e. one ($n > 1$) in Theorem IX.1.4 is what we call *shuffling* (§IX.5.2). Roughly speaking, at the crucial $0'''$ determined nodes, we are attempting to construct functionals Δ that, to working towards the maximality of the $\mathbf{g}_i$, try to compute some given n-r.e. set $W = \Phi(A)$ from one G_i that we are building over the full construction. The most delicate part of the verification of the first construction is the correctness of these functionals (Lemma IX.6.8). We argue that the cause of an incorrect computation, say of $\Delta(u)$, must be that some number z entered A for the first time and allowed $W(u)$ to change. Another delicate argument shows that if W also changed for the first time, we could correct the functional Δ (or see that we are not on the true path). If the change in W was not that u entered for the first time, we argue that we can shuffle A between two past values (giving, via Φ, two different values for $W(u)$) by repeatedly taking z out and putting it back in as necessary so as to eventually show that $W \neq \Phi(A)$.The point here is that z has entered A for the first time while the change in W is not a first change. Thus as W_l can make no more than n changes overall, it can make no more than $n-2$ additional changes. On the other hand, as z has entered A for the first time, we can make $n-1$ more changes in A and so eventually guarantee that $W \neq \Phi(A)$.

In the second construction (§IX.9.2), the correctness of the functionals Δ becomes immediate as we simply change G_i when necessary. The crucial problem then becomes guaranteeing the correctness of computations from G_i diagonalizing against D (§IX.10). Here we take advantage of the fact that D can change one more time than any other set by using a procedure like one used in [YY06] to remove a number (that entered for the first time) from D. In our case it allows us to either cure some problem we are facing or start a shuffling procedure for A diagonalizing against the offending W.

IX.2 Basic Notions and Conventions

Given a set A, let $A \upharpoonright u$ be the initial segment of A of length u.

In this chapter, we use upper case Greek letters to denote Turing functionals. For any Turing functional Δ, the *use* of a convergent $\Delta(A; x)$ is defined as the least number u such that $\Delta(A \upharpoonright u; x) \downarrow$. We use lower case Greek letters corresponding to the Turing functional to denote the use, e.g. $\delta(A; x)$ denotes the use of Δ at x. More importantly, we injure the computation by adding $\delta(A; x) - 1$ into A, but not by adding $\delta(A;x)$ into A. If it doesn't cause confusion, we may omit A and write $\delta(x)$.

We will have families Ψ, Π, Θ and Φ which specify standard enumerations of all the Turing functionals. We follow the usual conventions for such standard enumerations such as the approximations to these functionals for any (approximation to an) oracle set at stage s asks questions about (makes use of) only numbers less than s and converges only at inputs less than s. We also assume, without loss of generality, that for the standard enumerations with two oracles such as $\Theta(G \oplus P; x)$ the uses on both are always the same and we denote it by $\theta(x)$.

We will also construct two families of Turing functionals $\Delta(G)$ and $\Gamma(W \oplus P)$. For the ones with two oracles, we *do not* require that the uses of W and P are the same. Hence we can write $\gamma(W;x)$ and $\gamma(P;x)$ to denote the W and P parts of the use, respectively. Although for simplicity we generally work as if we are specifically defining these oracles at each individual x with the associated uses, we really are assuming that the uses are monotonic in x and make all changes to keep them that way, usually without explicit mention. As Q is r.e., when we are computing it from $W \oplus P$ by Γ, except for this monotonicity condition, we only need to produce computations (axioms) that at x give output 0 when $x \notin Q$. These may be injured and new ones put into Γ (perhaps with larger use). In the case that $x \notin Q$ and we are expecting Γ to compute Q, we must eventually settle on a convergent computation (axiom) applying to $W_i \oplus P$. If $x \in Q$, when x enters Q it suffices to kill any current computation of 0 from $W_i \oplus P$. We do this by putting a number less than the P-use into P . We can then simply keep the value of Γ at 1 without changing the use (remembering that P is r.e.).

In our two constructions, we specify priority trees which grow downward. At each stage s of the construction, we build a path of length s {at most s} of *accessible* nodes along the priority tree. Our convention is that, the nodes to the left of, or above, a node α have higher priority. We always preserve the information used at previous stages by the nodes that are to the left of the accessible ones by initializing the nodes that are to the right of the accessible ones, i.e., remove all information from previous stages such as witness numbers, defined functionals and imposed restraints.

Nodes can impose two types of restraint: a permanent one or an alternating one. Permanent restraint means that no node of lower priority can act so as to injure the restraint by changing a set where restrained. By convention permanent restraint imposed at stage s restrains the initial segments of length s of L and all the G_i {A, D and G}. Any permanent restraint on P must be mentioned specifically. [We never need to restrain Q.] Alternating restraints are caused by the announcements of A-stages or P-stages which we describe later in the construction. Basically, during A -stages, we remove the alternating restraint for L and the G_i {A and D} allowing numbers to enter (or leave) these sets and we impose an alternating restraint on P and Q {and G} so that no numbers can enter P or Q {or G} at this stage. During P-stages, we do the opposite (except that no numbers ever leave the r.e. sets P or Q {or G}).

IX.3 Requirements I

We now begin the proof of our main technical result.

Theorem IX.3.1. *Given a recursive partial order $(\omega, \leq_*)$ and an $n \geq 1$, there* *exist uniformly n-r.e. sets G_i for each $i \in \omega$, an n -r.e. set L and r.e. sets P and Q such that:*

1. *Each $\mathbf{g}_i$ is a maximal n-r.e. degree below $\mathbf{a}$ such that $\mathbf{q} \not\leq \mathbf{g}_i \vee \mathbf{p}$ where $A = \bigoplus_i G_i$.*
2. *$\mathbf{g}_i \leq \mathbf{g}_j \vee \mathbf{l}$ if and only if $i \leq_* j$.*

First, for the negative order facts, we have requirements for each pair $i \not\leq_* j$ and each e:

$$\Psi_{e,i,j} : \ \Psi_e(L \oplus G_j) \neq G_i.$$

Similarly for each triple (i, j, e) with $i \neq j$, we also want:

$$\Pi_{e,i,j} : \ \Pi_e(G_j) \neq G_i,$$

i.e., the G_i's are pairwise incomparable.

Then for each pair (i, e) we need:

$$\Theta_{e,i} : \ \Theta_e(G_i \oplus P) \neq Q.$$

We also need the main requirements that each $\mathbf{g}_i$ is a maximal n-r.e. degree $\mathbf{g} \leq_\mathbf{T} \mathbf{a}$ such that $\mathbf{q} \not\leq \mathbf{g} \vee \mathbf{p}$. We let W_i be an effective list of all the n-r.e. sets.

$$\Phi_{e,i} : \ \Phi_e(A) = W_i \rightarrow [\exists \Gamma(\Gamma(W_i \oplus P) = Q) \vee (\exists k(W_i \leq_T G_k))].$$

- Note that these Φ requirements by themselves do not ensure that each $\mathbf{g}_i$ is maximal. That is why we need the Π requirements to make all the G_i's pairwise incomparable. The Φ requirements then do guarantee that the $\mathbf{g}_i$ are maximal.

If it does not cause confusion, we may omit the subscripts of the requirements and sets in our argument to simplify the notation.

Finally, we have to deal with the positive order facts, i.e., $G_i \leq_T L \oplus G_j$ for $i <_* j$. We will guarantee that, for $x > i, j$, $x \in G_i \Leftrightarrow x \in L$ or $x \in G_j$. Putting numbers into a G_i is initiated only by a Ψ or Π requirement. For Π action, we simply put a witness x that is going into G_i (for diagonalization) into L as well. When action is initiated for diagonalization by Ψ at stage s, we put x into G_i and also into each G_l with $l >_* i$ for each $l < x$. As, in this case, $i \not\leq_* j$, this action does not add elements to G_j and so it does not injure the Ψ computation initiating the action. We say that each witness x (for a Ψ or Π requirement) has an *associated block of sets* (the G_l such that $l < x$ and $i \leq l$ or G_i and L, respectively). During the construction x moves into or out of all the sets in its block simultaneously.

IX.4 Priority Tree I

We put all the Ψ, Π, Θ and Φ requirements into one priority list. Our *priority tree* consists of *nodes* and *branches*. Each node is associated with a requirement in the list and each branch leaving a node is assigned an outcome. We label each node with its associated requirement and each branch with the assigned outcome. When we list outcomes of a node we do so in a left to right order that specifies the left to right order on the priority tree of the branches leaving that node

A Ψ or Π node has two outcomes: d and w, which stand for "diagonalization" and "wait" respectively.

A Φ node has outcomes s_{n-1}, s_{n-2}, $\ldots$, s_1, i and w. Outcome s_i stands for "shuffle" for the i-th time. We will explain what this means in detail in the construction. Roughly, it means that we expect to shuffle between two versions of A (by removing numbers from A and then possibly putting them back in) as we cycle back to this node. The expected result of this shuffling is to guarantee that $\Phi(A) \neq W$ by a diagonalization. Outcome i stands for "infinite" agreement between $\Phi(A)$ and W and outcome w stands for "wait".

A Θ node β has outcomes d, g_{α_1}, g_{α_2}, $\ldots$, g_{α_k} and w. As usual, d and w stand for "diagonalization" and "wait" respectively. Each α_i is a Φ node above β which has outcome i along β. If γ is a node below β extending the g_{α_i} branch from β, then we say that γ *sees* an $\alpha_i - \beta$ pair. [The intuition here is that γ believes that α_i and its associated requirement is satisfied by β.]

A Φ node α is *active* at $\gamma \supset \alpha$ if α has outcome i along γ and γ does not see an $\alpha' - \beta'$ pair such that $\alpha' \subseteq \alpha \subset \beta' \subset \gamma$. For there to be a g_{α_i} outcome of a Θ node β, we also require that α_i be active at β. We order these g_{α_i}'s from left to right in descending order going down the tree to β, i.e., $\alpha_1 \subset \alpha_2 \subset \cdots \subset \alpha_k$. [This choice of left to right order comes into play at the very end of the proof of Lemma IX.6.16.]

The priority tree is defined recursively as follows: suppose τ is an immediate extension of σ, we associate τ with the highest priority requirement among all requirements which either have not appeared above τ or are Φ requirements that, above τ, have appeared only at nodes δ with outcome i such that τ sees an $\alpha - \beta$ pair with $\alpha \subset \delta \subset \beta$. [So δ looks inactive but not really satisfied, i.e. if satisfied at some earlier point it has since been "captured" by some other pair.] Then we add the corresponding number of branches (outcomes) below τ. It is easy to see that this tree is recursive.

IX.5 Construction I

At stage s of the construction, we build a path of length s of the *accessible nodes* along the priority tree. It is possible that at some accessible node we will *announce* that s is an A-stage or a P -stage. All later nodes accessible at s must respect this announcement by acting according the the rules governing A-stages or P-stages: no changes in A can occur once a P-stage has been announced and none in P or Q once an A-stage has been announced. In the construction, we will make sure that the first accessible Θ node with a type g outcome (if any) makes the announcement for the stage s. An over-riding rule is that permanent restraint imposed by a node (not since initialized) is not violated by action at any node of lower priority (i.e.

below it or to its right). If any instruction below leads to any such situation, we do not carry it out and instead go to outcome w [and do nothing].

In this section, we first describe the construction at stage s for each node when there has been as yet no announcement for the stage and then specify the modifications for when there has already been one.

IX.5.1 No announcement, Ψ or Π node

The actions at Ψ and Π nodes are quite standard: If it is the first time we come to this node (after it was last initialized), then we pick a witness number x which is *fresh*, i.e., larger than any number we have seen by this point in the construction. In general, at a Ψ ($\Psi(L \oplus G_j) \neq G_i$) or a Π ($\Pi(G_j) \neq G_i$) node with a witness x already assigned (and not yet canceled by initialization), we check whether the computation at x converges to 0. If it diverges or converges to a nonzero number, then we do nothing and go to the w outcome. If it converges to 0 and $x \notin G_i$, then we do a diagonalization: put x into G_i and into all the other sets in its block as described in Section IX.3, impose permanent restraint [to preserve the use of the computation] and go to the d outcome. If x is already in G_i, then we (again) go to outcome d [and keep the restraint already imposed].

IX.5.2 No announcement, Φ node

At a Φ node α ($\Phi(A) = W$) if we have not yet had a type s outcome (since α was last initialized) let t be the last stage at which α was accessible (since last initialized). If there is a such stage and a $u < l_\alpha(t), l_\alpha(s)$ such that $\Phi(u)$ (and so $W(u)$) differ at t and s with the difference not being that u has entered W for the first time and the only change in $A \restriction \phi(u)$ at t is that some z has entered its block of sets for the first time because of the action of a node extending α then we *initiate a shuffle on* z by removing z from its block of sets, impose permanent restraint and go to outcome s_1. We call this shuffle strategy *Plan S* with *shuffle points* $sp1(= t) < sp2(= s)$. [Note that these shuffle points have the property that $A_{sp1}(= A_{sp1} \restriction sp1)$ and $A_{sp2}(= A_{sp2} \restriction sp2)$ differ below $sp1$ only in that z is in its block of sets in A_{sp2} and out of them in A_{sp1}. More crucially, they produce different values for Φ at some u, i.e. $\Phi(A_{sp1}; u) \neq \Phi(A_{sp2}; u)$.] If we had an outcome of type s at the last stage t at which α was accessible, we check whether $W(u)$ is different at s than at t. If so, restore the initial segment of A to the version of A which is different from the current one (by putting z into or taking it out of its block of sets), impose permanent and let the outcome be s_{i+1}. If not, we stay at the s_i outcome. [This maintains any previously imposed permanent restraint.]

If we haven't initiated shuffling, let $l_\alpha(s)$ be the length of agreement between the current versions of $\Phi(A)$ and W. Note that whenever we initialize this node α, we also initialize the values of this function to be 0. If this is the first time that $l_\alpha(s) > 0$ after it has last been initialized, or $l_\alpha(s) > l_\alpha(t)$ where t is the last stage when α had an i outcome, then we go to the i outcome; otherwise we go to the w outcome.

If we go to the i outcome, we continue to define a functional Γ [aiming to make $\Gamma(W \oplus P) = Q$]. At this point, we enumerate a new axiom making $\Gamma(W \restriction l_\alpha(s) \oplus P \restriction v; w) = Q(w)$, where v is a fresh number and w is one more than the largest number where we have previously defined Γ (since it was last initialized). If P has changed on its Γ-use at some $x < w$ and the change was caused by the action of a node $\beta\hat{}g_\alpha$ [necessarily extending $\alpha\hat{}i$] with witness x as in §IX.5.4, then we redefine $\Gamma(x)$ to be the current value of $Q(x)$ with W-use $l_\alpha(s)$ and fresh P-use. [As P is r.e. this change permanently invalidates the previous axiom for $\Gamma(x)$.] Similarly, if W has changed on its use u_1 (where its old P-use is v_1) so as to make $\Gamma(x)$ divergent but x has not entered Q, we see if x is currently the witness for some Θ node β below α (for G). If so, we look at the last stage t at which β was accessible and see if its outcome was g_α. If G has not changed on $\theta(x)$ as defined at the point of stage t at which β was reached and the change in W includes one at some u making it different from the common value of $\Delta(u)$ and $W(u)$ at t, then we redefine $\Gamma(x)$ with W-use $l_\alpha(s) = u_2$ and fresh P-use. In all other cases of a W or P change on $\gamma(W;x)$ or $\gamma(P;x)$, respectively, that makes $\Gamma(x)$ divergent we redefine $\Gamma(x)$ with the same uses as it last had but for the new values of W and P (subject, of course, to our monotonicity requirements on the use).

IX.5.3 No announcement, Θ node

At a Θ node β accessible for first time after it has been last initialized, we pick a fresh witness x for diagonalizing $\Theta(G_i \oplus P) \neq Q$. In general, if we have a witness x already assigned (and not yet canceled by initialization), we check whether the computation converges at the witness x. [As usual when there are higher priority requirements that are expected to put infinitely many numbers into a set, we restrict our attention to computations that are consistent with our beliefs as prescribed by our actions in §IX.5.4. Here this means the following:] We also require that the computation be *believable*, i.e. for every requirement $\hat{\Theta}$ assigned to a node α with witness $\hat{x}$ and $\alpha\hat{}g_{\hat{\alpha}} \subseteq \beta$ for some $\hat{\alpha}$, $\theta(x) < \gamma_{\hat{\alpha}}(P;\hat{x})$ and if $\gamma_{\hat{\alpha}}(P;\hat{x})$ has been previously increased by a $\hat{W}$ change (as described at the end of §IX.5.2) from say u_1 to u_2 and $v_1 - 1$ is not yet in P then $\theta(x) < v_1$ as well. If $\Theta(x)$ does not converge with a believable computation or so converges to a nonzero number, then we go to the w outcome and do nothing.

[If the believable computation $\Theta(G_i \oplus P;x)$ converges to 0 with P-use $\theta(x)$, then we would like to diagonalize, i.e., put x into Q and preserve the P and G_i use of the computation. However, we must worry about whether doing so injures some already defined Γ computation at a node above β. For example, if there is a such a $\Gamma(W \oplus P) = Q$ which computes $Q(x) = 0$ with $\gamma(P;x) \leq \theta(x)$, then our desired diagonalization would falsify this computation of Q while correcting the Γ computation (by putting its use into P and redefining the functional) would injure our Θ computation for diagonalization. Our plans must be more subtle.] If $\Theta(x)$ converges with a believable computation we proceed as follows:

Let $\alpha_1 \subset \alpha_2 \subset \cdots \subset \alpha_k$ be all the active nodes above β with each α_j defining its functional $\Gamma_j(W_{l_j} \oplus P)$. Let $\gamma_j(P;x)$ be the P-use of Γ_j at x, if it has already been defined.

Plan D: diagonalization

If $\theta(x) < \gamma_j(P;x)$ for all j for which $\gamma_j(P;x)$ is defined, we do a modified diagonalization: We enumerate x into Q and also enumerate $\gamma_{\alpha_j}(P;x) - 1$ into P for each j. [This allows us to correct the $\Gamma_j(x)$ when $\alpha_j\hat{\ }i$ is next accessible.] We now impose the usual permanent restraint but also one on $P \upharpoonright \theta(x)$ [to preserve the Θ computation] and go to outcome d. Until β is initialized, it has outcome d at every later stage at which it is accessible.

IX.5.4 Stage announcements

If we cannot follow Plan D, i.e., $\theta(x) \geq \gamma_j(P;x)$ for some j, then we take the largest j such that $\theta(x) \geq \gamma_j(P;x)$ [and are likely to go to outcome g_{α_j} where we build a functional Δ computing W_{l_j} from G_i]. [The choice of j is relevant at the very end of the proof of Lemma IX.6.8 but our choice of the largest j (rather than say the smallest) doesn't make any difference in this construction. It does, however, matter in the at the end of the proof of Lemma IX.10.1 for our second theorem.]

Plan A: A-stage announcement

If this is the first time (since the last initialization) that we would go to the g_{α_j} outcome or the last time we went there we announced a P-stage then we go to outcome g_{α_j} and announce an A -stage [and so allow elements to be enumerated into or taken out of A].

Otherwise, let t be the last stage when $\beta\hat{\ }g_{\alpha_j}$ was accessible. By our construction and case assumption, t must have been announced as an A-stage at $\beta\hat{\ }g_{\alpha_j}$. (If some node to the right or left of $\beta\hat{\ }g_{\alpha_j}$ made an announcement at stage t then β would not have been accessible at t . If some node α above β made an announcement at t then one would also have to be made above β at s contrary to our case assumption that no announcement has been made at this stage before we reached β.)

Plan P: P-stage announcement

We now go to the g_{α_j} outcome and extend Δ by adding axioms computing $W_{l_j}(u)$ from G_i with fresh use for any $u < l_{\alpha_j}(s)$ for which Δ has not previously been defined. In addition, we put $\gamma_j(P;x)-1$ into P to injure the current Θ computation (since $\gamma_j(P;x) \leq \theta(x)$). [This kills the current computation of $\Gamma(x)$ and as P is r.e. it can never apply to $W \oplus P$ again.] Moreover, if $\gamma_j(P;x)$ has been previously increased by a $\hat{W}$ change (as described at the end of §IX.5.2 from say u_1 to u_2 and $v_1 - 1$ is not yet in P then we also put $v_1 - 1$ into P. [This kills the old computation of $\Gamma_j(x)$ as well as and guarantees that it too will never again apply to $W \oplus P$.] [We will redefine Γ_j with axioms using the new version of P with a fresh P-use and W_{l_j} use $l_{\alpha_j}(v)$ when we next get to $\alpha_j\hat{\ }i$ at v. The result of this action is that we increase the Γ_j use from P and W_{l_j} and so the next time when this β is accessible with the g_{α_j} outcome, the use $\theta(x)$ must be larger than that of this stage. If this happens infinitely often $\Gamma_j(x)$ diverges but we expect to satisfy the associated Φ requirement by building $\Delta(G_i) = W_{l_j}$ at $\beta\hat{\ }g_{\alpha_j}$. We then also satisfy the Θ requirement associated with β as $\Theta(x)$ diverges as well.] We now announce that the current stage is a P-stage.

If there has been a change in G_i that leaves $\Delta(u)$ undefined where it had previously been defined, we put in a new axiom computing the current value of $W_{l_j}(u)$ with the old use.

[We shall argue for β on the true path with true outcome g_{α_j} that we build Δ consistently and correctly compute W_{l_j} at each stage (Lemma IX.6.8). Typically, it turns out that, along the true path, if W_{l_j} has changed where previously computed, then G_i must have changed at the corresponding part used in the computation.]

IX.5.5 Modifications with a stage announcement

When there has already been a stage announcement before we reach β , the node β has to obey the appropriate rules. For a Ψ, Π or Θ node, we see what we would have done if there had been no stage announcement as yet. If that action is compatible with the current stage announcement (no announcement of an A-stage or change in A if a P-stage; no change in P or Q and no announcement of a P-stage if an A -stage), we proceed as if there had been no announcement. If not, we do nothing and go to outcome w.

For a Φ node, the modification is slightly trickier. [Later we will need the fact that each node along the true path passes down alternating A and P restraints in the construction.] Here in order to go to the i outcome, we need to wait (with outcome w) for a stage when the stage announcement is different from the last stage t when we had an i outcome, and also the length of agreement is longer than its last value. [In this way, the Φ node passes down alternating A and P restraints along the i outcome.] When we have already initiated shuffling, we act as before at A-stages and at P-stages we go to outcome w. [This maintains the permanent restraint imposed when we initiated shuffling or last shuffled as the nodes that imposed it are now to our left.]

IX.6 Verification I

IX.6.1 True path and true outcome

First of all, as in usual priority tree arguments, there is a leftmost path accessible infinitely often. (Each node has only finitely many outcomes.) This is the *true path* and the outcomes along it the *true outcomes.*

Lemma IX.6.1. *Numbers enter or leave A or L only when permanent restraint is imposed by a Π, Ψ or Φ node. When such nodes β impose permanent restraint, we move to the left of any previous outcome that has been accessible since β was last initialized.*

Proof. By inspection of the construction. □

Lemma IX.6.2. *At most one node acts to change A at any stage s.*

Proof. If we first act at α to change A at s then we move to an outcome to the left of all previously accessible ones (since α was last initialized) by Lemma IX.6.1. So all later nodes accessible at s that can change A are accessible for the first time since last initialized and so at most appoint fresh witnesses or (for Φ nodes) begin their construction of Γ anew. None of these witnesses can go in at s as no convergences can be seen at numbers larger than s. No shuffling can been initiated for any of the Φ nodes by construction. □

Lemma IX.6.3. *If a node α is initialized at stage s then it never later acts to change any set below s.*

Proof. If α is a Π, Ψ or Θ node it only acts to put numbers at least as large as its witness x into A or P and any witness appointed after s is larger than s. For Π and Ψ nodes this is immediate. For Θ, its action puts numbers of the form $\gamma(P;x) - 1$ into P and by construction $\gamma(P;x) > x$. For Φ nodes, the only action changing sets is shuffling. This shuffling only involves numbers appointed below α at stages when α was accessible since it was last initialized. □

Recursively along the true path, we now determine the actions of the nodes on it after no node to their left is ever accessible and prove that all the requirements are satisfied along it. For any node β on the true path we let $s(\beta)$ be the first stage at which β is accessible but after which no node to its left is ever accessible again.

Lemma IX.6.4. *Any permanent restraint imposed by a node β at any $s \geq s(\beta)$ is never injured by any other node.*

Proof. The only actions that can injure such restraint after $s(\beta)$ are ones by nodes above it on the true path. None can change A or L by Lemma IX.6.1. As for P, the only permanent restraint imposed on P is by Θ nodes when we go to outcome d and restrain $P \upharpoonright \theta(x)$. Now nodes $\hat{\beta}$ above β of type Θ with outcomes $g_{\hat{\alpha}}$ may put numbers into P but they only put in ones of the form $\gamma_{\hat{\alpha}}(P;\hat{x})$ and our believability requirement on the computation of $\Theta(x)$ guarantees that all of the current values of these $\gamma_{\hat{\alpha}}(P;\hat{x})$ are larger than $\theta(x)$. The only way one of them could decrease is if it had previously been increased from u_1 to u_2 by a change in W as described at the end of §IX.5.2 and then W changes back to the old value before the old computation is killed by $v_1 - 1$ going into P. However, our believability condition also requires that $\theta(x)$ is less than these v_1 as well. Any later change increases the use above the previous values Thus no changes every occur in P below $\theta(x)$. □

Lemma IX.6.5. *The final witness for any node β chosen at $s(\beta)$ is larger than any permanent restraint of higher priority than β.*

Proof. By construction the witness is chosen fresh and so larger than anything previously seen. The only nodes of higher priority that can impose permanent restraint later are ones above β. None of type Π, Ψ or Φ can do so by Lemma IX.6.1. One of type Θ also imposes permanent restraint only when it moves left to outcome d contradicting our definition of $s(\beta)$. □

Before we show that the requirements are satisfied we analyze the alternating restraint.

