Computability 1 (2012) 1–48
DOI
IOS Press

The search for natural definability in the Turing degrees

Andrew E.M. Lewis-Pye
Department of Mathematics, London School of Economics, UK
andy@aemlewis.co.uk
aemlewis.co.uk

1. Introduction

These notes came out of a course which I gave at Notre Dame a few years ago now, and which was essentially a course in the Turing degrees aimed at PhD students who had some experience with computability theory, but not necessarily with techniques specific to the study of the local degrees. The programme of study described is heavily influenced by the research of Barry Cooper, and so my hope is that this paper should be a fitting contribution to this memorial issue in his honour. Much of the material covered revolves around the jump classes, which were introduced by Barry and Bob Soare back in the 1970s.

The course assumes very little background knowledge: familiarity with the concepts of a Turing functional, a Turing degree, the halting problem and such things are assumed, but nothing more advanced than this. While the earlier material is obviously aimed at the reader who does not have much experience working with the Turing degree structure, my belief/hope is that even the seasoned professional will find much to interest them, especially in later sections. Despite the intensive work which has been put into the study of the Turing degrees, many very basic questions remain open. I've listed a good number here, together with descriptions of techniques which might sometimes suffice to give solutions.

The course may be thought of as being divided roughly into two parts:

Forming a picture of the Turing degrees. In the earlier sections of the course we shall establish some basic structural properties of the Turing degrees by answering the following kind of questions:

- Are there incomparable Turing degrees? What are the largest chains and anti-chains?
- Is the structure a lattice?
- Is the structure dense? We call a degree $\mathbf{a}$ *minimal* if $\boldsymbol{a} > \mathbf{0}$ and there doesn't exist any degree $\boldsymbol{b}$ with $\mathbf{0} < \boldsymbol{b} < \boldsymbol{a}$. Are there minimal degrees? If so, where can they be found?

As an introduction to some of the techniques we'll be using later, we will then take a look at some of the structural properties satisfied by $\mathbf{0}'$. In particular, we shall show that $\mathbf{0}'$ satisfies the cupping, join and meet properties – of course these properties will be defined in what follows.

The *jump hierarchy*, roughly speaking, is a way of measuring the position of a degree by comparing its nth jump with the nth jump of $\mathbf{0}$. A *jump class* consists of the set of degrees in one level of this hierarchy. We shall introduce the jump classes first for the local case, as originally suggested by Cooper in [BC] and Soare in [RS], and then we shall consider the generalized version suggested by Jockusch and Posner [JP], and we shall show that these jump classes are all distinct.

There are many classes of degrees used by computability theorists which are defined according to properties of their members rather than structural properties of the degree itself. Three of these classes will be of particular use to us in our analysis of the jump classes; we shall define and prove some basic properties of the PA, the a.n.r. and the 1-generic degrees. With all this preparation in place we shall then move on to consider more advanced topics.

Properties of the jump classes. A relation on the Turing degrees is said to be *definable* if there is a formula in the language of partial orders which is true of precisely those tuples of degrees in the relation. Computability theorists have been able to establish a good number of definability results. Shore and Slaman [SSl] showed, for example, that the jump is definable, while Nies, Shore and Slaman [NSS] showed that all the jump classes other than low are definable (a degree $\boldsymbol{a}$ is low if $\boldsymbol{a}' = \mathbf{0}'$, and the definability of the low degrees remains an open question). With some notable exceptions, however, these definability results have been established through the use of techniques which involve coding models of arithmetic into the degree structure. Such methods will not be covered in this course—for an introduction we refer the reader to [AN2]. While the success of these coding techniques is beyond question, a disadvantage is that these proofs of definability do not yield what might be considered *natural* definitions. The notion of a natural definition here is not precise, but by a natural definition what we essentially mean is a formula which suffices to define the relevant class and which is easy to understand. Roughly speaking, then, this means that the definition should not be too long and should not involve too many alternations of quantifier. It remains one of the holy grails for researchers in the area to find natural definability results for the jump classes, or for the jump function itself. In the second part of the course we shall detail some of the work[1] that has been done towards achieving this goal, by systematically studying which basic order theoretic properties are satisfied by the degrees in each jump class. The idea here is to begin with very simple properties and then gradually move on to consider properties which are more complex – the hope being that at some point natural definability results will precipitate out as a result of this process.

2. Turing reductions, and Turing degrees; A review of notation and some basic facts

2.1. Turing functionals

We shan't give the definition of a Turing machine and a Turing functional here (just look in any introductory textbook, such as [BC2]). We suppose we are given a fixed effective[2] listing of all the Turing functionals;

- For each $i \in \omega$ we write Ψ_i in order to denote the ith Turing functional in this list;
- Each Turing functional occurs infinitely many times in this list, i.e. for each i there exist infinitely many j for which $\Psi_i = \Psi_j$.
- For $A \subseteq \omega$ and $n \in \omega$, we write $\Psi_i(A;n)$ in order to denote the output of Ψ_i given oracle input A and on argument n (so if this computation does not converge then $\Psi_i(A;n)\uparrow$). We also write $\Psi_i(n)$ in order to denote $\Psi_i(\emptyset;n)$.
- We write $\Psi_i(A)$ in order to denote the (possibly) partial function which on argument n is equal to $\Psi_i(A;n)$. We identify subsets of ω with their characteristic functions, so that it makes sense, for example, to write $\Psi_i(A) = B$.

We let $2^{<\omega}