IX.6.2 Alternating A and P-stages

Lemma IX.6.6. *Every node along the true path above the first Θ node β with type g outcome on the true path never sees or makes a stage announcement (imposes alternating restraint) when accessible. For the other nodes α on the true path, their true outcomes, o, are accessible at infinitely many A and P-stages. Indeed, after $s(\alpha\hat{\ }o)$, the stages at which $\alpha\hat{\ }o$ is accessible alternate between A and P ones (the node passes down alternating A and P restraints along its true outcome).*

Proof. For any node above β the claim is immediate from the rules of the construction. For a Π or Ψ node below β, it is immediate from the construction that after $s(\beta)$ either we always have outcome w or, whenever we are at β after the first time we have outcome d we also have outcome d. So for these nodes the Lemma is obvious. For a Φ node below β, either the true outcome is shuffling (s), or waiting (w), or infinitary (i). In the two former cases, the outcome is again eventually constant: Once we move to a type s outcome, the construction guarantees that we can move only to the left to another type s outcome. Thus the outcome is eventually constant at some type s outcome. As for outcome w, any rightmost outcome that is the true outcome of a node β on the true path is the outcome at almost every stage at which β is accessible. In the third case, our construction in §IX.5.5 ensures that it passes down alternating A and P restraint along the outcome i as required.

The Θ node β which first makes the announcements along the true path must have true outcome some g_α. Then according to our construction, if it announced a P-stage the last time it was accessible, we follow Plan A and make an A-stage announcement. If it last announced an A-stage, then we follow Plan P and announce a P-stage.

Finally for any other Θ node β' on the true path, the claim is also immediate for true outcome d or w as above. In the case of a g_α outcome, the construction automatically guarantees that it always waits for an alternation in the type of restraint to move again to the true g_α outcome. ☐

We next analyze the functionals Δ and Γ that we construct.

IX.6.3 The functionals are well-defined and correct

Lemma IX.6.7. *The functionals Γ built at nodes $\alpha\hat{\ }i$ on the true path starting at $s(\alpha\hat{\ }i)$ are well-defined, i.e., we do not add contradictory axioms in the construction and when defined give the correct current value for Q. They are defined on arbitrarily large initial segments of ω and so, if convergent at every x, they correctly compute the desired sets.*

Proof. It is clear by construction that we define $\Gamma(x)$ at least once for each x: As $\alpha\hat{\ }i$ is on the true path, $l_\alpha(s)$ is going to infinity on the stages at which $\alpha\hat{\ }i$ is accessible. On each of these stages we define Γ on a new number. Once defined $\Gamma(x)$ is then defined at every stage at which $\alpha\hat{\ }i$ is accessible by construction. As for consistency and correctness, note first that it is immediate from the construction that at any stage at most one axiom in Γ applies to the current value of $W \oplus P$. Now, the only way any problem could arise is if $Q(x)$ has changed for some x but W and P have not changed on the corresponding use or they change and then

W reverts back to a previous value that applies to some older computation (with value 0). However, in our construction, $Q(x)$ can change only when we implement Plan D to put x into Q at a some stage v for a Θ node $\hat{\beta}$. Any number x put into Q by such nodes to the right of $\alpha\hat{\ }i$ must be both appointed fresh and then put in while we are to the right of $\alpha\hat{\ }i$ without $\alpha\hat{\ }i$ becoming accessible in between. Thus when $\alpha\hat{\ }i$ is again accessible Γ has not been defined at x and so when we define $\Gamma(x)$ we set it equal to 1 with the first axiom and all later ones as well. Any x put in by nodes to the left of $\alpha\hat{\ }i$ are in by $s(\alpha\hat{\ }i)$ and so we define Γ correctly on them as well.

Thus we can assume that $\hat{\beta}$ is below $\alpha\hat{\ }i$. If α is active at $\hat{\beta}$, then when x enters Q at v we put $\gamma(P;x)-1$ into P by construction and so kill the current computation and put in a new one giving the correct answer when we next reach $\alpha\hat{\ }i$. If α is not active at $\hat{\beta}$, there must be a first $\beta' \subset \hat{\beta}$ with $\beta'\hat{\ }\hat{\alpha} \subset \hat{\beta}$ for some $\hat{\alpha} \subseteq \alpha$. So in particular, α is active at β'. Now β' has a witness x' necessarily less than x (as x is appointed later) and at stage v when we reached $\beta'\hat{\ }g_{\hat{\alpha}}$ we put $\gamma(x') < \gamma(x)$ into P and so kill the current computation of $\Gamma(x)$ and correct it when we next reach $\alpha\hat{\ }i$. Note that both P and Q are r.e. so $\gamma(P;x)-1$ has never been in P before and x will never leave Q. As we impose permanent restraint when we go to outcome d and diagonalize, no later change in W can return us to any old computation. ☐

Lemma IX.6.8. *The functional Δ defined at the true g_α outcome of a Θ node β (for G_i) on the true path starting at $s(\beta\hat{\ }g_\alpha)$ is well-defined. When $\Delta(u)$ is convergent while we are at $\beta\hat{\ }g_\alpha$, it always give the current value of $W(u)$ on an initial segment of ω. Indeed, $\Delta(G_i) = W$ as desired. (For notational convenience we assume that α is assigned the Φ requirement for W which is constructing the functional Γ at its i outcome.)*

Proof. As for the final claim, note first that by construction if $\Delta(u)$ is ever defined it is defined at every $u' < u$ and then it is defined there at every later P-stage at which $\beta\hat{\ }g_\alpha$ is accessible. Moreover, at these stages we extend its domain of definition to $l_\alpha(s)$ which is going to infinity since $\alpha\hat{\ }i \subseteq \beta$ is on the true path. Once $\Delta(u)$ is defined, its use never changes by construction and so if, at every stage when defined at $\beta\hat{\ }g_\alpha$, it correctly computes $W(u)$, it does so in the end.

For correctness, we argue by induction on the stages at which $\beta\hat{\ }g_\alpha$ is accessible beginning at $s(\beta\hat{\ }g_\alpha)$ that $\Delta(u) = W(u)$ at every u at which Δ is defined. This is obviously true by construction if s is the first stage at which $\Delta(u)$ is defined. Suppose it is true at s and the next stage at which $\beta\hat{\ }g_\alpha$ is accessible is t and the problem occurs at u.

Note that no change in $A \restriction s$ or $P \restriction \theta(x)$ can occur between s and t. No node to the right of $\beta\hat{\ }g_\alpha$ can make such a change by Lemma IX.6.3. No node to its left can do so as they are never accessible after $s(\beta\hat{\ }g_\alpha)$. No node above $\beta\hat{\ }g_\alpha$ can change A at all by Lemma IX.6.1. Nodes above $\beta\hat{\ }g_\alpha$ may act to put numbers of the form $\gamma_{\hat{\alpha}}(P;\hat{x})$ into P via other Θ requirements $\hat{\beta}$ above β with outcome $g_{\hat{\alpha}}$ but by our believability condition on the $\Theta(x)$ computation, at stage s none of them are below $\theta(x)$ at s. The only way one of them could decrease is if it had previously been increased from u_1 to u_2 by a change in W as described at the end of §IX.5.2 and then W changes back to the old value before the old computation is killed by $v_1 - 1$ going into P. However, our believability condition requires that $\theta(x)$ is also less than these v_1.

If s was a P-stage then no change occurred in $A \upharpoonright s$ at s so none has by stage t. Thus if $W(u)$ at t is different from its value at s it would be different from $\Phi(A;u)$ at t since that is the same as it was at s. This would move us to outcome w at α and so $\beta\hat{\ }g_\alpha$ would not be accessible for a contradiction.

Suppose then that s was an A-stage. No change in $P \upharpoonright \theta(x)$ occurs at s as no node above the one announcing the A-stage can change P without declaring a P-stage or moving left and none after it can because it is an A-stage. Moreover, none can occur before t as above. If no change occurs in A at s then, as none occurs before t, $\Phi(u)$ and $\Delta(u)$ would be the same at t as at s. If this value is not that of $W(u)$ at t then $\alpha\hat{\ }i$ would again not be accessible at t for a contradiction. Thus there has been some change in A at s. By Lemma IX.6.2, there is precisely one z that entered or left its block of sets at s. By Lemmas IX.6.1 and IX.6.3 , the node σ causing the change must extend $\beta\hat{\ }g_\alpha$ as we are after $s(\beta\hat{\ }g_\alpha)$ and $\beta\hat{\ }g_\alpha$ is accessible. Now if there is no change in $W(u)$ between s and t, then the only way we could have a disagreement with $\Delta(u)$ at t (so the old axiom for $\Delta(u)$ at s is no longer valid but we cannot simply put in a new one with the same value) is that the change for z returns us to a previous computation of $\Delta(u)$ giving a different value. However, such a change in z can be caused only by σ shuffling z. Such a shuffle returns $A \upharpoonright v$ to its value at a previous stage v. If $\Delta(u)$ was defined at v then by induction it would have the same value as $\Phi(u)$ and $W(u)$ and no change can have happened in any of these when we reach t. Thus we would still have agreement at t as required. So we may also assume that $W(u)$ is different at s and t.

Suppose first that $z < \theta(x)$ and G_i is in its block of sets. Next, suppose the change occurred because of some shuffling procedure at σ . If $\Delta(u)$ was first defined before the shuffling began at σ, then we would have a contradiction as above.

If $\Delta(u)$ was first defined after the shuffling began, say at $v \leq s$ with, for the sake of definiteness, $z \in G$, then its use is fresh at v and so larger than z. Let $v' \geq v$ be the next stage after v at which we shuffle z at σ. If $v' = s$ then z has been in G_i from v to s and is removed at stage s by our action at $\sigma \supset \beta\hat{\ }g_\alpha$. Thus when we next return to $\beta\hat{\ }g_\alpha$ at t we have $z \notin G_i$ for the first time since $\Delta(u)$ was defined and we redefine it to be the current value of $W(u)$ as required. So we may assume that $v' < s$ and at stage v' we have $\Phi(u) = W(u) = \Delta(u)$ at $\beta\hat{\ }g_\alpha$ by induction. When we reach σ at v' we remove z from its block of sets including G_i and impose permanent restraint. When $\beta\hat{\ }g_\alpha$ is next accessible, say at $v'' \leq s$, we redefine $\Delta(u) = W(u) = \Phi(u)$ with $z \notin G_i$ (and its block of sets) but with the rest of $A \upharpoonright v'$ the same as it was at v' (because of the permanent restraint imposed at v' which, by Lemma IX.6.1, could be violated only by moving to the left of σ which could then not cause our problem at s). If $v'' = s$ then at stage s we shuffle z back into G_i (and its block of sets) at σ and impose permanent restraint. We next return to $\beta\hat{\ }g_a$ at t and have $\Phi(u)$ and $\Delta(u)$ and so $W(u)$ the same as they were at stage v' at $\beta\hat{\ }g_\alpha$, i.e. they all agree as required. Finally, if $v'' < s$ then later at stage s we shuffle at σ between the values for all of these sets and functionals that we had at v' and v''. Thus once again when we reach $\beta\hat{\ }g_\alpha$ at t all agree.

Thus the change that has occurred is that z entered G_i for the first time at s. If the change in $W(u)$ is not that u has entered for the first time, then we would have initiated a shuffling procedure at α at stage t and so move to $\alpha\hat{\ }s_1$ hence to the left of $\beta\hat{\ }g_\alpha \supset \alpha\hat{\ }i$ for a contradiction.

Thus through stage s, $W(u) = 0$. Suppose $\Delta(u)$ was first defined at s', of course with value 0 and fresh use q. We claim that $z < q$ and so its entry into G allows us to correct $\Delta(u)$ at t as desired. If not, it was chosen fresh as a witness for σ at a point in the construction during a stage $s'' \geq s'$ after $\Delta(u)$ was defined at s' but before s. In this case, however, $z > s''$ and so $z > \phi(u)$ at s''. Note that $\Phi(u)$ is defined at s'' because z is appointed at $\sigma \supset \beta \supset \alpha\hat{\ }i$. Now from the point of stage s'' at which $\alpha\hat{\ }i$ is accessible to stage s any change in $A \upharpoonright s''$ would initialize σ and so z could not enter A at s. (At s'' no node between $\alpha\hat{\ }i$ and σ can change A without moving left of σ. Then at σ all nodes to the right of σ are initialized and so cannot make any changes below s'' by Lemma IX.6.3. As σ appointed a witness at s'', this is the first stage at which σ has been accessible since it was last initialized so all A action by nodes below σ also involve only numbers larger than s''. Finally, any A action after s'' by a node of higher priority than σ would also initialize it by Lemma IX.6.1.) Thus at s, $\phi(u)$ and $A \upharpoonright \phi(u)$ are the same as they were at s'', i.e. $\Phi(A, u) = 0 = W(u)$ at s with the same computations as at s''. Now the only changes in A during stage s is that z enters its block of sets but $z > s''$ and then no changes occur in $A \upharpoonright s$ before stage t. Thus at stage t we also have $\Phi(A, u) = 0$ and so if $W(u) = 1$ at t, the outcome of α would not be i, for a contradiction.

Finally, suppose $z \geq \theta(x)$ or G_i is not in its block so no change occurs in $G_i \upharpoonright \theta(x)$ at s and so none before t. When $\Delta(u)$ was defined at the P-stage v (necessarily before the A-stage s), $u < l_\alpha(v)$ and so after v, $u < \gamma(W; x)$ whenever it is defined. In fact, at each P-stage during which we reach $\beta\hat{\ }g_\alpha$ (starting with v) we put $\gamma(P; x)$ into P and subsequently increase $\gamma(W; x)$ to $l_\alpha(v') > l_\alpha(v)$ when we are next at $\alpha\hat{\ }i$ (at $v' > v$). Each computation of $\Gamma(x)$ killed in this way can never to reapply to $W \oplus P$ as P is r.e. As we cannot reach $\beta\hat{\ }d$, the only other way $\gamma(x)$ can change requires a W change that causes a difference between the previously computed common values of Δ and W. By our induction assumption this cannot have occurred before s. So all axioms for $\Gamma(x)$ provided before s are invalid by the end of stage s.

As $W(u)$ has different values at s and t, the change in W introduces a value of $W(u)$ that we see at $\alpha\hat{\ }i$ at some first stage v'' after s but no later than t. As we have argued, no old computation of $\Gamma(x)$ is still valid at v''. Thus by construction we would increase $\gamma(P; x)$ at v'' to a fresh value larger than $\theta(x)$. When we return to $\beta\hat{\ }g_\alpha$ at t, $\gamma(P; x)$ is now larger than $\theta(x)$ which has not changed since s. Thus by construction, g_α cannot be the outcome of β at t for a contradiction. □

IX.6.4 All requirements are satisfied

Finally we want to show that all requirements are satisfied.

The positive order requirements are easily verified by our construction.

Lemma IX.6.9. *If $i <_* j$ then $G_i \leq_T L \oplus G_j$.*

Proof. Consider an $x > i, j$. To decide if $x \in G_i$ go to stage x of the construction and see if x has been appointed as a witness for some Π or Ψ requirement with G_i in its block. If not, then $x \notin G_i$. (Indeed x is not in any G_k.) If it is in the block for a Π requirement then L is also in its block. If for a Ψ requirement then G_j is in the block. In any case, as once appointed x moves into or out of all sets in its block during the entire construction, $x \in G_i \Leftrightarrow x \in L$ in the Π case and $x \in G_i \Leftrightarrow x \in G_j$ in the Ψ case. □

We now move to the negative (diagonalization) requirements.

Lemma IX.6.10. *The Π and Ψ requirements are satisfied.*

Proof. Suppose the requirement is assigned to the node β on the true path. If, after $s(\beta)$, we ever go to outcome d and so diagonalize, the result is immediate from the construction and Lemma IX.6.4. If not, it must be that the outcome is always w after after $s(\beta)$. If the relevant computation converged to 0 the correct computation would be available from some point on and so by Lemma IX.6.6 we would eventually see it at an A-stage and so move to outcome d by Lemma IX.6.5. If not, then x never enters G_i and we also satisfy the requirement as desired. □

We next consider the Θ requirements.

Lemma IX.6.11. *If a Θ node β on the true path has true outcome d then the associated requirement is satisfied.*

Proof. Consider the stage $s(\beta\hat{\ }d)$ when β has outcome d (and is never again initialized). We put the witness x into Q and impose permanent restraint to preserve the computations $\Theta(G_i \oplus P; x) = 0$. Lemma IX.6.4 shows that this computation is preserved. □

Lemma IX.6.12. *If a Θ node β on the true path has true outcome g_α, then its requirement is satisfied. Indeed, for x the final witness for β, both $\theta(x)$ and $\gamma_\alpha(x)$ go to infinity on the stages when $\beta\hat{\ }g_\alpha$ is accessible. Moreover, any time we increase $\gamma_\alpha(x)$ because of a W change as at the end of §IX.5.2, we later kill the P-use of the old computation as well by putting v_1 into P.*

Proof. In this case, by our construction (and Lemmas IX.6.5 and IX.6.6), we infinitely often put numbers $(\gamma(P; x) - 1)$ into P and redefine $\Gamma(W_i \oplus P)$ with a fresh P-use. (The first of these Lemmas implies that the numbers we want to put into P are larger than any permanent restraint as they are of the form $\gamma(P; x)$ which is larger than the witness x for β.) So by our criteria for going to outcome g_α, we infinitely often see $\theta(x) > \gamma(P; x)$, so $\theta(x)$ must go to infinity (when $\beta\hat{\ }g_\alpha$ is accessible) along with $\gamma(P; x)$ and the computation $\Theta(G_i \oplus P; x)$ diverges. As for any increase in $\gamma(x)$ because of W change as in §IX.5.2, the next time we are at $\beta\hat{\ }g_\alpha$ we put the associated v_1 into P by construction. □

Lemma IX.6.13. *If a Θ node β on the true path has true outcome w then the associated requirement is satisfied.*

Proof. Note that if the outcome of β were d at any stage after $s(\beta)$ then d would be the true outcome by construction. Thus in our case, we never put the final witness x for β into G_i. So our only concern is that $\Theta(G_i \oplus P; x) = 0$. In this case, there is a stage after which it always converges to 0 and with a fixed use. By the previous Lemma this computation is believable at almost every stage when we are at β. Thus by construction and Lemmas IX.6.5 and IX.6.6, as in Lemma IX.6.12, we would eventually have outcome d for a contradiction. □

We now turn to the Φ requirements.

Lemma IX.6.14. *If a Φ node β on the true path has true outcome of type s, then the associated requirement is satisfied.*

Proof. The nature of the shuffling points guarantees that, at every stage with outcome of type s, $\Phi(A;x)\downarrow\neq W(x)$ and so this is true at the end of the construction as well and the requirement is satisfied. The crucial point here is that the permanent restraint imposed by β which are increasing as we move left among the type s outcomes can never be injured (other than by the shuffling done by β itself) by Lemma IX.6.4. □

Lemma IX.6.15. *If a Φ node β on the true path has true true outcome w then the associated requirement is satisfied.*

Proof. If $\Phi(A) = W$ then the length of agreement would go to infinity and so, by construction and Lemma IX.6.6, we would eventually move to outcome i after $s(\beta\hat{\ }w)$ for a contradiction. □

Finally, we have to deal with the case that every Φ node on the true path has true outcome i.

Lemma IX.6.16. *Every Φ requirement is satisfied.*

Proof. As usual in a $0'''$ priority tree argument, we want to consider the last node α along the true path assigned to a given Φ requirement. To see that there is such a node, argue by induction on the Φ requirements. The point here is that any Φ requirement, once assigned to a node that is never again initialized, can return to the list of requirements from which we draw to make assignments of requirements to nodes (along the true path) only when a strictly higher priority node becomes inactive. So once no node with a higher priority Φ requirement assigned ever becomes inactive again, the next node β assigned to Φ either becomes inactive once along the true path (by being satisfied by action at a lower Θ node on the true path) and then remains inactive or it never becomes inactive on the true path. In either case, Φ is never assigned to a node below β by the definition of the priority tree.

Let α be the last node along the true path assigned to the Φ requirement $(\Phi(A) = W_i)$. By the previous two Lemmas, we may assume that its true outcome is i. If there is an Θ node β $(\Theta(G_k \oplus P) \neq Q)$ on the true path with true outcome g_α, then we have built a functional Δ at $\beta\hat{\ }g_\alpha$ that computes W_i from G_k by Lemma IX.6.8.

If there is no such Θ node β, then we claim that we have successfully built $\Gamma(W_i \oplus P) = Q$ starting at $s(\alpha\hat{\ }i)$. By Lemma IX.6.7, we only have to verify that $\gamma(x)$ is eventually constant for each x. We begin to define our Γ at $s(\alpha\hat{\ }i)$. Assume inductively that $\gamma(\hat{x})$ has stabilized for $\hat{x} < x$. Thereafter, once $\Gamma(x)$ is defined, our construction allows $\gamma(x)$ to increase because of a change in P at most once for each time some $\beta\hat{\ }g_\alpha$ below $\alpha\hat{\ }i$ is accessible and the Θ requirement assigned to β has witness x or once when $\beta\hat{\ }d$ is accessible (again with witness x for β). It can increase because of a W change at most finitely often for each such $\beta\hat{\ }g_\alpha$ and stage. (At worst only when W changes on the domain of Δ at that stage.) At most one β has x assigned as a witness. If β is not on the true path, it can be accessible with witness x at most finitely often. If β is on the true path, once $\beta\hat{\ }d$ is accessible, $\beta\hat{\ }g_\alpha$ cannot be accessible again unless β is initialized and so chooses a new witness. If $\beta\hat{\ }g_\alpha$ is accessible infinitely often, then some $g_{\hat{\alpha}}$ (possibly to the left of g_α) would be its true outcome and so $\hat{\alpha} \subseteq \alpha$. If $\alpha = \hat{\alpha}$ we contradict our case assumption. If $\hat{\alpha} \subset \alpha$ then α would be come inactive and Φ would be reassigned later to a node below $\beta\hat{\ }g_\alpha$ on the true path contradicting our choice of α. Thus $\Gamma(x)$ can change at most finitely often as required. $\square$

IX.6.5 Δ_2^0 and Δ_3^0 partial orders

To handle partial orders recursive in $0'$ we make the following changes in the construction:

We begin with a recursive approximation $f(i,j,s)$ to the (characteristic function of the) relation $i \leq_* j$. We now have requirements $\Psi_{e,i,j}$ for every e, i, j with a new additional leftmost outcome n. At stage s at a node α for $\Psi_{e,i,j}$, if $f(i,j,,s) = 1$ (so we think we do not want to diagonalize) we go to outcome n and do nothing. If $f(i,j,s) = 0$ we act as before with a new definition of the block for our witness x. When x (necessarily larger than i and j) is appointed as a witness, we determine its block by calculating $f(k,l,t)$ for each $k, l < x$ and $t > x$ until we reach a t at which either $f(i,j,t) = 1$ or the relation on numbers $k, l < x$ defined by $f(k,l,t)$ is a partial order $\preceq$. In the first case, the outcome is again n and we do nothing. In the second case, we put G_k into the block for x if and only if $i \preceq k$ (i.e. $f(i,k,t) = 1$). Note that $f(i,j,t) = 0$ by our case assumption and so G_j is not in the block.

To see that this modification works, note that if $i \not\leq_* j$ then from some point on $f(i,j,t) = 0$ and so we never again have an outcome n for $\Psi_{e,i,j}$ and so satisfy the negative order requirements as before. For the positive ones, suppose $k \leq_* l$ and for $t \geq t_0$, $f(k,l,t) = 1$. For any witness $x \geq k, l, t_0$, if its block does not contain k then, of course, $x \notin k$. If it does contain k it also contains l and so $x \in G_k$ if and only if $x \in G_l$. (The case for Π requirements is as before.)

The modifications needed for partial orders recursive in $0''$ are more complicated. For each i, j we insert a requirement into the priority order used for the Δ_2^0 case and so on each path of the priority tree a node ε that guesses in a Δ_3^0 way whether $i \leq_* j$, i.e. the node has infinitely many outcomes $\langle x, k\rangle$ with $x \in w$ and $k \in \{0, 1\}$ ordered lexicographically. We organize determining the outcome of ε at each stage s so that if $\langle x, k\rangle$ is the leftmost outcome accessible infinitely often then x is the least witness to the Σ_3 formula which says that $i \leq_* j$ if $k = 1$ and the least witness to the Σ_3 formula which says that $i \not\leq_* j$ if $k = 0$. In addition we coordinate this guessing with the stage announcements so that the true outcome passes on alternating restraint as before. We then act at nodes as in the Δ_2^0 case but using at each node only the information about the ordering coded on the outcomes of the ε type nodes above it. So for for a Ψ type requirement, if the relation given in this way is not a partial order $\preceq$ or says that $i \preceq j$, we go to outcome n. (We put the ε nodes on the tree so that any node for a requirement $\Psi_{e,i,j}$ has an ε type node above it assigned to $i \leq_* j$.) If it is does specify a partial order with $i \not\preceq j$, then we act as before but now the block of sets for a witness x consists of all G_k with $i \preceq k$. One can now verify that the construction works. The argument for the positive order relations runs as follows: If $i \leq_* j$, find the node σ on the true path by which that fact has been decided. Nodes to the left of σ put only finitely many x into G_i and can be ignored. For nodes to its right that appoint any witness x (necessarily before stage x), we can wait for the node to be initialized to see if x enters G_i. For nodes below σ in the tree assigned to any Ψ requirement, any witness x that puts G_i in its block also puts G_j and so for those x, $x \in G_i \Leftrightarrow x \in G_j$. Of course, for witness x for $\Pi_{e,i,k}$ type nodes, $x \in G_i \Leftrightarrow x \in L$ as before. Of course, for any x not appointed as a witness for one of these type nodes, $x \notin G_i$.

If the partial order is only Σ_3, then one adjust the previous procedure by instead of single nodes with Δ_3 guessing at $i \leq_* j$ putting in individual nodes for each i and j guessing that a particular number is the (least) witness to the Σ_3 fact that $i \leq_* j$. Along a path with the Π^0_2 outcome that the witness is correct, one follows a coding stratgey incorporating this individual fact. If it is true, then some node σ on the true path has the correct witness and all nodes below it obey the required coding strategy. Nodes not below this one, are handled as above. For each node guessing a witness for the Σ_3 fact, where we see that it is false, i.e. along a path with the Σ^0_2 outcome, we put in one more Ψ requirement for $i \not\leq_* j$. So if $i \not\leq_* j$, then along the true path, we will put in $\Psi_{e,i,j}$ requirements for every e and so satisfy the requirement.

IX.7 Requirements II

We now turn to our second technical result:

Theorem IX.7.1. *For any $n \geq 1$, there are r.e. degrees $\mathbf{g}, \mathbf{p}, \mathbf{q}$, an n-r.e. degree $\mathbf{a}$ and an $n+1$-r.e. degree $\mathbf{d}$ such that:*

1. *For every n-r.e. degree $\mathbf{w} \leq \mathbf{a}$, either $\mathbf{q} \leq \mathbf{w} \vee \mathbf{p}$, or $\mathbf{w} \leq \mathbf{g}$.*
2. *$\mathbf{d} \leq \mathbf{a}$, $\mathbf{q} \not\leq \mathbf{d} \vee \mathbf{p}$, and $\mathbf{d} \not\leq \mathbf{g}$.*

Our list of requirements is very similar to the one used for our first technical theorem:

1. $\Psi_e : \Psi_e(G) \neq D$;
2. $\Theta_e : \Theta_e(D \oplus P) \neq Q$;
3. $\Phi_{e,i} : (\Phi_e(A) = W_i) \rightarrow [\exists\Gamma(\Gamma(W_i \oplus P) = Q) \vee \exists\Delta(\Delta(G) = W_i)]$.

In addition, we need to make $D \leq_T A$. Note that we only add elements into D by diagonalization for Ψ requirements. Whenever we pick a witness x for D, x is fresh at that stage, and we reserve the pair $(x, x+1)$ for coding D into A. If x enters D for the first time, then we also put x into A. If x leaves D later, we either take x out of A or put $x+1$ into A. In the first case, we may shuffle x into and out of A and D simultaneously but allowing at most n changes. In the second case, we may shuffle $x+1$ into and out of A and D simultaneously again allowing at most n changes in A (but this may make for $n+1$ changes in D altogether). Therefore in the end x is in D if and only if x is in A and $x+1$ is not in A. No numbers other than these Ψ -witnesses enter or leave D in our construction, and so D is recursive in A, A is n-r.e. and D is $(n+1)$-r.e.

IX.8 Priority Tree II

Our priority tree here is almost the same as the one used in the first theorem. Of course, we do not have Π nodes. For any Θ node β, we put a new [temporary] outcome r_{α_j} to the left of each g_{α_j}. So the outcomes of β are d, r_{α_1}, g_{α_1}, r_{α_2}, …, g_{α_k} and w. We do not add nodes below these type r outcomes. [So a stage s may terminate at such an outcome before we reach level s of the priority tree. We show, however, in Lemma IX.10.1 that no node of type r can be on the true path.] The notions of active Φ nodes, $\alpha - \beta$ pairs are defined in the same way as in Section IX.4.

IX.9 Construction II

We only specify the construction at stage s when there is no stage announcement. In the case when there is a stage announcement, we act as in §IX.5.5. Note that in this construction we only change G during P stages. The default permanent restraint is on A, D and G while for P it must be specifically mentioned.

IX.9.1 Ψ node and Φ node

At a Ψ node, the action is the same as the one in §IX.5.1 with G for $L \oplus G_i$ and D for G_i.

At a Φ node α, we follow almost the same procedure as we did in §IX.5.2. The only difference is in how we revise the computations from old Γ-axioms. As in the first construction, if P has changed and the change was caused by some β with g_α outcome by putting the old use into P, then we increase the W-use to $l_s(\alpha)$ and P-use to be fresh. If a W change caused some $\Gamma(x)$ to be undefined, then we check whether x is a diagonalization witness for some β below α, if so, we also check whether D has changed by putting in some number for the first time at the previous stage when β was accessible (and β has not been initialized since). If so, we then redefine $\Gamma(x)$ with W-use up to $l_s(\alpha)$ and fresh P-use. In all other cases we redefine the axiom without changing the uses.

IX.9.2 Θ node

[At a Θ node, the obvious difference from the first construction is that we use D in our Θ computation but use G in our Δ computation. So the arguments in IX.6.3 are no longer valid. In the case that W changes, we have no reason to expect a G change. In fact, so far we have no requirements or procedures that put numbers into G. Here we actively put numbers into G to correct Δ computations. We will make full use of the fact that D is $n+1$-r.e., i.e., it has one more chance to change than A and the W_i. We may remove a number from D while leaving it in A but putting $z+1$ into A. This will afford us the opportunity to produce a situation in which we may initiate shuffling.]

At a Θ node β accessible for first time after it has been last initialized, we pick a fresh witness x for diagonalizing $\Theta(D \oplus P) \neq Q$. If we have a witness x already assigned (and not yet canceled by initialization) at β, we check whether the Θ computation converges at the witness x. If we do not have a believable (defined in the same way as in our first theorem) computation $\Theta(D \oplus P; x)$, we go to outcome w. If we do, we follow Plan D as in §IX.5.3 if we can. If not, we have a planned outcome g_{α_j} as in §IX.5.4 and check whether we have not been at this outcome since β was last initialized or whether the previous stage t when we went to this outcome was a P-stage. If so we announce an A-stage and continue the construction below β.

Otherwise, we have two possibilities.

Plan R: removal

Let t be the last stage at which β was accessible. If there is a $y < \gamma(W; x)$ such that $W(y)$ has different values at t and s and the only change in $A \upharpoonright t$ is that some element z entered D and A for the first time at stage t because of the action of a node below β [necessarily a Ψ node], we remove z from D and add $z+1$ into A [so we restore the version of D at stage t up to the θ use]. We go to the r_{α_j} outcome and terminate the current stage of the construction. We call the least such y the *key witness* for the removal plan.

[The idea here is that, before β can become accessible again without being initialized, we would see at α_j if y has left W. If so we would initiate a shuffle there on $z+1$ and initialize β. If not, we will argue that we must go to the left of g_{α_j} and r_{α_j}. Roughly, the idea is that the computation $\Theta(D \oplus P; x)$ will be the same as that at stage t while $\gamma_j(P; x)$ will have been increased above $\theta(x)$ by y entering W.]

Plan G: change G

If we satisfy none of the above criteria, we go to the outcome g_{α_j} and continue to build Δ consistently. For each u, if $\Delta(u)$ was defined at the last stage t at which $\beta\hat{\ } g_{\alpha_j}$ was accessible and $W(u)$ has not changed since then, we simply update the Δ axiom with the current version of G (if necessary) without changing the use. If $\Delta(u)$ was defined at t but $W(u)$ is now different, then let $\delta(u)$ be the use of G in the old Δ computation. We add $\delta(x) - 1$ into G and redefine Δ with a fresh use in G. [We preserve the consistency of Δ by doing this as G is r.e.] Then we also define a new computation $\Delta(u)$ for the next u which was undefined with fresh G-use. Finally, we follow Plan P to add γ uses into P as in Section IX.5.4 and announce a P-stage.

IX.10 Verification II

We can go through most of Section IX.6 and show that we have a leftmost path visited infinitely often (that it is actually infinite follows from Lemma IX.10.1), and each node along the true path is passing down alternating A-stages and P-stages along the true path. There are obvious alphabetic changes needed In Lemmas IX.6.1-IX.6.6 – no L or Π. Otherwise, note first that Plan R action for Ψ nodes are an exception to Lemma IX.6.1. Next, Lemma IX.6.2 applies to D as well as A and we have to remark that if we implement Plan R no node is even accessible thereafter, while Plan R action cannot be the second type to change A (or D) at s by the arguments given in the proof of Lemma IX.6.2. Finally, for the proof of IX.6.3 note that G-uses for Δ are also chosen fresh. It is then not difficult to see that functionals are well-defined and complete the job we assigned if they are along the true path. For the Γ's, use Lemma IX.6.7. For the Δ's, it is directly guaranteed by our construction. [The complicated argument for Lemma IX.6.8 is not needed but see the proof of Lemma IX.10.1 for some remnants of it.]

For the verification that all the requirements are satisfied we continue as in §IX.6.4. The positive order requirements (Lemma IX.6.9) are simply replaced by the requirement that $D \leq_T A$. Our construction guarantees that x is in D if and only if x is in A and $x+1$ is not. Except for the satisfaction of the Ψ requirements (Lemma IX.6.10) in the case of d outcome all the other verifications proceed as in §IX.6.4.

As for the Ψ requirements, the major issue is that here we add elements into G (when we construct Δ) and this action might, *a priori*, injure some apparently satisfied Ψ requirement of lower priority. To show that the Ψ requirements are all satisfied, we first need a few lemmas.

Lemma IX.10.1. *No outcome of type r can be on the true path.*

Proof. The argument is similar to the one in the end of the proof of Lemma IX.6.8. Consider any Θ node β on the true path and suppose, for the sake of a contradiction, that its true outcome is r_α.

Let $s = s(\beta\hat{\ }r_\alpha)$ and, as in the construction, let t be the previous stage at which β had outcome g_α; x, the diagonalization witness at β; y, the key witness for Plan R at s; and z, the unique element that entered D for the first time at t. We remove z from D at s following Plan R. Let s' be the next stage at which β is accessible with a believable Θ computation. If y was ever out of W between s and s' (when α was accessible) then, we would have initiated a shuffle plan at α and so moved left of $\beta \supseteq \alpha\hat{\ }i$ for a contradiction. (As we terminate stage s at $\beta\hat{\ }r_\alpha$ with no action, there is no change in A at s. Between s and s' Lemma IX.6.3 shows that $A \restriction s$ is preserved. So if $W(y)$ changes we satisfy the conditions for shuffling at α.) Thus we also assume that y remains in W at every stage at which α is accessible through stage s'.

Now at stage s we restored the computation $\Theta(D \oplus P, x)$ of stage t by removing z from D: z is the only change to D up to $\theta(x)$ at t by Lemma IX.6.2, and P is preserved by the A-stage announcement at t. Between t and s, $D \restriction t$ is preserved by Lemma IX.6.3 and $P \restriction \theta(x)$ is not injured by nodes to the right. Finally, our believability condition guarantees that $P \restriction \theta(x)$ is also not injured by nodes above β. Thus at stage s' we still have the same computation of $\Theta(x)$ as at stage t.

Now consider the Γ computation we build at $\alpha\hat{\ }i$. At stage s the conditions for Plan R guarantee that $y < \gamma(W; x)$ (at t) has entered W for the first time after t and by s. So when this happens and we are at α we see a change in the W part of $\Gamma(x)$ which makes $\Gamma(x)$ undefined. Therefore, by our construction, we add a new axiom with W-use the current length of agreement and P-use fresh, which is larger than $\theta(P, x)$ at t. This $\gamma(P, x)$ remains large since y remains in W, therefore at stage s' we will see that $\gamma(P, x) > \theta(P, x)$.

For other active α_l's between α and β, the corresponding $\gamma_l(W_l; x)$ are larger than $\theta(P, x)$ at stage t by the rules for going to outcome g_α. The only way that anyone of these uses can decrease is by W_l changing at some stage $v \leq s'$ from its value on $\gamma_l(W_l; x)$ at t back to an older version. If this happens, then when α_l is accessible at $v \leq s'$ we initiate a shuffle at α_l by shuffling $z+1$ which we added into A by Plan R at t with shuffle points t and v. This would move us to the left of β for a contradiction. Therefore these uses cannot decrease, and at stage s' they are still larger than $\theta(P, x)$. Hence at stage s' we will go to the left of the r_α outcome by our construction for the desired contradiction. ∎

Lemma IX.10.2. *In the construction at a Θ node β, if we change G as in IX.9.2 at a stage $s > s(\beta)$, then it must be the case that at the previous stage t when $\beta\hat{\ }g_\alpha$ was accessible, we followed either Plan S or Plan R to change D at some node σ below β.*

Proof. By our construction, if we must change G by putting in some $\delta(u) - 1$, then $W(u)$ and therefore $A \upharpoonright \phi(u)$ must be different at stage s from stage t. By Lemmas IX.6.3 (for nodes to the right), IX.6.2 (for nodes below), IX.6.1 (for nodes above not of type r) and IX.10.1 (to see that there are no type r nodes above), such a change in A can only happen at stage t. Thus we must have changed A and D at stage t below β since β was accessible.

We can change A and D in only three ways: diagonalization at a Ψ node, Plan S or Plan R. By Lemma IX.6.2 if we have applied diagonalization at a node below β, then it is the only change at stage t, so at stage s according to our construction we would want to apply Plan R to remove the element added into D at stage t [and restore the Θ computation]. Depending on the current stage announcement we would then have outcome either r or w. Since by assumption the outcome at s is g_{α_j}, we must have followed either Plan S or Plan R at t. □

Now we can prove that Ψ requirements are not injured by Plan G.

Lemma IX.10.3. *If σ is a Ψ node ($\Psi(G) \neq D$) on the true path and we go to outcome d after stage $s(\sigma)$, then the diagonalization will not be injured thereafter, i.e., the Ψ-use in G is preserved and the diagonalization witness x is not removed from D and so the Ψ requirement is satisfied.*

Proof. Suppose at stage $s > s(\sigma)$ we diagonalized at σ by putting x into D and so impose permanent restraint on D. First of all, x cannot be taken out of D at subsequent stages, since only nodes above σ could do so and then only if we follow either Plan S or Plan R at a node above σ. Plan S action would move us to the left for a contradiction and there are no outcomes r above σ by Lemma IX.10.1.

Next we claim that the G-use is also preserved. No node below or to the right of σ can add elements below this G-use into G after stage s, since their Δ-uses are defined to be fresh. The only worry is that some Θ node β with $\beta\hat{\ }g_\alpha$ above σ might follow Plan G (at stage $s_1 > s$) to add $\delta(y)-1$ into G for some changed u in W in order to correct some Δ axiom. The $\Delta(G;u)$ axiom being killed must have been enumerated before stage s, since its use was chosen fresh. So $W(u) = \Phi(A;u)$ at stage s.

By Lemma IX.10.2, the only circumstances under which we change G at stage s_1 in response to this change in W is that D has already changed by some other node τ below $\beta\hat{\ }g_\alpha$ following Plan S or Plan R at the previous stage t at which $\beta\hat{\ }g_\alpha$ was accessible. Such a τ cannot be above σ as we would then move to its left, so it must be to the right of, or below, $\sigma\hat{\ }d$.

Then there must be another Ψ' node σ' (below τ) which added a number into D after stage s (since at stage s we initialized all nodes to the right of τ, any change before s cannot be used in shuffling or removal) and the number is taken out by τ. However, σ' cannot be below, or to the right of, σ, as its witness would then be larger than s and so could not affect the value of $W(u)$ which agrees with $\Phi(A;u)$ at stage s. So we get a contradiction.

In another words, once we apply diagonalization for σ, we (automatically) implement restraints for A and D and hence to W's up to the part that we have coded in. So we can guarantee that there can be no change in W which makes us change G on the uses we have already seen, and in particular, the Ψ-use of G is preserved. $\square$

CHAPTER X
THREE THEOREMS ON N-REA DEGREES: PROOF-READERS AND VERIFIERS

This chapter will appear as a paper in Computability in Europe 2011.

X.1 Introduction

In this chapter, we first show that an n-**REA** degree is array recursive if and only if it is r.e. traceable. This gives an alternate proof that an n-**REA** degree has a strong minimal cover if and only if it is array recursive. Then we prove that an n-**REA** degree is strongly jump traceable if and only if it is strongly superlow. These two results both generalize corresponding equivalence theorems for the r.e. degrees. In these proofs, we provide an interesting technique to handle n-**REA** degrees, which also gives a new proof of an old result that every **FPF** n-**REA** degree is complete.

Our story begins with array recursive degrees. The term *array recursive* degrees actually comes from its complement, the *array nonrecursive* (**ANR**) degrees, which were defined in [DJS96] to generalize $\overline{\mathbf{GL_2}}$ degrees. In particular, **ANR** degrees share a lot of nice properties with $\overline{\mathbf{GL_2}}$ degrees. Examples of their common properties include the 1-generic bounding property, the cupping property and relative recursive enumerability (see [DJS96] and [CSh12]).

Recall in degree theory, a degree $\mathbf{a}$ is a *strong minimal cover* of $\mathbf{b}$ if $\mathcal{D}(< \mathbf{a}) = \mathcal{D}(\leq \mathbf{b})$, i.e., every degree strictly below $\mathbf{a}$ is below $\mathbf{b}$. It is a very difficult and long-standing question to characterize all the degrees which have strong minimal covers. The notion of **ANR** degrees comes into play in the following theorem:

Theorem X.1.1 ([DJS96])**.** *No* **ANR** *degree has a strong minimal cover.*

In particular, this lead to the first major progress on the question of strong minimal covers: In [Ish99], Ishmukhametov gave a characterization of the r.e. degrees with strong minimal covers by introducing the notion of r.e. traceability. Recall that W_e stands for the e-th r.e. set in any uniformly recursive indexing of r.e. sets, and $|X|$ denotes the number of elements in X.

Definition X.1.2. A degree $\mathbf{a}$ is *r.e. traceable* if there is a recursive function $f(n)$ such that for every function $g(n) \leq_T \mathbf{a}$, there is a recursive function $h(n)$ such that for every n, $g(n) \in W_{h(n)}$ and $|W_{h(n)}| \leq f(n)$.

We call $W_{h(n)}$ an *(r.e.) trace* for $g(n)$ (*admitting* f). It was shown in [Ish99] that:

Theorem X.1.3. *Every r.e. traceable degree has a strong minimal cover.*

Theorem X.1.4. *An r.e. degree is r.e. traceable if and only if it is array recursive.*

Together with Theorem X.1.1 one can easily get:

Corollary X.1.5. *An r.e. degree has a strong minimal cover if and only if it is array recursive.*

A natural generalization of r.e. degrees is the notion of iterates of the r.e. relation: A degree is 1-**REA** if it is r.e.; A degree is $(n+1)$-**REA** if it is r.e. in and strictly above an n-**REA** degree. In [Cai12] (see also Chapter III), we generalized Corollary X.1.5 to the n-**REA** degrees.

Theorem X.1.6. *An n-**REA** degree has a strong minimal cover if and only if it is array recursive.*

Interestingly, the proof of Theorem X.1.6 in [Cai12] does not follow from the corresponding generalization of Theorem X.1.4. In fact, it was not known whether Theorem X.1.4 has a generalization to the n-**REA** degrees. Greenberg asked whether it is true that an n-**REA** degree is array recursive if and only if it is r.e. traceable. Here we give a positive answer.

Theorem X.1.7. *An n-**REA** degree is r.e. traceable if and only if it is array recursive.*

In addition, it is not difficult to see that r.e. traceable degrees are downward closed in the Turing degrees, so we can actually get a little bit more from this result (compared to Theorem X.1.6):

Corollary X.1.8. *If a degree is below an array recursive n-**REA** one, then it has a strong minimal cover.*

Our second theorem is related to some recent research on two lowness notions, namely strongly jump traceable degrees and strongly superlow degrees. They were introduced aiming to give combinatorial characterizations of the *K-trivials*, an important notion in randomness. For details of the motivations and other examples, see [Nie08].

An *order function* is an unbounded nondecreasing recursive function. A degree $\mathbf{a}$ is *strongly jump traceable* if for every order function $r(n)$ there is a trace $W_{h(n)}$ admitting $r(n)$ such that $\varphi_n^A(n) \in W_{h(n)}$ whenever $\varphi_n^A(n)$ converges (it is easy to see that this definition does not depend on the choice of the set A in the degree $\mathbf{a}$). A degree $\mathbf{a}$ is *strongly superlow* if for every order function $r(n)$ there is a recursive limit approximation $\lambda(n, s)$ of A' (i.e., $\lim_s \lambda(n, s) = A'(n)$) such that the number of changes through the n-th column $\langle \lambda(n, s) \rangle_{s \in \omega}$ is bounded by $r(n)$. In other words, $\mathbf{a}$ is superlow with arbitrary order function as the recursive bound on the number of changes.

It is known that every strongly superlow degree is strongly jump traceable, and the other direction holds for the r.e. degrees (see [Nie08, Section 8.4]). In [Ng09], this is generalized to the n-r.e. degrees, which form a proper subclass of the n-**REA** degrees. In Section X.4, we continue to generalize this result to the n-**REA** degrees, and interestingly the proof has a flavor similar to that of Theorem X.1.7.

Theorem X.1.9. *An n-**REA** degree is strongly jump traceable if and only if it is strongly superlow.*

The third theorem is a generalization of Arslanov's Completeness Criterion that **FPF** r.e. degrees are complete ([Ars81]). This was already generalized to the n-**REA** degrees in [JLSS89], i.e., every **FPF** n-**REA** degree is above $\mathbf{0}'$.

Recall that a function f is *fixed-point-free* if $\varphi_e \neq \varphi_{f(e)}$ for any index e, and a degree is *fixed-point-free* (**FPF**) if it computes a fixed-point-free function. It is a classical result that a degree is **FPF** if and only if it computes a diagonally nonrecursive (DNR) function: f is DNR if $f(e) \neq \varphi_e(e)$ for any e with $\varphi_e(e)\downarrow$.

We will show the following theorem, which is a variation of the generalized completeness criterion in [JLSS89]:

Theorem X.1.10. *Suppose* **a** *is an* n-**REA** *degree, then the following are equivalent:*

1. **a** *computes a function* f *which is DNR;*
2. **a** *computes a function* f *which* dominates $\varphi_e(e)$, *i.e.,* $f(e) \geq \varphi_e(e)$ *holds for any* e *with* $\varphi_e(e)\downarrow$.

Note that (2) implies (and so is equivalent to) **a** being above $\mathbf{0}'$: For every $\varphi_e(e)$, one can effectively find an e' such that $\varphi_{e'}(e')$ converges to the number of steps in the computation of $\varphi_e(e)$ if it converges, and diverges if $\varphi_e(e)$ diverges. Then using $f(e')$ one can effectively tell whether $\varphi_e(e)$ converges or not. Moreover, it is trivial that (2) implies (1), and so we only need to show that (1) implies (2). We use Lewis' characterization of non-**FPF** degrees in terms of a weaker notion of traceability (see [Lew07]):

Theorem X.1.11. *For every degree* **a**, *the following are equivalent:*

1. **a** *is not* **FPF***;*
2. *there is a recursive function* $f(n)$ *such that for every function* $g(n)$ *recursive in* **a**, *there is a recursive function* $h(n)$ *such that* $|W_{h(n)}| \leq f(n)$ *for every* n, *and* $g(n) \in W_{h(n)}$ *for infinitely many* n.

For simplicity we say that **a** is *weakly r.e. traceable* if (2) holds (and similarly $W_{h(n)}$ is a *weak trace* for $g(n)$). Now we only need to show the following:

Theorem X.1.12. *Given an* n-**REA** *degree* **a**, *if for every function* f *recursive in* **a** *there are infinitely many* e *such that* $f(e) < \varphi_e(e)$, *then* **a** *is weakly r.e. traceable.*

In the setting of Theorem X.1.10, this shows that the negation of (2) implies the negation of (1). We will give a proof of the above theorem in Section X.5.

A final remark is that, the n-**REA** degrees seem to be strongly related to different notions of traceability. We hope that research along these lines may suggest other connections between the n-**REA** degrees and various combinatorial properties.

X.2 Basic Conventions and Notions

We use W_e^σ to denote the recursive enumeration with the index e and the oracle σ in $|\sigma|$ many steps. Without loss of generality, we can assume that no number x can be enumerated before step $x+1$, and we always let W_e^σ be a string of length $|\sigma|$.

It is worth noting here that we always regard W_e^σ as a string, so the notation $W_e^\tau \supset W_e^\sigma$ means that W_e^τ extends W_e^σ as a string, not as a set.

For A an m-**REA** set, we put $\emptyset = B_0 <_T B_1 <_T B_2 <_T \cdots <_T B_m = A$ where each B_{i+1} is r.e. in B_i. We say that B_i is at the *i-th level* of the enumeration.

Similarly any binary string σ which is assumed to be an initial segment of B_i is said to be at *the i-th level*, or simply an *i-th level string*. In this chapter, we always use the subscript of a string to denote its level. Given an m-**REA** set A with the sets B_i's as above, an i-th level string τ_i is *true* if it is an initial segment of B_i, and a string is *wrong* if it is not true.

When we say "let x ($\sigma \subset B_i$) be large (long) enough to have property P", we actually mean to find the least x (shortest σ) which has property P. Another y (τ) is *large enough* (*long enough*) with property P if it is larger than or equal to x (extends σ) in this setting.

For convenience, we use 0^t to denote the binary string of t zeros, and use W_e^t to denote $W_e^{0^t}$.

X.3 Proof of the First Theorem: Proof-readers

For A an m-**REA** set, we first give detailed proofs for $m = 2$ and $m = 3$ to supply some intuition. Based on this intuition, we then sketch a full proof for any m. The following Lemma will be very useful and we present it explicitly here for convenience. The proof is almost obvious by the Limit Lemma.

Lemma X.3.1. *If* $\mathbf{a}$ *is array recursive, then for every function $l(n) \leq_T$* $\mathbf{a}$, *there is a recursive function $\lambda(n, s)$ such that* $\lim_s \lambda(n, s)$ *exists and is greater than $l(n)$ for each n; in addition, the number of changes in each column of approximation $\langle \lambda(n, s) \rangle_{s \in \omega}$ is bounded by n.*

X.3.1 $m = 2$

Since r.e. traceable degrees have strong minimal covers and hence are array recursive, we only need to show that every array recursive 2-**REA** degree $\mathbf{a}$ is r.e. traceable, i.e., there is a recursive function $f(n)$ such that for every function $g(n) \leq_T \mathbf{a}$, there is an r.e. trace $W_{h(n)}$ for $g(n)$ admitting $f(n)$.

We pick $A \in \mathbf{a}$ a 2-REA set, i.e., $A = B_2 >_T B_1 >_T B_0 = \emptyset$ with $B_1 = W_i^{\emptyset} = W_i$ and $B_2 = W_j^{B_1} = W_j^{W_i}$.

Now for a function $g(n) = \varphi_e^A(n)$, we first define a function $l_0(n) \leq_T A$ as follows: Let $\sigma_2 \subset B_2$ be long enough to compute the value $g(n)$, i.e., $\varphi_e^{\sigma_2}(n) \downarrow= g(n)$. Then we let $\sigma_1 \subset B_1$ be long enough to enumerate all elements in σ_2, i.e., $W_j^{\sigma_1} \supset \sigma_2$ (note that it is not necessary and usually not the case that $W_j^{\sigma_1} \subset B_2$). Finally let $l_0(n)$ be large enough to enumerate all elements in σ_1, i.e., $W_i^{l_0(n)} \supset \sigma_1$.

Now by Lemma X.3.1 we have a recursive function $\lambda_0(n, s)$ whose limit is greater than $l_0(n)$. We first try to use values from this function to compute $g(n)$: let x be the first value greater than or equal to $l_0(n)$ in the column $\langle \lambda_0(n, s) \rangle_{s \in \omega}$. It is easy to see that W_i^x extends σ_1, but it might contain some wrong information after the σ_1 part. So if we simply use W_i^x in the next level enumeration and find $W_j^{W_i^x}$, we might get something which is wrong, since the wrong "tail" of W_i^x could enumerate something which is not in B_2. If we use this wrong string in the computation, we might get a wrong answer for $g(n)$.

The solution is to define another function $l_1(n) \leq_T A$ to "correct" or to "proof-read" the wrong information in W_i^x. Now we fix functions l_0 and λ_0 as above. For each n, let x be the first value greater than or equal to $l_0(n)$ in the corresponding column of λ_0. Now W_i^x might contain some error, so we let t be the first number such that $W_i^x(t) \neq B_1(t)$ (if such t does not exist, then let t be the first number enumerated into B_1 which is greater than $|W_i^x|$). The only possible situation is that $W_i^x(t) = 0$ and $B_1(t) = 1$. We then let $l_1(n)$ be large enough to enumerate t into B_1, i.e., $t \in W_i^{l_1(n)}$. This function l_1 is recursive in A, and so by Lemma X.3.1 there is a recursive function $\lambda_1(n, s)$ whose limit is greater than $l_1(n)$.

We say that W_i^x *requires proof-reading* and we call $W_i^{l_1(n)}$ (or any later enumeration which sees $t \in B_1$) a *proof-reader* for W_i^x.

With these two limit functions λ_0, λ_1 in hand, one can give a uniform enumeration of $W_{h(n)}$ as follows: Fix n; for each pair (p, q) such that $0 \leq p, q \leq n$, we go through the columns $\langle \lambda_0(n, s) \rangle_{s \in \omega}$ and $\langle \lambda_1(n, s) \rangle_{s \in \omega}$ respectively for the p-th change and the q-th change. If we cannot find either one, the computation simply diverges. Now let x be the value of $\lambda_0(n, s)$ after the p-th change and y be the value of $\lambda_1(n, s)$ after the q-th change. Take the longest common initial segment of W_i^x and W_i^y, say τ_1, then enumerate $\tau_2 = W_j^{\tau_1}$ and compute $\varphi_e^{\tau_2}(n)$. If this computation does not converge, do nothing. If it converges, then enumerate the value into $W_{h(n)}$. It is easy to see that if p is the number of changes in the column when it is the first time that the value of $\lambda_0(n, s)$ is greater than or equal to $l_0(n)$ and q is similarly the number of changes in the column when it is the first time that the value $\lambda_1(n, s)$ is greater than or equal to $l_1(n)$, then this computation must give us the correct answer $g(n)$. So $g(n) \in W_{h(n)}$ and $|W_{h(n)}| \leq (n+1)^2$, i.e., $W_{h(n)}$ is a trace for $g(n)$ admitting $(n+1)^2$.

X.3.2 $m = 3$

Now we have $A = B_3 >_T B_2 >_T B_1 >_T B_0 = \emptyset$, $B_1 = W_i^{B_0}$, $B_2 = W_j^{B_1}$ and $B_3 = W_k^{B_2}$.

Given a function $g(n) = \varphi_e^A(n)$, similarly we first define a function $l_{00}(n) \leq_T A$ as the number large enough to enumerate $\sigma_1 \subset B_1$, which is long enough to enumerate $\sigma_2 \subset B_2$, which is long enough to enumerate $\sigma_3 \subset B_3$, which is long enough to compute $g(n)$.

Now we have, by Lemma X.3.1, a limit function $\lambda_{00}(n, s)$. Using the same idea as in the previous case, we get a function $l_{01}(n)$ and a limit approximation $\lambda_{01}(n, s)$ such that with λ_{00} and λ_{01} one can get a true initial segment τ_1 of B_1 which is longer than σ_1.

For a similar reason, now if we use τ_1 in the enumeration, it might produce something with a wrong tail, and so we cannot use $W_j^{\tau_1}$ in the next step enumeration. Then we need another (second-level) proof-reading process. The problem is that now we are at the second level of the enumeration and a proof-reader of $W_j^{\tau_1}$ requires some true initial segment of B_1, which is not recursively given.

We fix such λ_{00} and λ_{01}, and we define a new function $l_{10}(n)$: for each n, use the first values in the approximations $\lambda_{00}(n, s)$ and $\lambda_{01}(n, s)$ which are greater than or equal to $l_{00}(n)$ and $l_{01}(n)$ respectively, and find τ_1 as above, then let $l_{10}(n)$ be large enough to enumerate some ξ_1 which is long enough to enumerate a proof-reader for $W_j^{\tau_1}$. With a limit function $\lambda_{10}(n,s)$, a similar recursive enumeration process gives us a different η_1 instead of ξ_1 and this η_1 itself needs proof-reading, i.e., it is long enough but may contain some wrong information which we cannot use in the next level enumeration. So we have another function l_{11} (and the corresponding limit function λ_{11}) which is large enough to enumerate a string at the first level to proof-read η_1 and give a true initial segment of B_1, which is long enough to enumerate some initial segment of B_2 which proof-reads $W_j^{\tau_1}$. After this second level proof-reading process, we have a true initial segment of B_2 and this initial segment is long enough to enumerate some string at the third level to compute $g(n)$.

So finally we have four functions recursive in A and four corresponding recursive approximations. It is easy to find a trace $W_{h(n)}$ for $g(n)$ in the same way and the recursive bound for the number of elements in $W_{h(n)}$ is $(n+1)^4$.

X.3.3 General case

With the above ideas in mind, we sketch the proof.

Now we have $A = B_m >_T B_{m-1} >_T \cdots >_T B_0 = \emptyset$ and each $B_{i+1} = W_{e_i}^{B_i}$. For simplicity of notions, we write W_i instead of W_{e_i}. We can show the following claim by induction on $i \in [1, m-1]$:

Claim X.3.2. *To get a true and long enough τ_i in the procedure of computing $g(n)$, one needs 2^i functions recursive in A in its proof-reading process.*

The base case is §X.3.1. For the inductive step, to get a true and long enough τ_i, we need two strings at the i-th level: a recursively-generated one ($W_{i-1}^{\tau_{i-1}}$) which is long enough but may contain errors in its tails, and a proof-reader ($W_{i-1}^{\xi_{i-1}}$) which corrects the first mistake in the first string. These two $i-1$-th strings τ_{i-1} and ξ_{i-1} must be true and long enough. So by induction hypothesis, we need 2^{i-1} functions to generate each one. Therefore we need 2^i functions for a true and long enough τ_i.

In the end, note that to correctly compute $g(n)$, we only need a long enough string at the m-th level, so we need a true and long enough initial segment at the $m-1$-th level. Finally we have 2^{m-1} functions recursive in A and each has a corresponding recursive limit approximation as in Lemma X.3.1. The construction of $W_{h(n)}$ is analogous to these in §X.3.1 and §X.3.2, and the recursive bound for $|W_{h(n)}|$ is $f(n) = (n+1)^{2^{m-1}}$.

X.4 Proof of the Second Theorem: Verifiers

We will follow the same strategy: we first present a detailed proof for $m = 2$ and sketch a proof for $m = 3$, then the general case will be clear by following the same pattern. We again do not try to write out a detailed full proof, because it is neither necessary to write out nor easy to read.

We will need the following lemma, which is quite easy to prove.

Lemma X.4.1. *A degree a is strongly jump traceable if and only if for every order function $r(n)$ and every partial function $\varphi_e^A(n)$ there is a trace $W_{h(n)}$ admitting $r(n)$ such that if $\varphi_e^A(n)\downarrow$ then $\varphi_e^A(n) \in W_{h(n)}$.*

X.4.1 $m = 2$

We only need to show that strongly jump traceability implies strongly superlowness. Following the same notion, let $B_1 = W_i$ and $A = B_2 = W_j^{W_i}$. To show that A is strongly superlow, we need to give a limit computation of A' and guarantee that the number of changes in the approximation can be bounded by any given order function $r(n)$.

We first define a partial function $\varphi_{e_2}^A(n)$ as follows: given n, we try to compute $\varphi_n^A(n)$: If it diverges then $\varphi_{e_2}^A(n)$ diverges; if it converges then let $\varphi_{e_2}^A(n)$ be the initial segment $A \upharpoonright u$ where u is the use of the computation. Applying Lemma X.4.1, we can get a trace $W_{h_2(n)}$ for $\varphi_{e_2}^A(n)$ admitting some recursive $r_2(n)$ which we will specify later. For each σ enumerated into $W_{h_2(n)}$, we can use it as an oracle and try to compute $\varphi_n^\sigma(n)$. If it converges then we might guess that $\varphi_n^A(n)$ converges. However, it is possible that some wrong σ enumerated into $W_{h_2(n)}$ makes $\varphi_n^{(\cdot)}(n)$ converge and so we might have a wrong guess on whether $\varphi_n^A(n)$ converges.

Now the solution is to use a *verifier* of such σ at a lower level of the enumeration. Given any σ at the second level, let $B_1 \upharpoonright u$ be the initial segment of B_1 which is long enough to enumerate a true initial segment of $A = B_2$ of length $|\sigma|$, and we call such $B_1 \upharpoonright u$ a *verifier* of σ. We define another partial function $\varphi_{e_1}^A(n,k)$ as follows: Enumerate $W_{h_2(n)}$ and wait for the k-th element to appear. If such element does not appear then $\varphi_{e_1}^A(n,k)$ diverges; if the k-th element is σ, then let $\varphi_{e_1}^A(n,k)$ be the verifier of such σ.

Then by Lemma X.4.1 again we have a trace $W_{h_1(n,k)}$ for $\varphi_{e_1}^A(n,k)$ admitting some recursive $r_1(n,k)$ which we will also specify later. The intuition is that, for each σ enumerated into $W_{h_2(n)}$ as the k-th element, we need to enumerate a verifier of it in $W_{h_1(n,k)}$ for us to believe that σ is true.

To give a limit computation of A', we shall have, at each stage s, a guess as to whether $n \in A'$, and guarantee that our guess is eventually correct. At stage s, we can enumerate $W_{h_2(n)}$ up to s steps and let $\sigma^0, \sigma^1, \ldots, \sigma^i$ be all the elements appeared in $W_{h_2(n)}$, in the enumeration order. For each σ^j, we can enumerate $W_{h_1(n,j)}$ up to s steps. For each τ enumerated, we say that it is *verified* if $\tau \subset W_i^s$, i.e., it is the initial segment of our current guess as to $W_i = B_1$.

We say that σ^j is *verified* (at stage s) if for the longest verified τ in $W_{h_1(n,j)}^s$, we have $\sigma^j \subset W_j^\tau$, i.e., σ^j looks like a correct initial segment of A at this stage. We also call $W_{h_1(n,j)}$ the *verifier set* for σ^j.

So at this stage s, we guess that n is in A' if there is such a verified σ^j with $\varphi_n^{\sigma^j}(n)\downarrow$. Now we need to show two facts: 1, this is a limit computation of A'; 2, we can arrange r_1 and r_2 to make the number of changes in each column be bounded by any given order function $r(n)$.

For the first claim, if $\varphi_n^A(n)$ converges, then eventually a true initial segment σ of A which is long enough to make $\varphi_n(n)$ converge will be enumerated and it will eventually be verified (though not necessarily by its verifier), since eventually only true initial segments of B_1 are verified at the first level and the longest such is long enough to verify that σ is a true initial segment of A by our construction. In the other direction, if $\varphi_n^A(n)$ diverges, then for any σ enumerated into $W_{h_2(n)}$, eventually it cannot be verified, since the longest true initial segment τ in its verifier set $W_{h_1(n,j)}$ is long enough to see that σ is wrong.

Given a σ in $W_{h_2(n)}$ as the j-th element, we know that $|W_{h_1(n,j)}| \leq r_1(n,j)$ and so we switch our mind at most $r_1(n,j)$ times for this σ. Then the total number of changes in the approximation of $A'(n)$ is bounded by $r_1(n,0) + r_1(n,1) + \cdots + r_1(n, r_2(n)) \leq r_2(n) r_1(n, r_2(n))$ (the pairing function is monotone).

Now in order to make $r_2(n) r_1(n, r_2(n)) \leq r(n)$, we can first find an order function r_2 such that $r_2(n) \leq \sqrt{r(n)}$, and we pick an order function r_1 such that $r_1(n, r_2(n)) \leq \sqrt{r(n)}$. The number of changes in the limit approximation above is then bounded by $r(n)$.

X.4.2 $m = 3$

To simplify our notations (and to save some letters for other uses) we have $A = B_3 = W_3^{B_2}$, $B_2 = W_2^{B_1}$ and $B_1 = W_1$.

We first define a partial function $\varphi_{e_3}^A(n)$ which outputs an initial segment of A which is long enough to compute $\varphi_n^A(n)$. Then we have a trace $W_{h_3(n)}$ for this partial function. Similarly we define $\varphi_{e_2}^A(n,k)$: for the k-th element σ_3 enumerated into $W_{h_3(n)}$, we output its verifier τ_2 at the second level, and we also get a trace $W_{h_2(n,k)}$ for it.

Now for any such verifier, we again need to verify it by a string at the first level. We define another $\varphi_{e_1}^A(n,k,l)$: for the k-th element σ_3 enumerated into $W_{h_3(n)}$ and for the l-th element τ_2 enumerated into $W_{h_2(n,k)}$, we output a verifier of τ_2 at the first level of the enumeration. Then we can get a trace $W_{h_1(n,k,l)}$ for it admitting $r_1(n,k,l)$.

Our limit computation is as follows: At each stage s, we enumerate every trace up to s steps. We say that a first level string σ_1 is *verified* (at stage s) if $\sigma_1 \subset W_1^s$. An i-th level string σ_i is *verified* if $\sigma_i \subset W_i^{\sigma_{i-1}}$ where σ_{i-1} is one of the longest verified strings in the verifier set of σ_i. We guess that $\varphi_n^A(n)$ converges if there is a verified third level string which converges at $\varphi_n^{(\cdot)}(n)$.

Similarly one can show that this is a limit computation for A': we prove by induction that eventually true initial segments and only true initial segments at each level can be verified: at the first level this is obvious; for level i, we eventually have, at level $i-1$, only true initial segments that are verified by induction hypothesis, and then according to our construction, the longest such true σ_{i-1} in each verifier set extends the verifier of the corresponding i-th level string σ_{i-1} is verifying, therefore only true initial segments at the i-th level can be verified in a long run.

In addition, it is easy to see that the number of changes in the column n of this limit computation is bounded by $r_3(n) \times r_2(n, r_3(n)) \times r_1(n, r_3(n), r_2(n, r_3(n)))$. In order to make it bounded by $r(n)$ we can easily make each term in the product bounded by the cubic root of $r(n)$.

X.4.3 General case

With the discussion above, the construction and the verification in the general case are clear. Shortly speaking, we will define partial functions $\varphi_{e_i}^A$ with their traces level by level. In the limit approximation, we guess at stage s that $\varphi_n^A(n)$ converges if there is a verified m-th level string which makes $\varphi_n^{(\cdot)}(n)$ converge. The number of changes is bounded by a product of m terms and one can bound each by the m-th root of $r(n)$. The details of the proof are omitted.

X.5 Proof of the Third Theorem: Proof-readers again

In this section we prove Theorem X.1.12. With the idea of the previous two theorems, we only sketch the proof for $m = 2$ and leave the other parts to the reader.

Again we let $B_1 = W_i$ and $A = B_2 = W_j^{W_i}$. Given $g(n) = \varphi_e^A(n)$, we first define $l_0(n)$ exactly the same way as in the proof of the first theorem in Section X.3, i.e., it is large enough to enumerate some first level string which is long enough to enumerate a second level initial segment of B_2 which computes the value $g(n) = \varphi_e^A(n)$. By the property given, we know that $\varphi_n(n) > l_0(n)$ infinitely often, and so at these n's where $\varphi_n(n) > l_0(n)$ we may expect to use $\varphi_n(n)$ to give a correct value of $g(n)$. However, we have a similar problem that $\varphi_n(n)$ may enumerate some string with a "wrong tail", and so we need to define a proof-reader for it.

The trick here is that $\varphi_n(n)$ is not total, so we need to modify our strategy. Recursively in A, we define a sequence x_i as follows: let x_0 be the first number n that $\varphi_n(n)$ converges and is greater than $l_0(n)$; given x_i, let x_{i+1} be the first number $n > x_i$ that $\varphi_n(n)$ converges and is greater than $l_0(n)$. It is easy to see that this sequence is infinite, strictly increasing and recursive in A. Then we define a function $l_1(n)$ recursively in A such that $l_1(n)$ is large enough to enumerate a proof-reader for $\varphi_{x_n}(x_n)$. Then similarly $\varphi_n(n) > l_1(n)$ infinitely often.

We can now find a weak trace $W_{h(n)}$ for $g(n)$ as follows: Given n, we try to compute $\varphi_n(n)$ and in addition, every $\varphi_m(m)$ for $m \leq n$ simultaneously. If $\varphi_n(n) \downarrow = x$ and if any $\varphi_m(m)$ converges to y, then we use W_i^x with a proof-reader W_i^y (i.e., find their longest common initial segment) to enumerate a second level string to compute $\varphi_e^{(\cdot)}(n)$, and finally enumerate the value into $W_{h(n)}$ if it converges. It is easy to see that $|W_{h(n)}| \leq n$.

Then we need to show that infinitely often $g(n) \in W_{h(n)}$: for these n's with $\varphi_n(n) > l_1(n)$, it is easy to see that $n \leq x_n$ and $g(x_n)$ is in $W_{h(x_n)}$ by a correct proof-reading process.

In general, for an m-**REA** degree A, we need a similar proof-reading process as in Section X.3, and our construction gives a recursive bound $f(n) = n^{2^{m-1}-1}$ for the weak r.e. traces.

CHAPTER XI
A NOTE ON STRONG MINIMAL COVERS: TREE TRACEABILITY

XI.1 Introduction

In this chapter, we discuss a notion of tree traceability which extends r.e. traceability introduced in [Ish99]. We show that a tree traceable degree has a strong minimal cover if and only if it is array recursive. This result gives new examples of degrees with strong minimal covers. In addition, we discuss the connections between this notion and an old question of Yates.

In classical recursion theory, minimal degrees and minimal covers are among the most interesting topics, typically because that they are naturally definable from the partial order of Turing degrees (see definitions in §I.1.1).

By studying minimal degrees and minimal covers, one can get some interesting definability results. A typical example is that all arithmetic degrees are definable in the language of partial orders in the Turing degrees by analyzing Sacks' minimal degree construction (see [JSh84]).

Spector first proved the existence of minimal degrees, and he noted that a natural relativization gives a minimal cover over any given degree, but the construction does not yield a strong minimal cover. It becomes a very interesting and difficult problem to classify all the degrees that have strong minimal covers. Simply saying, we are interested in the following class of degrees:

$$\mathcal{C} = \{\mathbf{a} : \mathbf{a} \text{ has a strong minimal cover}\}.$$

The problem is actually two-fold: we are looking for properties which guarantee the existence of a strong minimal cover, and also for properties which make the degree have no strong minimal covers. For the latter, actually all degrees we know that do not have strong minimal covers do have the *cupping property*: $\mathbf{a}$ has the cupping property if for every $\mathbf{b} > \mathbf{a}$, there is a $\mathbf{c} < \mathbf{b}$ such that $\mathbf{b} = \mathbf{a} \vee \mathbf{c}$. Typical examples of the cupping property results are the following two (see definitions in Chapter I):

Theorem XI.1.1 (Kučera [Ku94]). *Every* **PA** *degree has the cupping property.*

Theorem XI.1.2 (Downey, Jockusch, Stob [DJS96]). *Every* **ANR** *degree has the cupping property.*

From another point of view, if we try to find degrees with strong minimal covers, then we can automatically assume that the degree is array recursive (i.e., not **ANR**) and not **PA**. Interestingly, if we strengthen both conditions, they imply the existence of strong minimal covers. Recall that being hyperimmune-free is a strengthening of being array recursive and being **FPF** is a natural weakening of being **PA**. We have the following:

Theorem XI.1.3 (Lewis [Lew07]). *Every hyperimmune-free degree which is not* **FPF** *has a strong minimal cover.*

However, it is not known whether either strengthening is sufficient. Another general result about having a strong minimal cover is related to r.e. traceability, also known as weak recursiveness. A degree $\mathbf{a}$ is *r.e. traceable* if there is a recursive function f such that for every function $g \leq_T \mathbf{a}$, there exists a recursive function h such that for every n, $|W_{h(n)}| \leq f(n)$ and $g(n) \in W_{h(n)}$, where W_e denotes the e-th recursively enumerable set and $|X|$ denotes the cardinality of the set X.

Theorem XI.1.4 (Ishmukhametov [Ish99]). *Every r.e. traceable degree has a strong minimal cover.*

Moreover, we have the following:

Theorem XI.1.5 (Ishmukhametov [Ish99]). *An r.e. degree is array recursive if and only if it is r.e. traceable.*

Therefore in the r.e. degrees, the notion of r.e. traceability or array recursiveness completely characterize the degrees with strong minimal covers.

Another interesting and related question is the following:

Question XI.1.6 (Yates). *Does every minimal degree has a strong minimal cover?*

It is well-known that minimal degrees are neither **ANR** nor **PA**, so neither of the cupping property theorems above can be used to produce a negative answer. In addition, a lot of minimal degree constructions give minimal degrees which satisfy conditions in either Theorem XI.1.3 or XI.1.4, hence they all have strong minimal covers. It seems that it is the highly-specified tree constructions we use to produce these minimal degrees that make them have strong minimal covers. In this chapter we partially confirm this assertion by showing that some certain types of tree constructions automatically produce degrees with strong minimal covers. In particular, this narrows down the possible ways of constructing a minimal degree without strong minimal covers, i.e., if we want to construct such a degree, we must avoid these certain types of tree construction.

In Section XI.3 we define a new notion of *tree traceable* degrees and continue our discussion there.

XI.2 Definitions and Notions

We write σ^- to denote the string formed by removing the last number from σ. We can regard each Turing functional Φ as a function on the set of finite strings, and we call a string σ *Φ-proper* if $\Phi(\sigma^-) \subsetneq \Phi(\sigma)$.

As usual, $\omega^{<\omega}$ (resp. ω^ω) denotes the set of all finite (resp. infinite) strings and $2^{<\omega}$ (resp. 2^ω) denotes the set of all finite (resp. infinite) binary strings.

We use $\leq_{llex}$ to denote the length-lexicographic order on the strings.

There are commonly two different representations of trees: one as (downward closed) subsets of $\omega^{<\omega}$, the other as (possibly partial) functions from $\omega^{<\omega}$ to $\omega^{<\omega}$ with order-preserving and nonorder-preserving properties. In a specific construction, one notion might be easier to implement than the other.

In this chapter, we will mainly use the notion of function-like trees and occasionally we will (automatically) transform these to set-like trees in the discussion.

A *tree* is a partial function $T : \omega^{<\omega} \to \omega^{<\omega}$ with the following properties:

1. (Length-lexicographic order property) $(\sigma \leq_{llex} \tau \wedge T(\tau)\downarrow) \Rightarrow T(\sigma)\downarrow$.
2. (Order preserving) $(T(\sigma)\downarrow \wedge T(\tau)\downarrow) \Rightarrow (\sigma \subsetneq \tau \Leftrightarrow T(\sigma) \subsetneq T(\tau))$.
3. (Nonorder preserving) $(T(\sigma)\downarrow \wedge T(\tau)\downarrow) \Rightarrow (\sigma|\tau \Leftrightarrow T(\sigma)|T(\tau))$.

In particular, the first property says, if $T(\tau)$ converges then $T(\sigma)$ converges for every initial segment σ of τ; and if $T(\sigma * j)$ converges then $T(\sigma * i)$ converges for every $i < j$.

We call a tree *full binary tree* if the domain is exactly $2^{<\omega}$.

Given two trees T and S, we say S is a *subtree* of T if every node on S is a node on T. If $\tau = T(\sigma)$ is a node on T, then the *full subtree* T' of T above τ is defined as $T'(\xi) = T(\sigma * \xi)$ for $\xi \in \omega^{<\omega}$.

XI.3 Tree Traceable Degrees and Summary of Results

Our definition of tree traceability was partially motivated by the coding idea in Proposition XI.3.6 attempting to find a minimal degree without strong minimal covers. It is a "tree version" of traceability and so gets its name.

Given a function $f : \omega^{<\omega} \to \omega$, f is *pathwise nondecreasing* if for every pair $\sigma \subset \tau$, if both $f(\sigma)$ and $f(\tau)$ converge, then $f(\sigma) \leq f(\tau)$, i.e., longer strings compute larger values.

Definition XI.3.1. A degree $\mathbf{a}$ is *tree traceable* if there is a pathwise nondecreasing partial recursive function $f(\sigma)$ and a set $A \in \mathbf{a}$ such that for any Turing functional Φ with $\Phi(A) = A$, there is a partial recursive tree T such that the following holds:

1. $A \in [T]$;
2. for each σ on T, let σ' be the parent of σ on T, then there are at least two Φ-proper nodes in $[\sigma', \sigma]$;
3. for each σ on T, if it is not a leaf, then $f(\sigma)$ converges and the number of children of σ on T is bounded by $f(\sigma)$.

The second property above might seem strange, but one can regard it as a weakening of the requirement that every node on T is Φ-proper. The main construction in Section XI.5 will provide a more concrete idea about this property.

We call f a *tracing function* for $\mathbf{a}$, and T a *tracing tree* for Φ (with respect to f). Also note that in the definition we actually do not need the fact that A is binary, and any string α in the same degree also works, since we can use the canonical coding between binary strings and strings as a translation.

This notion extends r.e. traceability:

Proposition XI.3.2. *Every r.e. traceable degree is tree traceable.*

Proof. Let f be the recursive function in the definition of r.e. traceability of degree $\mathbf{a}$ and pick any set $A \in \mathbf{a}$. For every Φ with $\Phi(A) = A$, we need to find a partial recursive tree T as a tracing tree for Φ.

Define a function $g \leq_T A$ inductively as follows: let $g(0)$ be (the code of) the empty string and let $g(k+1)$ be the first initial segment of A extending $g(k)$ with a properly longer Φ image, i.e. $g(k+1) \supsetneq g(k)$ is the first such that $\Phi(g(k+1)) \supsetneq \Phi(g(k))$. By r.e. traceability we have a recursive h such that for every n, $g(n) \in W_{h(n)}$ and $|W_{h(n)}| \leq f(n)$.

We can, of course, assume that each element in $W_{h(n)}$, as a code of a finite string, is at the n-th level of Φ computation as $g(n)$ above and also extends a string in $W_{h(n-1)}$. This uniform enumeration naturally gives us a tree: let the empty string be the root, and given any node at level k, its immediate successors are these strings in $W_{h(k+1)}$ that extend it. Assuming that f is nondecreasing, it is easy to see that $f'(\sigma) = f(|\sigma|+1)$ is a tracing function for A and the tree generated by $\langle W_{h(n)} \rangle$ above is a tracing tree for Φ with respect to f. Here every node on the tree is Φ-proper. $\square$

It is easy to show that r.e. traceable degrees are automatically array recursive. So one might naturally ask whether tree traceables have the same property. Interestingly, the answer is no.

Proposition XI.3.3. *There is a tree traceable degree which is array nonrecursive.*

Proof. See Section XI.6. $\square$

Our main result is that, array nonrecursiveness is the only restraint for tree traceable degrees to have strong minimal covers, i.e.,

Theorem XI.3.4. *Every array recursive tree traceable degree has a strong minimal cover.*

Proof. See Section XI.5. $\square$

Combining Theorem XI.3.4 and Theorem XI.1.2, we get:

Corollary XI.3.5. *A tree traceable degree has a strong minimal cover if and only if it is array recursive.*

This also leads to the question whether all array recursive tree traceable degrees are r.e. traceable, i.e., whether the result of Theorem XI.3.4 is already covered by Theorem XI.1.4. In fact one can find an example of a minimal degree which is not r.e. traceable but tree traceable. In addition, one can guarantee that this minimal degree is hyperimmune, and so our result is not covered by Theorem XI.1.3 either.

Proposition XI.3.6. *There is a minimal degree which is hyperimmune, tree traceable and not r.e. traceable.*

Proof. See Section XI.7. $\square$

Then one might ask whether all minimal degrees are tree traceable, or whether all minimal degrees which have strong minimal covers are tree traceable. If so the notion would be a nice classification of minimal degrees with strong minimal covers. Unfortunately, though not surprisingly, the answer is no.

Proposition XI.3.7. *There is a minimal degree which is not tree traceable.*

Proof. See Section XI.8. □

In fact, by some results in [Lew07] one can argue that this minimal degree is hyperimmune-free and not **FPF**, and hence has a strong minimal cover.

In the last section, we end with some further discussions and questions.

XI.4 Tree Systems

Before we prove our main theorem, we want to introduce a notion of *tree systems* which we will use in our proof. The idea was motivated by Lewis' proof of Theorem XI.1.3. In his proof ([Lew07]), Lewis used a notion of *tree basis* which was defined to be degree $\mathbf{a}$ with the following property:

(†): for every full binary tree recursive in $\mathbf{a}$, there is a full binary subtree T' of T such that every path on T' computes $\mathbf{a}$.

Lewis showed that such a degree has strong minimal covers by using a standard Spector minimal cover construction. Our construction in the next section can be modified to show that every tree traceable array recursive degree is a tree basis, but here we present the proof in the form of tree systems as we want to emphasize that it is possible to have this property (†) only for the trees through the minimal cover construction, and these trees might have some special properties which guarantee nice subtrees as in (†). A typical example is the main theorem in [Cai12] (Theorem III.5.1).

A tree system is intuitively a "tree of trees". One could have different definitions for them in different specific situations, just like in the case of trees. So the following definition is just a suitable choice for our construction, but not the only possible definition of tree systems. In particular, we embed recursiveness into our definition just for simplicity of notions.

Definition XI.4.1. A *tree system* is a pair (T, S) where T is a partial recursive tree $\omega^{<\omega} \to 2^{<\omega}$, and S is a function from $2^{<\omega}$ (viewed as the world containing the nodes on T) to finite binary trees with the following properties:

1. for all τ, $S(\tau)$ is a finite tree with domain 2^n for some n, and n is called the *height* of the tree $S(\tau)$.
2. for every $\tau \supset \tau'$, if $S(\tau')$ has height n, then $S(\tau)$ has height at least n, and $S(\tau) \restriction_{dom(S(\tau'))} = S(\tau')$, i.e., $S(\tau)$ must preserve the tree structure of $S(\tau')$;

In this definition, T and S do not seem related, but usually in our construction, S will be regarded as being defined for the nodes of T only and extended naturally to other strings by the second property above in the definition.

In tree systems, we will restrict the use of lower case Greek letters: we use the letter σ to only denote the strings in the domain of T, and τ to denote the strings in the range of T and domain of S; μ denotes the strings in the domain of $R = S(\tau)$ and ρ (together with η) denotes the strings in the range of such an R. When we use other Greek letters, they could mean any string, i.e., they are not necessarily on the tree or the system.

In the definition, T is a partial recursive tree and sometimes we also regard T as the set of its nodes. From this viewpoint, T is also an r.e. subset of $2^{<\omega}$ with the property that in the recursive enumeration, every new element entering T is a leaf at that point. (Lewis called such set T *weakly r.e. trees.*) So in this chapter, we will sometimes confuse the notion of function-like tree T and the corresponding set-like tree T.

Note that $S(A) = \cup_{\tau \subset A} S(\tau)$ defines a (possibly partial) binary tree recursive in A, and actually every binary tree recursive in A can be viewed as an $S(A)$ in some tree system (T, S). Note that $S(A)$ is a full binary tree if and only if the height of $S(\tau)$ is eventually increasing for these $\tau \subset A$.

We first make a definition which is an analog of full subtrees:

Definition XI.4.2. Given a tree system (T, S), a node τ on T and a node ρ on $S(\sigma)$. The *full subtree system* of (T, S) above (τ, ρ) is defined to be (T', S') where T' is the full subtree of T above τ and $S'(\tau')$ is the full subtree of $S(\tau')$ above ρ for every τ' on T'.

To prove the theorem in the next section, we also give definitions which are analogs of splitting trees in the standard minimal degree construction and provide two technical lemmas.

Definition XI.4.3. A tree system is called *e-splitting* if for every node τ on T, all leaves of $S(\tau)$ pairwise e-split.

Lemma XI.4.4. *If (T, S) is an e-splitting tree system and $S(A)$ is a full binary tree, then for every B, a path on $S(A)$, $B \leq_T \varphi_e^B \oplus A$.*

Proof. Relativize the standard computation lemma for splitting trees, see also [Ler83, Lemma V.2.6]. ☐

To find a strong minimal cover, what we actually need is $B \leq_T \varphi_e^B$. For that we need a stronger property:

Definition XI.4.5. A tree system is called *totally e-splitting* if for every two incompatible nodes τ_0, τ_1 on T, all leaves of $S(\tau_0)$ and $S(\tau_1)$ pairwise e-split.

Lemma XI.4.6. *If (T, S) is a totally e-splitting tree system and $S(A)$ is a full binary tree, then for every B, a path on $S(A)$, $B \leq_T \varphi_e^B$.*

Proof. We start to enumerate T (note that the nodes form an r.e. set), for every node τ on T, we try to compute φ_e^ρ for every leaf ρ on $S(\tau)$. If φ_e^ρ is compatible with φ_e^B, we know that the corresponding ρ is an initial segment of B, since every ρ' a leaf on other $S(\tau')$ or other leaves on $S(\tau)$ would give incompatible φ_e-images. ☐

XI.5 Main Theorem

In this section we prove Theorem XI.3.4. Given a tree traceable array recursive degree $\mathbf{a}$, we first pick a set $A \in \mathbf{a}$ as in Definition XI.3.1. We will construct a set $B \geq_T A$ and make sure that B is a strong minimal cover of A. We do this by constructing a sequence of tree systems $\langle T_i, S_i \rangle$ such that $\langle S_i(A) \rangle_{i \in \omega}$ is a nested sequence of full binary trees and B is the only path on every $S_i(A)$.

We will step by step handle the following requirements:

1. $P_e:\ \varphi_e^A \neq B$;
2. $Q_e:$ if φ_e^B is total, then either it is recursive in A, or $B \leq_T \varphi_e^B$.

So P_e guarantees that B is strictly above A and Q_e guarantees that B is a strong minimal cover of A. In general, we say a tree system (T, S) *forces* a property (requirement) P if P is satisfied for all $A \in [T]$ and all $B \in [S(A)]$. Since our list of requirements is countable, it is easy to see that we only need to give an initial tree system (T_0, S_0) (which forces $A \leq_T B$) and provide modules, given a tree system (T, S), to find the next tree system (T', S') such that $A \in [T']$, $S'(A)$ is a subtree of $S(A)$ and (T', S') forces one requirement above.

XI.5.1 Initial tree system

Our initial T_0 is the identity function on $2^{<\omega}$ (it is partial recursive, if viewed as a function from $\omega^{<\omega}$ to $2^{<\omega}$). For every $\tau \in 2^{<\omega}$, $S_0(\tau)$ maps $2^{|\tau|}$ to $2^{<\omega}$, sending each μ to $\tau \oplus \mu$.

It is easy to see that (T_0, S_0) forces $A \leq_T B$ for all $B \in [S_0(A)]$.

XI.5.2 Force $B >_T A$

In the standard minimal degree construction, forcing the minimal degree to be nonrecursive is automatically guaranteed by the Posner's Lemma. Here we could give an analog of the Posner's Lemma, but we want to present a direct construction to introduce how to work with tree systems.

Suppose we are given a tree system (T, S) and a requirement P_e. Take $\tau \subset A$ such that $S(\tau)$ has height at least one. Suppose ρ_0 and ρ_1 are any two incompatible nodes on $S(\tau)$. For example, let $\rho_i = S(\tau)(i)$ for $i = 0, 1$. At least one of them is incompatible with φ_e^A, if it is total. Without loss of generality we can assume that ρ_0 is one which is incompatible with φ_e^A, then by taking the subtree system of (T, S) above (τ, ρ_0), we have forced $B \neq \varphi_e^A$.

XI.5.3 Force minimality

We first briefly describe the construction: We can take an e-splitting tree system which is an analog of a splitting tree (if possible) and this splitting subtree system forces $B \leq_T \varphi_e^B \oplus A$. Then in order to force $B \leq_T \varphi_e^B$ we want to construct a totally e-splitting tree system (see Lemma XI.4.6). The key is, of course, tree traceability, which basically gives us a prediction on the number of children of any given node. Following this intuition, we will define a functional Φ and use the definition of tree traceability to narrow down the number of nodes that we need to work with. The final touch is to use array recursiveness to "regulate" the enumeration of nodes (see also [Cai12]) and give a tricky but nevertheless recursive procedure to get a totally e-splitting tree system.

Now given a tree system (T, S) and a requirement Q_e. As in a usual minimal cover construction, we first ask whether we can force φ_e^B to be partial or recursive in A:

$$\exists\tau \subset A \exists\rho \text{ on } S(\sigma) \forall\tau' \supset \tau, \tau' \subset A \forall\rho_0, \rho_1 \supset \rho, \rho_0, \rho_1 \text{ on } S(\tau') \neg[\rho_0|_e\rho_1].$$

That is, we ask whether there is an initial segment $\tau \subset A$ (without loss of generality we can assume that τ is on T) and a node ρ on $S(\tau)$ such that the full subtree system of (T, S) above $\tau, \rho)$ forces that there are no e-splittings above ρ on the tree $S(A)$, and so forces that φ_e^B is either partial, or recursive in A.

If the answer is yes, then of course we can take the full subtree as above and we have satisfied the requirement Q_e. If the answer is no, then we know that:

$$(\dagger): \ \forall\tau \subset A \forall\rho \text{ on } S(\tau) \exists\tau' \supset \tau, \tau' \subset A \exists\rho_0, \rho_1 \supset \rho, \rho_0, \rho_1 \text{ on } S(\tau')[\rho_0|_e\rho_1]$$

Basically it says that we can always find e-splittings on $S(A)$. Now we first define an e-splitting tree system (T^*, S^*): start from the root λ of T^* and without loss of generality we can assume that $S(\lambda)$ is of height 0. We enumerate λ into T^* and let $S^*(\lambda) = S(\lambda)$. By induction, suppose we have enumerated τ into T^* and defined $S^*(\tau)$ to be a finite binary tree of height k, then we search above τ on T^* for any τ' such that for every leaf ρ of $S^*(\tau)$, there is an e-splitting extending ρ on $S(\tau')$. For any first such τ' we find along the path (i.e., no $\tau'' \in (\tau, \tau')$ has the same property), we enumerate τ' into T^* as the next children of τ and define $S^*(\tau')$ to be a finite binary tree of height $k+1$ extending $S^*(\tau)$, each leaf of $S^*(\tau)$ extended by the two e-splitting nodes we find on $S(\tau')$.

By $(\dagger)$, we know that $A \in [T^*]$ and $S^*(A)$ is an infinite e-splitting binary tree. Next we want "thin out" T^* to T^{**} such that every τ on T^{**} has at most $f(\tau)$ many children of T^{**}, and for any child τ' of τ, $S^*(\tau')$ has height at least the height of $S^*(\tau)$ plus $f(\tau)$. Then we can apply the following lemma:

Lemma XI.5.1. *In the setting above, suppose $S^*(\tau)$ has height m and $\tau_1, \tau_2, \ldots, \tau_l$ (so $l \leq f(\tau)$) are children of τ on T^{**}. In addition, the height of each $S^*(\tau_i)$ is at least $m+l$. Then for every leaf ρ of $S^*(\tau)$, we can respectively find two extensions ρ_{i0}, ρ_{i1} of ρ on each $S^*(\tau_i)$ such that all $\rho_{ij} : i = 1, .., l; j = 0, 1$ pairwise e-split.*

Now with T^{**} (we haven't say anything about how to find T^{**} by tree traceability) and the lemma above, it is easy to see that we can construct a S^{**} such that (T^{**}, S^{**}) is a totally e-splitting tree system: once we have define $S^{**}(\tau)$ then for all the children $\tau_1, \ldots, \tau_l$ we can use the lemma to define $S^{**}(\tau_i)$ such that all leaves pairwise e-split.

The argument above has one crucial problem and we have to be very careful here: trees T^* and T^{**} will be partial recursive trees and there is no way to recursively list all the children of one given node. The only thing we can do is to enumerate the tree T^* or T^{**} and wait for the children to appear one by one, and we don't know when this process will stop. So if we only do the previous construction for part of the children when they appear, then later if a new child appears, we might have no way to define S^{**} to make it totally e-splitting.

Now we have to use the property that **a** is array recursive and by this we can "group up" all children of one node into a limited number of groups. Children in each group will be recursively enumerated at the same time and have S^{**} defined together by Lemma XI.5.1. Children in different groups will have their corresponding leaves e-split by a pre-selected space from some "pseudo-leaves" to leaves on the tree of their parent.

Define Φ

With (T^*, S^*) in hand, we define a Turing functional Φ and a level function lev as follows: Take the root λ of T^* and define $\Phi(\lambda) = \lambda$, $lev(\lambda) = 0$ (we can assume that $f(\lambda)$ converges). Suppose we have defined $\Phi(\tau) = \tau$, $lev(\sigma) = k$ and the height of $S^*(\tau)$ is m, then we search above τ (in length-lexicographical order) for strings τ' such that $S^*(\tau')$ has height at least $m + f(\tau) + [\log_2(k+3)] + 1$ and $f(\tau')$ converges.

For such τ', we define $\Phi(\tau') = \tau'$ (and $\Phi(\tau'') = \tau$ for every $\tau'' \in (\tau, \tau')$), and we also define $lev(\tau') = k + 1$. The last $[\log_2(k + 3)] + 1$ levels on each $S^*(\tau')$ will be used as a coding space to separate children nodes of different groups.

Note that by the definition of tree traceability, there are infinitely many initial segments α of A such that $f(\alpha)$ converges. So for this Φ we have $\Phi(A) = A$.

Apply tree traceability and array recursiveness

For this Φ, we have a Φ-tracing partial recursive tree T^{**} by the definition of tree traceability. Note that T^{**} is not necessarily a subtree of T^* as function-like trees, but a subtree as set-like trees. The number of children of any node τ on T^{**} is bounded by $f(\tau)$. As we discussed before, the problem is that we can only enumerate these children one by one without knowing which one is the last one. So we need array recursiveness to get a final tree T'.

First define a function $k(n) \leq_T A$ as follows: fix the recursive enumeration of T^{**} and let α_i be the $i+1$-st initial segment of A on T^{**}; let $k(0)$ be the time when the root (i.e., α_0) enters T^{**} in the enumeration; similarly let $k(n)$ be the time when α_n enters T^{**}. By array recursiveness, we know that the modulus function of K dominates this $k(n)$, and so by the limit lemma there is a recursive function $\lambda(n, s)$ such that for each n, $\lim_{s\to\infty} \lambda(n, s)$ exists and is greater than $k(n)$. In addition, the number of changes in each column $\langle \lambda(n, s)\rangle_{s\in\omega}$ is bounded by n.

Using this function $\lambda(n, s)$, we define a group function grp and a new partial recursive tree T' from T^{**} as follows: First for simplicity put α_0 into T' as the root and let $grp(\alpha_0) = 0$. Suppose we have put τ into T' and τ is originally at level k of T^{**} (note that this level is different from the function lev we define earlier). Now look through the column $\langle \lambda(k + 1, s)\rangle_{s\in\omega}$ for changes in value. For each change, say the current value is x and the value has changed t times, we then enumerate T^{**} up to x steps, and put all children of τ we see into the new tree T'. Moreover, for each child τ', we let $grp(\tau') = t$.

It is easy to see that each α_n is on the new tree T'. Now using this tree T', properties in Definition XI.3.1 and Lemma XI.5.1 we can build a new tree system (T', S').

Define S'

First we analyze some properties of parent-child relationship on T'.

Suppose τ is the parent of τ' on T', hence also on T^{**}, and τ is at level k on T'. By property (3) we know that $f(\tau)$ converges and by property (2), there are at least two Φ-proper nodes in $[\tau, \tau']$. Say they are ξ_1, ξ_2 and ξ_2 is extending ξ_1. It is easy to prove by induction that $lev(\xi_1)$ is at least the level of τ on T', and by the definition of Φ, we know that the height of $S^*(\xi_2)$ is at least the height of $S^*(\xi_1)$ plus $f(\xi_1)+[\log_2(lev(\xi_1)+3)]+1$. The latter number is greater than or equal to $f(\tau)+[\log_2(k+3)]+1$, since f is pathwise nondecreasing. Therefore finally the height of $S^*(\tau')$ is at least the height of $S^*(\tau)$ plus $f(\sigma)+[\log_2(k+3)]+1$.

In the setting above, on such $S^*(\tau')$, we call the nodes at co-level $[\log_2(k+3)]+1$ the *pseudo-leaves* on $S^*(\tau')$.

We will define $S'(\tau)$ to be a subtree of $S^*(\tau)$, and every leaf of $S'(\tau)$ is a pseudo-leaf of $S^*(\tau)$. We begin the definition by picking α_1 and any pseudo-leaf ρ of $S^*(\alpha_1)$ and define $S'(\alpha_1)$ to be the single tree of height 0 with node ρ.

For τ at level k of T, suppose we have defined $S'(\tau)$ and every leaf of $S'(\tau)$ is a pseudo-leaf of $S^*(\tau)$. Recursively list all its children $\tau_1,\dots,\tau_l$ in group t (i.e., $grp(\tau_1)=\dots=grp(\tau_l)=t$). Pick any leaf ρ of $S'(\tau)$. There are $[\log_2(k+2)]+1$ levels from pseudo-leaves to leaves and so there are at least $k+2$ leaves on $S^*(\tau)$ extending ρ. Note that the number of groups for all children of τ is bounded by $k+2$, so we have enough leaves here to code the group information. We code group number t by taking the $t+1$-st leaf ρ' extending ρ on $S^*(\tau)$. Note that by property (3), $l\le f(\tau)$. Finally using Lemma XI.5.1 we can find two extensions of ρ' on each $S^*(\tau_i)$ $(i=1,2,\dots,l)$ such that all of them pairwise e-split, and it is easy to see that we can make these extensions below their corresponding pseudo-leaf level hence we can extend them to their corresponding pseudo-leaf level. The two pseudo-leaves we pick on $S^*(\tau_i)$ will be the leaves of $S'(\tau_i)$ extending ρ. Of course we can do this for every leaf of $S'(\tau)$ and this finishes the inductive step of our construction.

The tree system (T',S') is then a totally e-splitting tree system and so forces $B\le_T \varphi_e^B$: it suffices to check this for all children $\tau_1,\dots,\tau_s$ of one parent τ. For any two τ_i and τ_j, if they are in different groups, then all their leaves already pairwise e-split by our coding of groups, since $S^*(\sigma)$ is an e-splitting tree. If they are in the same group, then our construction by Lemma XI.5.1 guarantees e-splitting.

Note that in this construction (T',S') is not naturally nested in (T,S) since T' is not necessarily a subtree of T. However, if we view them as set-like trees and make them downward closed, then T' is a subtree of T in this sense. Therefore from this point of view, the sequence of tree systems (T_i,S_i) we build is a nested sequence.

Proof of Lemma XI.5.1

To finish the proof we are left with Lemma XI.5.1. First for simplicity we can assume that each $S^*(\tau_i)$ has height exactly $m+l$, i.e., there are exactly l levels of binary structure above the leaf ρ on each $S^*(\tau_i)$.

Now we prove the Lemma by induction on l. If $l=1$, the claim is trivial, since $S^*(\tau_1)$ is already an e-splitting tree. Suppose that the claim holds for $l=k$ and now we have $k+1$ children. First for each $k+1$-level binary structure above ρ, we hide the last level and view them as k-level structures. We call these nodes k-levels above ρ *key-nodes*. Note that each key-node has two e-splitting extensions at the $k+1$-st level. Next we pick a key-node with the longest φ_e-image, say η on the tree corresponds to τ_1. Now for the rest children $\tau_2,\dots,\tau_{k+1}$ with their k-level structure (remember that we first hide one level), we apply the induction hypotheses and get pairwise e-splitting key-nodes η_{i0},η_{i1} for $i=2,..,k+1$.

Since η has the longest φ_e-image, at least one of η_{20}, η_{21} e-splits with η. Without loss of generality, say η_{20} e-splits with η. Then we reveal the $k+1$-st level and pick two extensions ρ_{20}, ρ_{21} of η_{20}. We can do this for every $i = 2, 3, .., k+1$ and get ρ_{i0}, ρ_{i1} for $i = 2, 3, \ldots, k+1$. Finally we pick two extensions of η at the $k+1$-st level and let them be ρ_{10} and ρ_{11}. It is easy to see that these ρ_{ij}'s pairwise e-split.

This finishes the proof of Lemma XI.5.1 and the whole construction.

XI.6 An ANR Tree Traceable Degree

Since r.e. traceability guarantees that the degree has strong minimal covers, one might ask whether we could remove the property of array recursiveness in Theorem XI.3.4, or equivalently whether tree traceability guarantees array recursiveness. In this section, we show that this is not true (Proposition XI.3.3).

We construct an infinite string α of **ANR** degree by approximation through partial recursive trees $T : \omega^{<\omega} \to \omega^{<\omega}$. In order to code in some information about m_K at any stage of the construction, we need all trees in the construction to have the following property:

$$(*) : \ T(\sigma * i) \downarrow \text{ implies } T(\sigma * i) \supset T(\sigma) * i.$$

Our first tree T_0 will be the identity function on a domain specified as follows. We take the standard limit computation $\lambda(n, s)$ of $m_K(n)$, the modulus function of K. For the root, we set $T_0(\emptyset) = \emptyset$. Suppose we have defined $T_0(\sigma) = \sigma$ and $|\sigma| = n$. We always have $T(\sigma * 0) \downarrow = \sigma * 0$. Then we run through the column $\langle \lambda(n, s) \rangle_{s \in \omega}$ looking for the i-th change. If the i-th change is found, then let $T_0(\sigma * i) = \sigma * i$; otherwise it diverges. It is easy to see that all paths on T_0 are bounded by the function $f(n) = n$, and if we can code in the correct number of changes through the columns of $\lambda(n, s)$ infinitely often, then the final path will compute a function which is not dominated by m_K and hence is **ANR**: simply take $g(n) \leq \alpha$ be the value after the $\alpha(n)$-th change in the corresponding column of the limit computation.

Given a tree T at the current stage of the construction, by property $(*)$, we can pick the true number of changes through the column $\langle \lambda(|T(\emptyset)|, s) \rangle_{s \in \omega}$, say i, and take the full subtree of T above $T(i)$ as the next tree. This ensures that in the end our α is **ANR**.

In addition, we need to guarantee tree traceability. Note that it is equivalent to use an infinite string α instead of a set A in the definition. For any Turing functional Φ, we simply force the totality of Φ as follows. Given T, we first ask whether we can force nontotality:

$$\exists \sigma \textit{ on } T \exists x \forall \tau \supset \sigma, \tau \textit{ on } T (\Phi(\tau, x) \uparrow).$$

If so, we can take the full subtree of T above σ. It is easy to see that $\Phi(\alpha)$ is forced to be partial. If not, then we know:

$$\forall \sigma \textit{ on } T \forall x \exists \tau \supset \sigma, \tau \textit{ on } T (\Phi(\tau, x) \downarrow).$$

Now we can define a subtree T' of T which forces totality. Let $T'(\emptyset) = T(\emptyset)$. Suppose we have defined $T'(\sigma) = T(\tau) = \rho$, then for each i, we search above

$T(\tau * i)$ on T for η with a new convergence of Φ, i.e., a longer Φ-image. If found, we set $T'(\sigma * i)$ to be that node η (this η must be on T and so it is not necessarily Φ-proper). Note that if $T(\tau * i)$ originally diverges, then this search also diverges.

The definition of tree traceability requires two Φ-proper nodes between two levels of T'. So we need to slightly modify this forcing totality construction that at each step, we try to force totality "twice", i.e., above $T(\tau * i)$, we search for two nodes $\eta_0 \subsetneq \eta_1$, each with a longer Φ-image, and define $T'(\sigma * i)$ to be η_1. Finally note that T' preserves property $(*)$ and it is a tracing tree for Φ with the tracing function $f(\sigma) = |\sigma| + 1$.

XI.7 Minimal Degree Construction I

In this section we show that the notion of tree traceability properly extends r.e. traceability. In fact, we can construct a hyperimmune minimal degree which is tree traceable but not r.e. traceable. The idea to force hyperimmunity comes from [Cai10] and the idea to force non-r.e. traceability is from [Gab04].

Our trees are divided into even levels and odd levels. Even levels have some guessing process, which handles hyperimmunity, and odd levels have multiple branchings, which handles non-r.e. traceability. More formally, we use trees T such that for every σ in the domain of T, if $|\sigma|$ is even, then $T(\sigma * 0)$ always have children, and $T(\sigma * 1)$ has children if and only if $\varphi_n(n)$ converges, where $|\sigma| = 2n$; if $|\sigma|$ is odd and $T(\sigma)$ is not a leaf, then it has $|\sigma|$ many children and $T(\sigma * i) \supset T(\sigma) * i$ for $i = 0, 1, \ldots, |\sigma| - 1$.

We list the requirements that we need to satisfy for the infinite string α:

1. R_e: φ_e^α is nontotal, or recursive, or $\alpha \leq_T \varphi_e^\alpha$.
2. P_e: if φ_e is total, then there is an x such that $\beta(x) > \varphi_e(x)$ (for some fixed $\beta \leq_T \alpha$).
3. Q_e: for every recursive function $f = \varphi_e$, there is a function $g \leq_T \alpha$ such that for any potential trace $W_{h(n)}$, either $|W_{h(n)}| > f(n)$, or $g(n) \notin W_{h(n)}$ for some n.

So R_e handles minimality, P_e handles hyperimmunity and Q_e handles non-r.e. traceability. R_e is satisfied by a more or less standard splitting tree construction. We will give intuitions to satisfy P_e and Q_e after we introduce the first tree T_0.

XI.7.1 Initial tree T_0

Our initial tree $T_0 : \omega^{<\omega} \to \omega^{<\omega}$ is defined as follows: First we let $T_0(\emptyset) = \emptyset$. After we define $T_0(\sigma) = \sigma$ for $|\sigma| = 2n$, then we let $T_0(\sigma * i) = \sigma * i$ for $i = 0, 1$. On the 0 side, we let $T_0(\sigma * 0 * j) = \sigma * 0 * j$ for $j = 0, 1, \ldots, n+1$; on the 1 side, we check whether $\varphi_n(n)$ converges: if not, then $T_0(\sigma * 1)$ is a leaf; if so, we let $T_0(\sigma * 1 * j) = \sigma * 1 * j$ for $j = 0, 1, \ldots, n+1$. We call such a 2-level structure a *block*.

Typically, the 1 side of one block guesses whether a partial recursive function at a certain input converges or not, and we will handle our construction such that at each step, we are still doing a similar guessing for all partial recursive functions.

Finally β is computed from α in the following way: for each n, we simply check whether $\alpha(2n) = 1$, if not, then let $\beta(n) = 0$; if so, we compute $\varphi_n(n)$ (and our initial tree construction guarantees that this converges) and let the value plus 1 be $\beta(n)$. So we can infinitely often know a number computed by a total recursive function and β will exceed this value at the corresponding entry.

To satisfy Q_e, we first check whether $f = \varphi_e$ is total or not. If it is partial, then we don't have to do anything; if it is total, then we need to make our tree enough "branchy". Then we can define a function g recursive in α, picking the even-nodes we meet as we go along the tree along path α (see detailed construction in §XI.7.4). The tree is enough branchy so that for any enumeration $W_{h(n)}$, either it enumerates more than $f(n)$ many nodes, or there is one branch which is not in $W_{h(n)}$.

XI.7.2 Blocks

We provide some notions for later use about blocks. We usually think of a tree consisting of blocks. We use letter B to denote blocks. Each block has the following nodes: $T(\sigma), T(\sigma*0), T(\sigma*1), T(\sigma*0*0), T(\sigma*0*1), \ldots, T(\sigma*0*i)$ and, if $T(\sigma*1)$ is not a leaf, also $T(\sigma*1*0), T(\sigma*1*1), \ldots, T(\sigma*1*j)$. We will call $T(\sigma*0)$ at the 0-*node* of the block, etc. We call $\min\{i+1, j+1\}$ the *branching number* of B. All the third level nodes are called *top-nodes* or *branching-nodes* of the block. The block B is *associated* with index e if its 1-node is guessing whether $\varphi_e(x)$ converges for some x. Note that each block is associated with infinitely many indices, by the Padding Lemma.

XI.7.3 Force hyperimmunity

We need an aiding recursive function $l(n)$ for each tree T in our construction such that the $l(n)$-th block from the root (along every path) is guessing at $\varphi_n(x)$ for some x. So for the initial tree T_0, $l(n) = n$ for each n. A lot of our trees do have this identity function as aiding function.

Now we are given a tree T and we want to force P_e. We simply take $l(e)$-th block along any path and the 1 side $T(\sigma*1)$ there is guessing at $\varphi_e(x)$ for some x. If it doesn't converge, then we don't have to do anything to T. If it converges, we then simply take the full subtree above $T(\sigma*1*0)$, and we use the Padding Lemma to get a new aiding function for the full subtree we get.

XI.7.4 Force non-r.e. traceability

To handle Q_e, as we argued above, we can assume that $f = \varphi_e$ is total. For the given tree T, we first need to make it "branchy" in the following way. First compute $f(0)$ and we start with a block guessing at $\varphi_0(x_0)$ (for some x_0) and the branching number is greater than $f(0)$; then we compute $f(1)$, and extend the first block to a block guessing at $\varphi_1(x_1)$ and the branching number is greater than $f(1)$. Similarly we get a new subtree T' of T such that the identity function is the aiding function for T', and every n-th block from the root along every path has branching number greater than $f(n)$, i.e., its number of branches is at least $f(n)+1$.

Define $g(n) \leq_T \alpha$ as follows: to compute $g(n)$ go along α on T' to the n-th block, and take the initial segment of α which is a branching node of that block (and of course we need some canonical coding from strings to natural numbers). Then we can make this $g(n)$ not traced by any $W_{h(n)}$ at subsequent stages.

Later in the construction, we try to preserve the branching of a block (see details in the next subsection) in sense that in any tree T^* we see later, the branching nodes on the first block will extend branching nodes on T' for some block. Suppose this block is the k-th one from the root. Then we look at $W_{h(k)}$, if it has more than $f(k)$ many elements, then we don't have to take care of this h; if the number of elements is bounded by $f(k)$, then at least one of the branching nodes on the k-th block of T' is not in the set, and so we simply take the corresponding branching node on the first block of T^* and we have forced $g(k) \notin W_{h(k)}$. We can do this infinitely often for every h and force the function $g(n)$ to be not traced by any $W_{h(n)}$ with the appropriate property.

XI.7.5 Force minimality

Now we need to handle the classical minimal requirements R_e for any given tree T. The only difficulty is to make sure that the above two types of constructions still work in the subtrees. That is to say, for the new subtree we need a new aiding function, and we want to keep the multiple branching nodes. Fortunately, these are easily handled by the following construction.

Following [Cai12], we first ask whether there is an even-node τ which forces non-e-splitting, i.e., no pair of even-nodes above τ is an e-splitting pair. If so, we can take the full subtree above τ and use the Padding Lemma again to get an aiding function. Branching is of course preserved.

Now assume that this is not the case, i.e., above every even-node, we can find two even-nodes which e-split. Then we need to construct an e-splitting subtree T' of T. First we let $T'(\emptyset) = T(\emptyset)$. By induction, suppose we have defined $T'(\sigma) = T(\tau)$ and $|\sigma| = 2n$. We search above $T(\tau)$ for a pair of e-splitting even-nodes ρ_0, ρ_1 on T, then extend them to blocks guessing at $\varphi_n(x)$ for some x. Let η_0 be extending ρ_0 at the 0-position of the block B_0 and let η_1 be extending ρ_1 at the 1-position of the block B_1. On the 0-side, suppose we have t branching nodes in B_0 extending η_0. We have to keep the branching nodes of B_0 by finding t pairwise e-splitting nodes above η_0, each extending one of the branching nodes extending η_0 (by Lemma XI.5.1). On the 1-side, we do the same thing if $\varphi_n(x)$ converges; we don't do anything if it doesn't. This finishes the construction.

XI.7.6 Verification

We only need to verify that this minimal degree $\mathbf{a} = \mathbf{deg}(\alpha)$ is tree traceable. By the remark after Definition XI.3.1, we can prove this by using α in place of A in the definition. The tracing function $f(\sigma)$ is $f(\sigma) = |\sigma|$. For any Turing functional Φ with $\Phi(\alpha) = \alpha$, we need to find a tracing tree for it. We first define a new Turing functional $\tilde{\Phi}$ as follows: Start from the root $\tilde{\Phi}(i) = \emptyset$ for every i. If we have defined $\tilde{\Phi}(\sigma) = \sigma'$ where $\sigma' \subsetneq \sigma$ and σ is Φ-proper, then we search above σ for the next Φ-proper extension τ and define $\tilde{\Phi}(\tau) = \sigma$ for such τ. Note that still we have $\tilde{\Phi}(\alpha) = \alpha$. Suppose this $\tilde{\Phi}$ is represented by $\varphi_e^{(\cdot)}$ in the requirement R_e.

In the construction, we have made α on an e-splitting partial recursive tree T, and we show that this T is a tracing tree for Φ. It is easy to see that on this tree T, the number of successors of each node is bounded by $f(\sigma)$. To see that there are at least two Φ-proper nodes between two successive nodes, we assume that τ_0 is an immediate successor of σ on T and for a contradiction we also assume that there are only one Φ-proper node in the interval $[\sigma, \tau_0]$ (There must be one since $\tilde{\Phi}(\tau_0)$ properly extends $\tilde{\Phi}(\sigma)$). By our construction of $\tilde{\Phi}$, $\tilde{\Phi}(\tau_0) \subset \sigma$, but then τ cannot $\tilde{\Phi}$-split (e-split) with other successors τ_i of σ. So there must be at least two Φ-proper nodes in the interval $[\sigma, \tau_0]$.

XI.8 Minimal Degree Construction II

In this section, we assume that the reader has some basic idea of minimal degree constructions as in the previous several sections and so we sketch the construction instead of writing out full proofs. The key point here is how to force the degree to be not tree traceable. Forcing minimality is slightly more complicated but very similar, and so omitted.

In order to force a minimal degree to be not tree traceable, we need a large number of immediate successors so that for any possible tracing tree we can always find one of the immediate successors which are not on the tree. The only trick is that a possible tracing function $f(\sigma)$ could be partial somewhere and we will force it to be total on the nodes we are working with.

Since we don't have to force hyperimmunity, we work with trees $T : \omega^{<\omega} \to 2^{<\omega}$ with a recursive domain. Our initial tree T_0 is the identity function with domain $2^{<\omega}$.

We call a tree T *f-branching* if for every node τ on T, it has more than $f(\tau)$ immediate successors on T. For example, T_0 is a 1-branching tree. To force that A is not tree traceable by a certain function f, we will make our trees f-branching and define a functional Φ such that we can avoid any tracing tree in the following steps of construction.

In a similar fashion as the construction in Section XI.7 above, after we construct a tree T, we want all of our following trees T' to *respect* the branching of T such that for every $T'(\sigma') = \tau = T(\sigma)$, each immediate successor ρ of τ on T must have at least one extension ρ' which is an extension of τ on T'.

XI.8.1 Force non-tree-traceability

Suppose we are given a recursive tree T and a path-wise nondecreasing partial recursive function $f(\sigma)$, to force non-tree-traceability we want to do the following for each $B \equiv_T A$ (let $B = \varphi_j^A$): find a subtree T' of T which respects the branching of T, and find a Turing functional Φ such that $\Phi(\varphi_j^C) = \varphi_j^C$ for every path $C \in [T']$ and such that there is no tracing tree for B via tracing function f.

For simplicity we first describe the construction for the case when $B = A$, and briefly sketch for the general case.

Now first let $B = A$. The idea is to make the tree f-branching so that at later stages of the construction we can pick a branch within sufficiently many branches and the branch we pick is not on the potential tracing tree considered at that stage.

A problem is that, f is partial recursive and so it might diverge on some input σ. We cannot wait forever since we are going to build a perfect tree without leaves. The solution is as follows: we first ask the question

$$\exists \tau \text{ on T } \forall \chi \supset \tau (f(\chi) \uparrow).$$

That is, we ask whether there is a node τ such that f diverges everywhere above τ (χ might not be on the tree T). If so we can surely take the full subtree of T above τ as our tree at the next stage, and the non-tree-traceable requirement for this f is satisfied.

If not, then for every node τ on the tree T, we can predict the value of $f(\tau)$ by looking for convergence of $f(\chi)$ for $\chi \supset \tau$. By the negative answer to the question above, we are guaranteed to find such a convergence, and the value of $f(\chi)$ is at least the value of $f(\tau)$, assuming the latter converges.

Then at τ, we can define the new tree T' as follows: We find a number n such that the number of nodes n-levels above τ on T exceeds $f(\chi)$ we find as above. Then we define T' by making all these n-level nodes as immediate successors of τ on T'. This process definitely makes T' respect f.

We define a functional Φ as follows: going along the paths of T', at each node τ we let $\Phi(\tau) = \tau$ and only these nodes are Φ-proper.

At any subsequent stage, we have the current tree T^* in the construction and we are given a potential tracing tree Tr. Since in our construction, we always make new trees respect the branching of old trees, it is not difficult to see that the root η of Tr has to be an initial segment of the root η^* of T^*. By property (2) each immediate successor of η on Tr has to be above one immediate successor of η^* on T', and by property (3), the number of all immediate successors of η is bounded by $f(\eta)$, which is bounded by $f(\chi)$ for the χ we found for η^* as in the construction of T'. So there is at least one immediate successor ξ of η^* which is incomparable with any node on Tr except the root η. In addition, we make T^* respect the branching of T', so there is at least one immediate successor ξ^* of η^* which is above ξ, and we can take the full subtree of T^* above this ξ^*. This construction surely prevents Tr to be a tracing tree, and it is easy to see that we can do this for all potential tracing trees for A at subsequent stages.

Now for the general case, note that we will have splitting tree constructions when we force minimality, so for each $B = \varphi_j^A$ in the same degree, we will have $A = \varphi_k^B$ and such φ_j and φ_k can be viewed as functionals only defined on a certain splitting tree T and in particular they are bijections if we restrict the domain to the nodes on such T. These two provide us translations between two worlds (just like the case for binary strings and general strings). We will write $\varphi_i(\tau)$ instead of φ_i^τ.

First we ask a similar question with $\chi \supset \tau$ replaced by $\chi \supset \varphi_i(\tau)$. If the answer is yes, then also take a full subtree above that node; if the answer is no, then we construct a similar T' which respects $f \circ \varphi_i$. Then we define a functional Φ such that $\Phi(\varphi_i(\tau)) = \tau$ for every τ on T. Now for any potential tracing tree Tr of Φ (with respect to f), we can argue that we can find at least one immediate successor ξ of the root of T', any subtree of T which preserves the branching, such that the full subtree above ξ avoids Tr in the same way. The details of the proof is not difficult and left to the reader.

XI.8.2 Force minimality

In the splitting tree construction, we need to preserve the branching of the previous tree and again Lemma XI.5.1 is used to find splittings above multiple nodes. The details of the construction and the final verifications are easy and also left to the reader.

XI.9 Remarks and Questions

The proof of Proposition XI.3.6 actually shows that, in order to find a minimal degree without strong minimal covers, one must avoid certain types of tree constructions where one can predict the number of children of any given node. It might be interesting if there is any progress in either of the following: extending this argument to further restrain the type of tree construction we use, or providing a tree construction of a minimal degree without strong minimal covers using trees that are multiple-branching as in Section XI.8.

It is still unknown whether the property of having strong minimal covers is downward closed in the Turing degrees. We also want to know whether this is true for tree traceability, and typically a negative answer could be very interesting.

Question XI.9.1. *Are the tree traceable degrees downward closed in the Turing degrees?*

It is proved that in the definition of r.e. traceability, we can use any eventually increasing recursive function in the place of the recursive bound f (see [Nie08, Section 8.2]). It might be interesting to know whether similar phenomenon holds for tree traceability.

Question XI.9.2. *In Definition XI.3.1, can we replace f by a fixed function and get the same class of degrees?*

Ishmukhametov showed that an r.e. degree is r.e. traceable if and only if it is array recursive, and hence solved the strong minimal cover problem for r.e. degrees. We might expect to have such nice characterizations for a broader class of degrees in terms of tree traceability or some other tracing property.

CHAPTER XII
DEGREES OF RELATIVE PROVABILITY

There are many classical connections between the proof-theoretic strength of systems of arithmetic and the provable totality of recursive functions. In this chapter we directly study the provability strength of the totality of recursive functions by introducing a natural ordering on all recursive algorithms and investigating the degree structure induced by it. We prove several results about this proof-theoretic degree structure using recursion-theoretic techniques such as diagonalization and the Recursion Theorem.

XII.1 Historical Background

It is well known that it is sometimes difficult to prove the totality of a recursive function, especially if it is fast-growing. For example, Goodstein's Theorem, which is a Π^0_2 sentence in arithmetic, can be naturally interpreted as the totality of a recursive function; and it is known (see [KiPa82]) that we cannot prove Goodstein's Theorem in Peano Arithmetic (**PA**). Another similar example is the Modified Ramsey Theorem of Paris and Harrington (see [PH77]).

Note that the totality of a partial recursive function is a Π^0_2 sentence in arithmetic, and conversely a Π^0_2 sentence in arithmetic can be naturally interpreted as the totality of a partial recursive function. Therefore the study of the totality of recursive functions can be viewed as the study of the Π^0_2 fragments of theories.

The totality of recursive functions has been studied in terms of subrecursive hierarchies and rates of growth of functions (as in [FW98]). Starting from a countable ordinal α with some structure, one can define fast-growing functions by induction on α and get a hierarchy of such functions. These number-theoretic functions turn out to be a bridge connecting proof-theoretic complexity and computational complexity. One then can prove various theorems relating the axioms systems and the provably total functions in these systems.

For example, it is a classical result that the recursive functions whose totality can be proved in $\mathbf{I\Sigma_1}$ (a fragment of **PA** restricting the induction scheme to only Σ_1 formulae) are exactly the primitive recursive ones, and similarly **PA** corresponds to α-recursive functions for $\alpha < \epsilon_0$ (intuitively, α-recursive functions can be computed by induction below α).

While this approach mainly works for fragments or extensions of Peano Arithmetic, there have also been attempts to look at different theories, for example, in the language of set theory. One can apply an "ordinal analysis" to axiom systems and get an ordinal α as its "norm", then for axiom systems that are good enough (regular), there is similar theorem that the provably total functions in an axiom system T are exactly the functions primitive recursive in some function related to an ordinal below the norm of T, which is closely related to the fast-growing functions (see [Poh98, Theorem 2.1.4.5]). However, such an analysis works mainly for small fragments of **ZFC** such as **KP**.

It is worth mentioning here the study of the degree structure of honest recursive functions (see for example, [Kri98] and [Kri99]). A recursive function is *honest* if its computing time is bounded by a finite iterate of itself. One can induce a degree structure on honest functions by domination and iteration. In particular, there are also generalizations ([KSWxx]) which directly connect the "ϵ_0" version of this degree theory with provability over **PA**. Based on this one can prove, for example, a minimal pair theorem which states that there are two honest functions whose totality form a minimal pair over **PA**, i.e., any honest function whose totality can be proved from the totality of each of these two functions is provably total in **PA** ([KSWxx]).

In reverse mathematics (see [Sh10] for a survey), we have a weak base theory ($\mathbf{RCA}_0$) in second order arithmetic and the provability strength of different sentences (mainly Π^1_2 sentences) and theories are discussed over this base theory. Interestingly, some strong Π^1_2 sentences have so-called "finite miniaturizations". It turns out that some of these finite miniaturizations are natural examples of Π^0_2 sentences that are fast-growing and not provable. For example the finite Kruskal's Theorem is not provable in Π^1_1-$\mathbf{CA}_0$ ($\mathbf{RCA}_0$ plus the comprehension scheme for Π^1_1 formulae, see [Fri98] and [Sim85]).

In set theory, by studying rank-to-rank embeddings, one can induce a purely combinatorial structure called Laver Tables. There is also a recursive function associated with Laver Tables and the interesting fact is that the only known proof of its totality requires a very strong large cardinal axiom (**I3**, see [Deh10]). The best result so far is that this function is not primitive recursive ([Deh10]). So we want to know whether a large cardinal axiom is really necessary to prove its totality, or whether **ZFC** suffices.

In summary, there are some quite different approaches to the provably total functions over various theories in different languages, and we might want to know whether there are common features among them (using fast growing hierarchy, [Loe92] gave unified proofs of unprovability over **PA** for Goodstein's Theorem, Modified Ramsey Theorem and the finite Kruskal's Theorem). In addition, old approaches seem to have difficulty classifying the provably total functions in stronger theories such as **ZFC** or **ZFC** plus some large cardinal axiom. So we would like to develop a basic framework to study the relative provability strength of the totality of recursive functions in a wide variety of theories.

There are two notions of proofs: one is the social version, which is written in natural language and can be used in communication among mathematicians; the other is the formal version, which is defined as a sequence of formal sentences which satisfy certain properties and which is usually not used in communication (see more discussions in [Bus98]). In recursion theory, we have a similar phenomenon for computations: there are social and formal versions and we mainly use the social version in communication and in the proofs we write. In fact, recursion theory has taken great advantage of using the social or informal version of computations so that we can write out much more complicated constructions such as injury arguments and tree constructions, which are almost impossible to write out in formal Turing machine language. In this chapter, we will use social proofs when we argue for proofs in formal systems, but we shall keep in mind that, when we argue formally over some theory T, the natural numbers we use may be nonstandard, and the argument has to work with nonstandard numbers. If we do not specify that we are working in a formal system, then we always assume that we are arguing in the standard model of the natural numbers.

XII.2 New Approach: Base Theory

Let $\varphi_i(x)$ be a total recursive function with a fixed algorithm indexed by i. By the *totality* of φ_i we mean the sentence stating that φ_i is total, i.e., $\forall x \exists s \varphi_{i,s}(x)\downarrow$ (here we can use Kleene's T predicate, which is primitive recursive, to express $\varphi_{i,s}(x)\downarrow$, i.e., the computation with an index e and an input x converges within s steps). For convenience we use $tot(\varphi)$ to denote this sentence.

In reverse mathematics, we choose a weak base theory ($\mathbf{RCA}_0$) which is powerful enough to develop the basics of mathematics: pairs, strings, formulae, sentences, proofs, etc., then we can discuss the relative provability of sentences and theories over it.

Similarly, here we also want to fix a base theory T. However we are not specific about which theory T we use here: it just needs to be strong enough in a sense which we will explain, and in particular, it could be very strong and does not have to be in a fixed language. All of our discussions and all the theorems will work for any possible theory satisfying the requirements we list below.

First, we require T to be axiomatizable, i.e., it has a recursive list of axioms. This is to say, given a proof (coded as some number s), we can effectively tell whether it is a valid proof in T. In the theorems we are going to prove, we will construct recursive functions by diagonalization, e.g., diagonalize against all possible functions whose totality is provable in the theory T, therefore it is crucial that we can recursively decide whether a proof is a proof in T. It is convenient here to assume that the coding of the recursive list of T-axioms is a provably total function in T (for example, we do not need to run a Goodstein sequence computation to decide whether a sentence is an axiom in T), but this is not crucial, since we can modify the definition of a proof and require that each axiom has an affiliated computation which confirms that it is an axiom.

We require that T is consistent with true arithmetic ($\mathbf{TA}$) in the following sense: In T one has a fixed interpretation of arithmetic, i.e., one can define the domain of natural numbers, zero and the successor operation, together with plus, times and the order of natural numbers. Then each sentence ψ in arithmetic has a translation $\ulcorner\psi\urcorner$ in the language of T and we require that if $T\vdash\ulcorner\psi\urcorner$ then $\mathbf{TA}\vdash\psi$. So if we can prove in T that some partial recursive function φ_e is total (i.e., $T\vdash\ulcorner tot(\varphi_e)\urcorner$), then it is total. (For convenience, we will omit $\ulcorner\urcorner$ in the notation and simply write $T\vdash tot(\varphi_e)$.)

In addition, like $\mathbf{RCA}_0$, with the interpretation of arithmetic as above, T is powerful enough to develop the basic notions of pairs, strings, sentences and proofs, therefore we can write out formal sentences such as "$T\vdash\varphi$" for some sentence φ in the language of T. It can also express the notion of Turing machines and the computation sequence of Turing machines. In particular, we can talk about φ_e, the e-th partial recursive function.

Finally we want to impose a convention on our partial recursive functions. We automatically convert them into partial recursive functions satisfying the convention. We need $\mathbf{I\Sigma_1}$ to show that we can do such a conversion and the new function is "equivalent" to the old one as far as we are concerned. We will discuss the details in the next section.

Therefore $\mathbf{I\Sigma_1}$ is a good candidate for our base theory T and any stronger theory will also work. Note that some of the theorems we will prove may not require the convention we impose and so they may work in even weaker base theories. It might also be interesting to discuss our degree structure when the base theory is weaker than $\mathbf{I\Sigma_1}$, but this is beyond our scope here.

As opposed to the "honesty restraint" on recursive functions in the analysis of degrees of honest functions, we don't have any special restraint on recursive functions. Another major difference is that, the honest function constructions, as well as the old approaches using ordinals we discussed in the previous section, are mainly based on iteration of functions and fast-growing hierarchies, while ours are mostly based on diagonalizations [1]. For example, as we mentioned earlier, in [KSWxx] they have a theorem on minimal pairs similar to Theorem XII.6.1, but it only works for honest recursive functions and only for **PA** as a base theory, while our minimal-pair theorem works for all recursive functions and for both weaker and stronger theories without modification. It might be interesting to decide if there are natural connections between these two degree structures.

We thank Richard A. Shore for helpful comments and suggestions, especially for the discussions in reverse mathematics, and also thanks Justin Moore for introducing the topic on Laver Tables to him, which initially motivated the research in this chapter.

XII.3 Basics

We start with the definition of the order:

Definition XII.3.1. Given φ_i and φ_j, we say φ_i is *provably reducible* to φ_j if $T + tot(\varphi_j) \vdash tot(\varphi_i)$. We denote this by $\varphi_i \leq_p \varphi_j$.

It is easy to see that $\leq_p$ is reflexive and transitive, so it naturally induces a degree structure: φ_i and φ_j are *provably equivalent* ($\varphi_i \equiv_p \varphi_j$) if they are provably reducible to each other. Given φ_i, we use $[\varphi_i]$ to denote the collection of φ_j's that are provably equivalent to φ_i, i.e., the equivalence class of φ_i. We will call such $[\varphi_i]$ a *provability degree* and we use $\mathcal{P}$ to denote the class of all provability degrees, ordered by the induced provability reducibility.

It is very important to note here that, strictly speaking, this order is defined on recursive *algorithms* instead of recursive *functions*. As we will discuss in Section XII.7, each function f has many *representations* φ_i, each is the same function as f, but these representations might have different provability degree. In this chapter, we will use Greek letters φ, ψ, θ for (possibly partial) recursive functions with fixed algorithms, and f, g, h for recursive functions as functions. So when we write φ, it is assumed that it corresponds to a fixed algorithm φ_i, or we are constructing such a φ and we refer to the algorithm we are defining.

We use 0 to denote the constant zero function computed in the easiest way, i.e., output 0 regardless of input, so $[0]$ is the bottom degree in $\mathcal{P}$. It is also the collection of all functions whose totality is provable in T.

To simplify some arguments, we will impose the following convention, which is commonly used in recursion theory: For a recursive function φ, if $\varphi(x)$ converges in s steps, then all $\varphi(y)$ for $y < x$ converge in less than s steps. More precisely, given any recursive function φ, we first convert φ to another function $\tilde{\varphi}$: for each x, we wait for all $\tilde{\varphi}(y)$ for $y < x$ to converge before we start computing $\varphi(x)$, then output the value $\varphi(x)$ as $\tilde{\varphi}(x)$. We need to show that $[\varphi] = [\tilde{\varphi}]$ (and so we can always assume that every function satisfies this convention). One direction $\varphi \leq_p \tilde{\varphi}$

[1] Such diagonalization ideas have been used in Fischer's work ([Fis65]) to construct recursive functions with various kinds of provable or nonprovable properties (such as provably one-to-one, provably onto, or provably increasing).

is trivial. For the other direction $\tilde{\varphi} \leq_p \varphi$, we assume that $\tilde{\varphi}$ is not total. Then we can find the least x that $\tilde{\varphi}(x)$ does not converge (using $\mathbf{I\Sigma_1}$), and so get a contradiction with the construction of $\tilde{\varphi}$ and the fact that $\varphi(x)$ converges. One important fact based on our convention is that if φ converges at infinitely many inputs then φ is total.

Next we show that in $\mathcal{P}$ one can define a join and a meet operator, and in fact $\mathcal{P}$ is a distributive lattice.

Definition XII.3.2. Given two recursive functions φ and ψ, the join of φ and ψ, denoted as $\varphi \boxplus \psi$, is defined as the following recursive function: for each input n, we compute $\varphi(n)$ and $\psi(n)$ simultaneously, and output $(\varphi \boxplus \psi)(n) = 0$ only if both converge. The meet of φ and ψ, denoted as $\varphi \boxtimes \psi$, is defined in a very similar way, except that we output $(\varphi \boxtimes \psi)(n) = 0$ if either $\varphi(n)$ or $\psi(n)$ converges.

Note that by these two definitions we really mean that we find two recursive functions $k_\boxplus$ and $k_\boxtimes$ (which are provably total by the *s-m-n* theorem) such that $\varphi_i \boxplus \varphi_j = \varphi_{k_\boxplus(i,j)}$ and $\varphi_i \boxtimes \varphi_j = \varphi_{k_\boxtimes(i,j)}$.

Also note that in the above definitions, one can change the output value quite arbitrarily: for $(\varphi \boxplus \psi)(n)$, we can also use either $\varphi(n)$ or $\psi(n)$ as the output value, since we need to wait for both to converge; for $(\varphi \boxtimes \psi)(n)$, we may use the value of $\varphi(n)$ or $\psi(n)$, whichever converges first.

Recall that in a partial order, $[\varphi] \vee [\psi]$ denotes the join of two degrees and $[\varphi] \wedge [\psi]$ denotes the meet. One can easily show the following:

Proposition XII.3.3. $[\varphi \boxplus \psi] = [\varphi] \vee [\psi]$, *and* $[\varphi \boxtimes \psi] = [\varphi] \wedge [\psi]$.

Proof. For the first claim: $[\varphi] \leq [\varphi \boxplus \psi]$ and $[\psi] \leq [\varphi \boxplus \psi]$ follows directly from the definition. Given $[\theta]$ which is above both $[\varphi]$ and $[\psi]$, $[\theta]$ is also above $[\varphi \boxplus \psi]$ since we can prove the totality of $\varphi \boxplus \psi$ from the totality of both φ and ψ, which we can get from the totality of θ.

For the second: it is easy to see that φ being total guarantees that $\varphi \boxtimes \psi$ is total; and similarly ψ being total also proves that $\varphi \boxtimes \psi$ is total. Now given some θ whose degree is below both $[\varphi]$ and $[\psi]$, we want to show that $\theta \leq_p \varphi \boxtimes \psi$. Note that the totality of $\varphi \boxtimes \psi$ shows that either φ is total or ψ is total by our convention. Then since we have proofs that $\theta \leq_p \varphi$ and $\theta \leq_p \psi$, we know that θ has to be total in either case. $\square$

Thus $\mathcal{P}$ is a lattice with these two operations, and distributivity easily follows from the distributivity of "and" and "or" in Definition XII.3.2.

Here is a lemma which will be used later but it is convenient to make explicit now.

Lemma XII.3.4. *If* $T \vdash \varphi = \psi$, *then* $[\varphi] = [\psi]$.

By $\varphi = \psi$ we mean that, for each n, if $\varphi(n)$ converges, then $\psi(n)$ has to converge to the same value, and vice versa (therefore they are the same as partial functions). The proof is almost obvious.

XII.4 Jump Operator

As in the Turing degrees, we can also define a natural jump operator in the provability degrees.

Definition XII.4.1. Given a recursive function φ, the jump of φ, denoted as φ^*, is defined inductively as follows: at each stage s, we check whether s is (the Gödel number of) a proof witnessing $T + tot(\varphi) \vdash tot(\varphi_e)$ for some recursive function φ_e: If so, we let $\varphi^*(s) = \varphi_e(s) + 1$; If not, we let $\varphi^*(s) = 0$; and then proceed to the next stage $s + 1$.

By the *s-m-n* theorem, this definition also corresponds to a recursive function k_* such that $\varphi_{k_*(i)} = \varphi_i^*$. Before we show that this is a good definition of a jump operator, we will need the following Padding Lemma (and that it is provable in T).

Lemma XII.4.2 (Padding Lemma)**.** *Suppose s is (a code of) a proof which witnesses $\Sigma \vdash \psi$, then there exists an infinite recursive list $s_0, s_1, \ldots, s_n, \ldots$ of proofs witnessing the same result. Moreover, s_i as a function of i is provably total in T.*

The proof is almost obvious since we can add arbitrary redundancy into proofs. This lemma works for proofs in the same way as the Padding Lemma in recursion theory does for algorithms.

Proposition XII.4.3. *$T + tot(\varphi^*) \vdash$ "if $T + tot(\varphi) \vdash tot(\varphi_e)$, then φ_e is total", and in fact for every ψ, $[\psi] \geq [\varphi^*]$ if and only if $T + tot(\psi) \vdash$ "if $T + tot(\varphi) \vdash tot(\varphi_e)$, then φ_e is total".*

Proof. We argue in $T + tot(\varphi^*)$ for the first claim. If we have a proof s witnessing $T + tot(\varphi) \vdash tot(\varphi_e)$, then by the Padding Lemma we have infinitely many proofs $s_0, s_1, \ldots$ witnessing the same result. Therefore in the construction of φ^* at each of these stages s_i we make $\varphi^*(s_i) = \varphi_e(s_i) + 1$. Then it is easy to see that the totality of φ^* guarantees the totality of φ_e (by our convention).

If $T + tot(\psi)$ proves the assertion that every $T + tot(\varphi)$-provably total function is total, then it is easy to see that $T + tot(\psi)$ proves the totality of φ^*, and so $[\psi] \geq [\varphi^*]$. ☐

This proposition directly shows that $[\varphi] \leq [\varphi^*]$ since $T + tot(\varphi) \vdash tot(\varphi)$. It is direct from the diagonalization that $[\varphi^*] \not\leq [\varphi]$, therefore $[\varphi] < [\varphi^*]$, i.e., $[\varphi^*]$ is strictly above $[\varphi]$.

Corollary XII.4.4. *There are no maximal degrees in $\mathcal{P}$.*

More importantly, the jump operator preserves $\leq_p$:

Proposition XII.4.5. *If $\varphi \leq_p \psi$, then $\varphi^* \leq_p \psi^*$.*

Proof. We prove this by contradiction (in $T + tot(\psi^*)$): Assume that φ^* is not total, then there exists a least s where $\varphi^*(s)$ diverges (by $\mathbf{I\Sigma_1}$), so s is a proof witnessing $T+tot(\varphi) \vdash tot(\varphi_e)$ and $\varphi_e(s)$ diverges. Since we have a proof witnessing $T+tot(\psi) \vdash tot(\varphi)$, from s we can find an s' which is a proof witnessing $T+tot(\psi) \vdash tot(\varphi_e)$. Then by the Padding Lemma we can find an s'' which proves the same result as s' and $s'' > s$. The totality of ψ^* shows that $\varphi_e(s'')$ converges, which contradicts the divergence of $\varphi_e(s)$. ☐

This proposition shows that the jump operator is well-defined on the degrees, and we can write $[\varphi]^*$ for $[\varphi^*]$ since it is independent of the choice of the function φ in $[\varphi]$.

XII.5 Incomparable Degrees

Before we prove our next theorem, we want to mention a technique for constructing recursive functions using the Recursion Theorem. Briefly, we can assume that we know the index of the function we are constructing prior to our construction. This may sound very strange to readers who are not familiar with recursion theory, and so we shall explain it in detail. The Recursion Theorem is stated below, and a proof can be found in [Rog87] or any other standard textbook in classical recursion theory.

Theorem XII.5.1 (Recursion Theorem)**.** *For every recursive function f, there is an index e such that $\varphi_e = \varphi_{f(e)}$.*

We will use the Recursion Theorem in the following way: We first give an explicit construction of a partial recursive function with a parameter i, then by the *s-m-n* theorem, there is a recursive function h such that $\varphi_{h(i)}$ is the function constructed from parameter i; we apply the Recursion Theorem to h and get an index e such that $\varphi_e = \varphi_{h(e)}$, and let this function be ψ. That is, when we describe the construction of ψ we can assume that we already know its index e.

There are two points to make: First, our base theory T is powerful enough to prove the Recursion Theorem, and by the proof of the Recursion Theorem, T actually proves $\varphi_e = \varphi_{h(e)}$ for one index e assuming that $h(e)$ converges. Although in this chapter we only need h's that are provably total in T, this argument actually works for any total h regardless of its provability degree. By Lemma XII.3.4, $[\varphi_e] = [\varphi_{h(e)}]$ and so we can identify φ_e with $\varphi_{h(e)} = \psi$ in our argument.

Second, in the Recursion Theorem it is possible that the φ_e we get is partial, and so we need to make sure that, in the construction of $\varphi_{h(i)}$, regardless of the parameter i we use in the construction, $\varphi_{h(i)}$ is always total. Then by applying the Recursion Theorem, we always get a total function ψ.

Here is the theorem we want to prove. It directly implies that there are incomparable degrees in $\mathcal{P}$.

Theorem XII.5.2. *For every $[\varphi] \neq [0]$, there is a $[\psi] \leq [\varphi]^*$ which is incomparable with $[\varphi]$, i.e., $[\varphi] \not\leq [\psi]$ and $[\psi] \not\leq [\varphi]$.*

Proof. Our construction of ψ is divided into even and odd stages. At even stages we try to satisfy $[\psi] \not\leq [\varphi]$. This is handled in the same way as the construction of the jump: at stage $2s$ we check whether s is a proof witnessing $T + tot(\varphi) \vdash tot(\varphi_e)$ for some φ_e: if so we compute $\varphi_e(s)$ and let $\psi(s) = \varphi_e(s) + 1$; if not we simply let $\psi(s) = 0$.

At odd stages we try to satisfy $[\varphi] \not\leq [\psi]$, i.e., we cannot prove the totality of φ from the totality of ψ. At stage $2s + 1$, we check whether s is a proof witnessing $T + tot(\psi) \vdash tot(\varphi)$ (by the Recursion Theorem): If not, do nothing; if so we terminate the whole construction and let ψ be the constant zero function afterwards, i.e., output $\psi(t) = 0$ for all $t > s$ (in this case we say that we apply *annihilation* to ψ).

First of all, it is easy to check that, no matter which φ_i we use, or whether annihilation happens or not, $\varphi_{h(i)}$ is always total. So the Recursion Theorem gives us a total ψ.

Next we need to show that $[\varphi] \nleq [\psi]$. If $[\varphi] \leq [\psi]$, then we can pick the least proof s_0 witnessing $T + tot(\psi) \vdash tot(\varphi)$. We can then show that the annihilation happens in the construction and ψ is then eventually constant 0, and in fact we can prove its totality in T: We write down the first such proof s_0 and verify that it is the proof we want; then for each $s < s_0$ we check that s is not a proof witnessing $T + tot(\psi) \vdash tot(\varphi)$ so annihilation does not happen before stage $2s_0 + 1$; also for each $s \leq s_0$ we write down the complete computation sequence of $\varphi_{e_s}(s)$ at even stages such that s is a proof witnessing $T + tot(\varphi) \vdash tot(\varphi_{e_s})$. Our base theory T is consistent with true arithmetic and φ is total, therefore if we can prove that φ_{e_s} is total then $\varphi_{e_s}(s)$ does converge and we can write out the complete computation sequence to prove that it converges (without proving the totality of φ_{e_s}).

So combining the sentences we write down we can show that at stage $2s_0 + 1$ the annihilation happens, i.e., we get a proof witnessing $T \vdash tot(\psi)$. Together with the proof s_0 witnessing $T + tot(\psi) \vdash tot(\varphi)$, we would get $T \vdash tot(\varphi)$, which contradicts the assumption that $[\varphi] \neq [0]$.

Therefore annihilation never happens and $[\varphi] \nleq [\psi]$. By the diagonalization at even stages we have also satisfied $[\psi] \nleq [\varphi]$. So $[\psi]$ is incomparable with $[\varphi]$.

Finally we check that $[\psi] \leq [\varphi]^*$: Arguing in $T + tot(\varphi^*)$, if annihilation happens, then ψ is total. If not, then ψ is the same function as φ^* by the construction at even stages. Since φ^* is total, ψ is also total in this case. □

XII.6 Minimal Pair

As in the Turing degrees, we can also find minimal pairs in the provability degrees.

Theorem XII.6.1. *There are two nonzero degrees $[\varphi]$ and $[\psi]$ such that $[\varphi] \wedge [\psi] = [0]$.*

Proof. We construct two functions φ and ψ simultaneously by induction. Suppose at stage s we have already defined φ and ψ up to x_s and y_s respectively.

Let $s = 2m$ be an even stage. We check whether m is a proof witnessing $T \vdash tot(\varphi_e)$ for some φ_e. If so, we define φ and ψ as follows:

$$\begin{cases} \varphi(x_s) = \varphi_e(x_s) + 1, & \\ \psi(y) = 0, & \text{for } y \in [y_s, y_s + t) \end{cases}$$

where t is the number of steps needed to compute $\varphi_e(x_s)$. This is to say, we try to compute $\varphi_e(x_s)$ and extend ψ by letting $\psi(y) = 0$ for $y \geq y_s$ until $\varphi_e(x_s)$ converges. So if $\varphi_e(x_s)$ diverges, then φ is partial and ψ is total (and eventually 0). If $\varphi_e(x_s)$ converges at step t, then we have made $\psi(y) = 0$ for $y \in [y_s, y_s + t)$ and we let $\varphi(x_s) = \varphi_e(x_s)+1$ for diagonalization. Then we go to stage $s+1$ with $x_{s+1} = x_s+1$ and $y_{s+1} = y_s + t$.

If m is not a proof witnessing $T \vdash tot(\varphi_e)$ for any e, then we simply let $\varphi(x_s) = \psi(y_s) = 0$ and go to stage $s+1$ with $x_{s+1} = x_s + 1$ and $y_{s+1} = y_s + 1$.

At odd stages, we basically do the same thing, except that we switch the roles of φ and ψ, that is, we try to diagonalize with ψ and extend φ by 0 until the corresponding computation converges.

Since T is consistent with true arithmetic, it is easy to see that φ and ψ are both total and both have nonzero degree by diagonalization. Then we prove in T that $\varphi \boxtimes \psi$ is total: If we ever met a divergent $\varphi_e(x_s)$ or $\psi_e(y_s)$ in the construction then we would make φ or ψ eventually constant 0 and so $\varphi \boxtimes \psi$ is still total; otherwise both φ and ψ are total and $\varphi \boxtimes \psi$ is obviously total. In either case $\varphi \boxtimes \psi$ is total, so $[\varphi] \wedge [\psi] = [0]$. □

It is not difficult to check that both $[\varphi]$ and $[\psi]$ are below $[0]^*$, and in fact $[\varphi] \vee [\psi] = [0]^*$: it is easy to check that they are both below $[0]^*$; conversely $tot(\varphi)$ and $tot(\psi)$ together prove that every function φ_e whose totality is provable in T is total, and so by Proposition XII.4.3 we know that they also prove the totality of 0^*. This shows that $[0]^*$ is the top of a "diamond".

XII.7 Degree Spectrum and Minimal Degrees

A given recursive function f has many *representations* $\varphi_{e_0}, \varphi_{e_1}, \ldots$, i.e., each $\varphi_{e_i} = f$ as functions, and they may have different provability degrees. So it is natural to give the following definition:

Given a recursive function f, we define the *degree spectrum* of f, denoted as (f), to be the collection of provability degrees that contain a function which is the same as f as functions, i.e., $(f) = \{[\varphi_i] : f = \varphi_i\}$.

It is not difficult to show that the degree spectrum is closed upwards and under meet.

Proposition XII.7.1.

1. *If $[\psi] > [\varphi]$, then there is a function θ such that $\theta = \varphi$ as functions and $[\theta] = [\psi]$.*
2. *If $\varphi = \psi$ as functions then there is a function θ such that $\varphi = \psi = \theta$ as functions and $[\theta] = [\varphi \boxtimes \psi]$.*

Proof. In the definition of the join and the meet, we noted that the output values for $\varphi \boxplus \psi$ or $\varphi \boxtimes \psi$ can be quite arbitrary. For the first claim, we can change the output value of $(\varphi \boxplus \psi)(n)$ to be $\varphi(n)$, and this gives a function θ we want. For the second claim, we change the output value of $(\varphi \boxtimes \psi)(n)$ to be either $\varphi(n)$ or $\psi(n)$ whichever converges first, and this also gives a desired function θ. □

For example, (0), the degree spectrum of the constant zero function (as a function), is the collection of all degrees. In contrast, (0^*), the degree spectrum of 0^* (as a function), does not contain the bottom degree $[0]$. One might naturally ask whether (0^*) has a minimum element, or whether there are other degree spectra which are principal ones (i.e., contain a minimum element).

Interestingly, the answer is no by the following theorem, i.e., the only principal degree spectrum is (0).

Theorem XII.7.2. *Given $[\varphi] \neq [0]$, then there is a function ψ such that $\psi = \varphi$ as functions and $[\psi] < [\varphi]$.*

Proof. We again divide the construction of ψ into even and odd stages. At stage $2s$ we let $\psi(s) = \varphi(s)$, i.e., follow the same algorithm and output the same value. At stage $2s+1$ we check whether s is a proof witnessing $T + tot(\psi) \vdash tot(\varphi)$ (by the Recursion Theorem). If not, we do nothing; if so, we let ψ be a constant 0 function afterwards (similarly we call such process the *annihilation* of ψ).

By the same argument as in the proof of Theorem XII.5.2 we can show that $[\varphi] \not\leq [\psi]$ and so $\psi = \varphi$ as functions. It is also easy to argue that $[\psi] \leq [\varphi]$: if annihilation happens then ψ is total; if not then by the totality of φ and our construction at even stages we also know that ψ is total. $\square$

One might have some intuition that the provability degree of a recursive function φ corresponds to the growth rate of the computing time function $\bar{\varphi}$ (i.e., a function which outputs the number of steps in the computation) of φ[2], or that the functions which need more computing time have higher provability degree. However, the above construction shows that such an intuition is not true: our new function ψ needs more steps in the computation than φ, but has strictly lower degree. It is true, in contrast, that if T proves that $\bar{\varphi}$ dominates $\bar{\psi}$, then $[\varphi] \geq [\psi]$.

This theorem also gives strict conditions on minimal degrees (though we do not know whether they exist). Recall that a nonzero degree is *minimal* if there is no nonzero degree strictly below it.

Corollary XII.7.3. *If $[\varphi]$ is minimal, then for any $\theta \in [\varphi]$, θ has a representation which is provably total. In addition, all minimal degrees (if they exist) are below $[0]^*$.*

Proof. The first claim directly follows from the construction in Theorem XII.7.2: since ψ has degree strictly below $[\varphi]$, it must be the case that $[\psi] = [0]$. For the second claim, we follow the same construction and get a ψ from φ. By the same reason, $T \vdash tot(\psi)$, and then we can prove $tot(\varphi)$ from $tot(0^*)$ as follows:

Suppose φ is not total (at s). Since ψ is total, it must be the case that annihilation happens at some stage before $2s$. Therefore we have a proof witnessing $T + tot(\psi) \vdash tot(\varphi)$, and combining it with $T \vdash tot(\psi)$ we get a proof witnessing $T \vdash tot(\varphi)$. By Proposition XII.4.3, we know that φ is total (since we are arguing in $T + tot(0^*)$). This contradicts the assumption that φ is not total, therefore φ is total. $\square$

XII.8 Open Questions

We end with some open questions. Since this is a new subject, some of the open questions might be easy to answer. First, we want to know whether we can "control the jump" as in various theorems in the Turing degrees. In particular we can ask whether a jump inversion theorem holds in $\mathcal{P}$, i.e., whether every $[\varphi] \geq [0]^*$ is the jump of a degree $[\psi] < [\varphi]$. We are also interested in characterizations of

[2] Note that $\bar{\varphi}$ and φ have the same provability degree, since their computations are almost the same, and the only difference is the outputs.

degrees below $[0]^*$ (for example, whether there is an analogous version of the Limit Lemma).

One can also ask questions about various notions from Turing degree theory, for example, the cupping property, the join property, high/low hierarchy, diamond-bounding (see Theorem XII.6.1), etc. Continuing the discussion of the previous section, we want to know whether minimal degrees exist. It might be also interesting to consider different embedding problems, such as partial orders or distributive lattices, especially if we want to study the decidability or even the degree of the theory of $\mathcal{P}$.

In addition to these *degree-theoretic properties*, we are also interested in *combinatorial properties*, i.e., the combinatorial aspects of the functions in each degree, which might be useful in studying specific number-theoretic or combinatorial examples, such as the function associated with Laver Tables. For example, one can define a function f to be *diagonally nonprovable* (*DNP*) if $f(s) \neq \varphi_{e_s}(s)$ for every s which is a proof witnessing $T \vdash tot(\varphi_{e_s})$ for some φ_{e_s}, and say a degree is **DNP** if it contains a DNP function (this is motivated by the definition of **DNR** degrees in recursion theory). It is easy to see that **DNP** degrees are not zero, and one may ask whether there are nonzero non-**DNP** degrees.

Another important class of combinatorial properties is the class of domination properties. A function f *dominates* g if $f(x) \geq g(x)$ for cofinitely many x. For example, say a degree is $[0]$-*dominated* (or *hyperimmune-free*, using classical terminology from recursion theory) if every function in it is dominated by a function in $[0]$. We can also ask whether there are nonzero $[0]$-dominated degrees. (Note that Corollary XII.7.3 shows that minimal degrees, if they exist, are $[0]$-dominated and non-**DNP**.) In particular, the notions of domination properties may be more closely related to the subrecursive hierarchy or the degree theory about the honest functions we mentioned in the introduction.

It would also be interesting to analyze the provability degrees of the functions appearing in the different approaches we mentioned in the first section and in particular find relations between these degrees and $[0]^*$ in each of their base theories. For example, over **PA**, we want to know whether H_{ϵ_0} (the classical example of a function whose totality is not provable in **PA**, see [FW98]) has provability degree $[0]^*$. We also have some natural examples of Π^0_2 sentences (finite miniaturizations) in reverse mathematics and it is also interesting to decide their provability degree. For example, the finite Kruskal's Tree Theorem (see [Fri98]) seems to be much stronger than $[0]^*$ over $\mathbf{I\Sigma_1}$ or even over **PA**, and it would be very interesting to find natural notions in the provability degrees that correspond to these strong Π^0_2 statements.

BIBLIOGRAPHY

[ASDWY09] K. Ambos-Spies, D. Ding, W. Wang and L. Yu, *Bounding Non-GL_2 and R.E.A.*, The Journal of Symbolic Logic, Vol. 74, No. 3, 2009, pages 989–1000.

[Ars81] M. M. Arslanov, *On some generalizations of a fixed point theorem*, Izv. Vyssh. Uchebn. Zaved. Mat., 1981, no. 5, pages 9–16.

[Ars85] M. M. Arslanov, *Lattice properties of the degrees below* $\mathbf{0}'$, Doklady Ak. Nauk SSR, Vol 283, 1985, pages 270–273.

[Ars09] M. M. Arslanov, *Definability and elementary equivalence in the Ershov difference hierarchy*, in Logic Colloquium 2006, Lecture Notes in Logic, Association for Symbolic Logic and Cambridge University Press, New York, 2009, pages 1–17.

[Ars10] M. M. Arslanov, *The Ershov hierarchy*, in Computability in Context: Computation and Logic in the Real World, Cooper, S. B. and Sorbi, A. eds., Imperial College Press, London, 2010.

[AKL10] M. M. Arslanov, I. Sh. Kalimullin and S. Lempp, *On Downey's Conjecture*, Journal of Symbolic Logic, Vol 75, 2010, pages 401–441.

[Bus98] S. Buss, *An Introduction to Proof Theory* , in: Samuel R. Buss, Editor(s), Studies in Logic and the Foundations of Mathematics, Elsevier, 1998, Volumn 137, Handbook of Proof Theory, pages 1–78.

[Cai10] M. Cai, *A hyperimmune minimal degree and an ANR 2-minimal degree*, Notre Dame Journal of Formal Logic, Volume 51, Number 4, 2010, Pages 443–455.

[Cai11] M. Cai, *A* $\overline{\mathbf{GL}_2}$ *2-minimal degree*, Journal of Mathematical Logic, preprint.

[Cai12] M. Cai, *Array nonrecursiveness and relative recursive enumerability*, Journal of Symbolic Logic, preprint.

[CSh12] M. Cai and R. A. Shore, *Domination, forcing, array nonrecursiveness and relative recursive enumerablity*, Journal of Symbolic Logic, preprint.

[CSSxx] M. Cai, R. A. Shore and T. A. Slaman, *The n-r.e. degrees: undecidability and Σ_1 substructures*, preprint.

[Coo71] S. B. Cooper, *Degrees of Unsolvability*, Ph.D. Thesis, 1971, Leicester University.

[Coo73] S. B. Cooper, *Minimal Degrees and the Jump Operator*, The Journal of Symbolic Logic, Volume 38, Issue 2 (1973), pages 249–271.

[CLLSS91] S. B. Cooper, L. Harrington, A. H. Lachlan, A. H., S. Lempp and R. I. Soare, *The d-r.e. degrees are not dense*, Annals of Pure and Applied Logic, Vol 55, 1991, pages 125–151.

[Deh10] P. Dehornoy, *Elementray embeddings and algebra*, in: M. Foreman, A. Kanamori, Editor(s), Handbook of Set Theory, Springer, 2010, pages 737–774.

[Dow89] R. Downey, *D-r.e. degrees and the non-diamond theorem*, Bulletin of London Mathematical Society, Vol 21, 1989, pages 43–50.

[DGLMxx] R. Downey, N. Greenberg, A.E.M. Lewis, A. Montalbán, *Extensions of uppersemilattice embeddings below computably enumerable degrees*, preprint.

[DJS90] R. Downey, C. Jockusch and M. Stob [1990], *Array nonrecursive sets and multiple permitting arguments*, in Recursion Theory Week, K. Ambos-Spies, G. H. Müller and G. E. Sacks eds., Springer-Verlag, Berlin, 1990, pages 141–174.

[DJS96] R. Downey, C. Jockusch and M. Stob, *Array nonrecursive degrees and genericity*, London Mathematical Society Lecture Notes Series 224, 1996, University Press, Pages 93–105.

[ELxx] P. Ellison, A. E. M. Lewis, *A characterization of the high c.e. degrees*, preprint.

[Eps79] R. L. Epstein, *Degrees of Unsolvability: Structure and Theory*, Lecture Notes in Mathematics 759, Springer-Verlag, Berlin, 1979.

[Epsxx] R. Epstein, *Invariance and Automorphisms of the Computably Enumerable Sets*, preprint.

[Ers68a] Y. I. Ershov, *On a hierarchy of sets I*, Algebra and Logic, Vol 7, 1968, pages 25–43.

[Ers68b] Y. I. Ershov, *On a hierarchy of sets II*, Algebra and Logic, Vol 7, 1968, pages 212–232.

[Ers70] Y. I. Ershov, *On a hierarchy of sets III*, Algebra and Logic, Vol 9, 1970, pages 20–31.

[FW98] M. Fairtlough and S. S. Wainer, *Hierarchies of Provably Recursive Funcitons*, in: Samuel R. Buss, Editor(s), Studies in Logic and the Foundations of Mathematics, Elsevier, 1998, Volumn 137, Handbook of Proof Theory, pages 149–207.

[Fis65] P. C. Fischer, *Theory of Provable Recursive Functions*, Transactions of the American Mathematical Society, 1965, Vol 117, pages 494–520.

[Fri57] R. M. Friedberg, *A criterion for completeness of degrees of unsolvability*, Journal of Symbolic Logic, Vol 22, 1957, pages 159–160.

[Fri98] H. Friedman, *Internal finite tree embeddings*, in: Reflections on the foundations of mathematics, Lecture Notes in Logic, 15, Association for Symbolic Logic, 1998, pages 60–91.

[Gab04] Y. Gabay, *Double jump inversions and strong minimal covers in the Turing degrees*, 2004, Ph.D. thesis.

[GMS04] N. Greenberg, A. Montalbán, R. Shore, *Generalized high degrees have the complementation property*, Journal of Symbolic Logic, 69, 2004, pages 1200–1220.

[Gol65] E. M. Gold, *Limiting recursion*, Journal of Symbolic Logic, Vol 30, 1965, pages 28–48.

[Har98] V. Harizanov, *Turing degrees of certain isomorphic images of computable relations*, Annals of Pure and Applied Logic, Vol 93, 1998, pages 103–113.

[HaS82] L. Harrington and S. Shelah, *The undecidability of the recursively enumerable degrees (research announcement)*, Bulletin of the American Mathematical Society, N.S. Vol 6, 1982, pages 79–80.

[HSh07] D. Hirschfeldt and R. A. Shore, *Combinatorial Principles Weaker than Ramsey's Theorem for Pairs*, Journal of Symbolic Logic, Vol 72, 2007, pages 171–206.

[Hod93] W. Hodges, *Model Theory*, Cambridge University Press, Cambridge U.K., 1993.

[Ish99] S. Ishmukhametov, *Weak recursive degrees and a problem of Spector*, Recursion Theory and Complexity, Arslanov and Lempp eds., de Gruyter, 1999, pages 81–89.

[Joc69] C. Jockusch, *The Degrees of Hyperhyperimmune Sets*, The Journal of Symbolic Logic, Volume 34, Issue 3 (1969), pages 489–493.

[Joc77] C. Jockusch, *Simple proofs of some theorems on high degrees*, Canadian Journal of Mathematics, Vol 29, 1977, pages 1072–1080.

[JLSS89] C. G. Jockusch, M. Lerman, R. I. Soare and R. M. Solovay, *Recursively Enumerable Sets Modulo Iterated Jumps and Extensions of Arslanov's Completeness Criterion*, The Journal of Symbolic Logic Vol. 54, No. 4, 1989, pages 1288–1323.

[JP78] C. Jockusch and D. Posner, *Double jumps of minimal degrees*, The Journal of Symbolic Logic, Vol. 43, No. 4, 1978, pages 715–724.

[JSo72] C. Jockusch and R. Soare, Π^0_1 *classes and degrees of theories*, Trans. Amer. Math. Soc., 173 (1972), page 33–56.

[JSh84] C. G. Jockusch, Jr. and R. A. Shore, *Pseudo-jump operators II: transfinite iterations, hierarchies and minimal covers*, Journal of Symbolic Logic, Vol 49, 1984, pages 1205–1236.

[KlPo54] S. C. Kleene and E. L. Post, *The upper semi-lattice of degrees of recursive unsolvability*, Annals of Mathematics, Vol 59, 1954, pages 379–407.

[Kri98] L. Kristiansen, *A jump operator on honest subrecursive degrees*, Archive for Mathematical Logic 37 (1998), pages 105–125.

[Kri99] L. Kristiansen, *Low*$_n$*, high*$_n$*, and intermediate subrecursive degrees*, in Calude and Dinneen, Editor(s), Combinatorics, computation and logic (Proceedings), Springer, Singapore 1999, pages 286–300.

[KiPa82] L. Kirby and J. Paris, *Accessible independence results for Peano Arithmetic*, Bulletin of the London Mathematical Society, 1982, volume 14, pages 285–293.

[KSWxx] L. Kristiansen, J. Schlage-Puchta and A. Weiermann, *Streamlined Subrecursive Degree Theory*, preprint.

[Ku86] A. Kučera, *An alternative, priority-free, solution to Post's problem*, in: J. Gruska, B. Rovan and J. Wiedermann, Editor(s), Mathematical Foundations of Computer Science 1986, Lecture Notes in Computer Science, Volume 233, 1986, pages 493–500.

[Ku94] A. Kučera, *Measure, Π^0_1-classes and complete extensions of PA*, Recursion Theory Week (Oberwolfach, 1984), volume 1141 of Lecture Notes in Math. 245-259, Springer, Berlin.

[KLxx] M. Kumabe and A. E. M. Lewis, *A Fixed Point Free Minimal Degree*, Journal of the London Mathematical Society, preprint.

[Lac66] A. H. Lachlan, *Lower bounds for pairs of recursively enumerable degrees*, Proceedings of the London Mathematical Society, Vol 16, issue 3, 1966, pages 537–569.

[Lac68] A. H. Lachlan, *Distributive initial segments of the degrees of unsolvability*, Z. Math. Logik Grund. Math., Vol 14, 1968, pages 457–472.

[Ler83] M. Lerman, *Degrees of Unsolvability, Local and Global Theory*, Perspectives in Mathematical Logic, 1983, Springer-Verlag.

[Ler86] M. Lerman, *Degrees which do not bound minimal degrees*, Annals of Pure and Applied Logic, Vol 30, 1986, pages 249–276.

[LS88] M. Lerman and R. A. Shore, *Decidability and Invariant Classes for Degree Structures*, Transactions of the American Mathematical Society, Vol 310, No. 2, 1988, pages 669-692.

[Lew06] A. E. M. Lewis, *Strong minimal covers and a question of Yates: the story so far*, in: B.S. Cooper, H. Geuvers, A. Pillay, J. Väänänen, Editor(s), Lecture Notes in Logic, Vol 32, Logic Colloquium 2006, pages 213–228.

[Lew07] A. E. M. Lewis, Π^0_1 *classes, strong minimal covers and hyperimmune-free degrees*, Bulletin of the London Mathematical Society, volume 39, number 6, 2007, pages 892–910.

[Lewa] A. E. M. Lewis, *Properties of the jump classes*, Journal of Logic and Computation, preprint.

[Lewb] A. E. M. Lewis, *A note on the join property*, Proceedings of the American Mathematical Society, preprint.

[LewMon] A. E. M. Lewis and A. Montalbán, personal communication.

[Loe92] M. Loebl, *Unprovable combinatorial statements*, Discrete Mathematics, Vol 108, 1992, pages 333-342.

[Ma66] D. A. Martin, *Classes of recursively enumerable sets and degrees of unsolvability*, Z. Math. Logik und Grund. der Math, 12, 1966, pages 295–310.

[Mo06] A. Montalbán, *There is no ordering on the classes in the generalized high/low hierarchies*, Archive for Mathematical Logic, 45 (2006), pages 215–231.

[MM68] W. Miller and D. A. Martin, *The Degrees of hyperimmune sets*, Zeitschrift-fur Mathematische Logik und Grundlagen der Mathematik, vol. 14 (1968), pages 159–166.

[Nie08] A. Nies *Computability and randomness*, Clarendon Press, Oxford, 2008.

[NSS98] A. Nies, R. A. Shore and T. A. Slaman, *Interpretability and definability in the recursively enumerable degrees*, Proceedings of the London Mathematical. Society, Vol 77, issue 3, 1998, pages 241–291.

[Ng09] K. M. Ng, *Computability, Traceability and Beyond*, 2009, Ph.D. Thesis.

[PH77] J. Paris and L. Harrington *A mathematical incompleteness in Peano Arithmetic*, in: J. Barwise, Editor(s), Studies in Logic and Foundations of Mathematics, Elsevier, 1977, Volumn 90, Handbook of Mathematical Logic pages 1133-1142.

[Poh98] W. Pohlers *Set Theory and Second Order Number Theory*, in: Samuel R. Buss, Editor(s), Studies in Logic and the Foundations of Mathematics, Elsevier, 1998, Volumn 137, Handbook of Proof Theory, pages 209–335.

[Po44] E. L. Post, *Recursively enumerable sets of positive integers and their decision problems*, Bulletin of American Mathematical Society, Vol 50, 1944, pages 284–316.

[Put65] H. Putnam, *Trial and error predicates and the solution to a problem of Mostowski*, Journal of Symbolic Logic, Vol 30, 1965, pages 44–50.

[Rob71] R. W. Robinson, *Jump restricted interpolation in the recursively enumerable degrees*, Annals of Mathematics, 1971, Vol 93, issue 2, pages 586-596.

[Rog87] H. Rogers, *The Theory of Recursive Functions and Effective Computability*, MIT Press, 1987.

[Sac61] G. Sacks, *A Minimal Degree less than* $\mathbf{0}'$, Bulletin of American Mathematical Society, 67 (1961), pages 416–419.

[Sac63] G. Sacks, *On the degrees less than* $0'$, Annals of Mathematics, Vol 77, pages 211–231.

[Sac66] G. Sacks, *Degrees of unsolvability*, Annals of Math. Studies **55**, Princeton Univ. Press, 2^{nd} ed., Princeton NJ, 1966.

[Sac85] G. Sacks, *Some open questions in recursion theory*, Lecture Notes in Mathematics, Vol 1141, Springer Verlag Berlin, Heidelberg, New York, Tokyo, 1985, pages 333–342.

[Sas74] L. Sasso, Jr., *A minimal degree not realizing least possible jump*, The Journal of Symbolic Logic, Vol. 39, No. 3, 1974, pages 571–574.

[Shn59] J. R. Shoenfield, *On degrees of unsolvability*, Annals of Mathematics, 1959, Vol 69, issue 2, pages 644–653.

[Shn65] J. R. Shoenfield, *An application of model theory to degrees of unsolvability*, in Symposium on the Theory of Models, J. W. Addison, L. Henkin and A. Tarski eds., North-Holland, Amsterdam, 1965, pages 359–363.

[Sh81] R. A. Shore, *The theory of the degrees below* $0'$, Journal of the London Mathematical Society, Vol 24, 1981, pages 1–14.

[Sh85] R. A. Shore, *The structure of the degrees of unsolvability*, Recursion Theory, Proceedings of the Symposia in Pure Mathematics 42, A. Nerode and R. A. Shore, eds., American Mathematical Society, Providence, R.I., 1985, pages 33–51.

[Sh88] R. A. Shore, *Defining jump classes in the degrees below* $0'$, Proceedings of the American Mathematical Society, Vol 104, 1988, pages 287–292.

[Sh99] R. A. Shore, *The recursively enumerable degrees*, in Handbook of Recursion Theory, E. Griffor ed., North-Holland, Amsterdam, 1999, pages 169–197.

[Sh06] R. A. Shore, *Degree structures: Local and global investigations*, Bulletin of Symbolic Logic, Vol 12, 2006, pages 369–389.

[Sh07] R. A. Shore, *Direct and local definitions of the Turing jump*, Journal of Mathematical Logic, Vol 7, 2007, pages 229–262.

[Sh10] R. A. Shore, *Reverse Mathematics: the Playground of Logic*, Bulletin of Symbolic Logic 16 (2010), pages 378–402.

[Shxx] R. A. Shore, *The Turing Degrees: Global and Local Structure*, draft.

[SS99] R. A. Shore and T. A. Slaman, *Defining the Turing jump*, Mathematical Research Letters, Vol 6, 1999, pages 711–722.

[Sim77] S. G. Simpson, *First order theory of the degrees of recursive unsolvability*, Annals of Mathematics, Vol 105, issue 2, 1977, pages 121–139.

[Sim85] S. G. Simpson, *Nonprovability of certain combinatorial properties of finite trees* in Leo Harrington, et.al. (editors), Harvey Friedman's Research in the Foundations of Mathematics, North-Holland, Amsterdam, 1985, pages 87–117.

[Sim10] S. G. Simpson, *Subsystems of second order arithmetic*, Perspectives in Logic, Cambridge University Press, second edition, 2010.

[Sla83] T. A. Slaman, *The recursively enumerable degrees as a substructure of the Δ^0_2 degrees*, handwritten notes.

[Sla91] T. A. Slaman, *Degree structures*, in Proc. Int. Cong. Math., Kyoto 1990, Springer-Verlag, Tokyo, 1991, pages 303–316.

[SW01] T. A. Slaman and H. Woodin, *Definability in Degree Structures*, preprint.

[Sp56] C. Spector, *On Degrees of Recursive Unsolvability*, Annals of Mathematics, Vol 64, No. 3, 1956, pages 581–592.

[SYY09] F. Stephan, Y. Yang and L. Yu, *Turing degrees and the Ershov hierarchy* in Proceedings of the Tenth Asian Logic Conference, Kobe, Japan, 1-6 September 2008, World Scientific, 2009, pages 300–321.

[Ta62] M. A. Taitslin, *Effective inseparability of the sets of identically true and finitely refutable formulas of elementary lattice theory*, Algebra i Logika, Vol 3, 1962, pages 24–38.

[Tur39] A. M. Turing, *Systems of logic based on ordinals*, Proceedings of the London Mathematical Society, Vol 45, issue 3, 1939, pages 161–228.

[Wangxx] W. Wang, *Relative enumerability and 1-genericity*, Journal of Symbolic Logic, Journal of Symbolic Logic, preprint.

[Yu06] L. Yu,[2006], *Lowness for genericity*, Archive for Mathematical Logic, Vol 45, 2006, pages 233–238.

[YY06] Y. Yang and L. Yu, $\mathcal{R}$ *is not a* Σ_1*-elementary substructure of* $\mathcal{D}_n$, Journal of Symbolic logic, 71(2006), No.4, pages 1223–1236.

[Yat66] C. E. M. Yates, *A minimal pair of recursively enumerable degrees*, Journal of Symbolic Logic, Vol 31, 1966, pages 159–168.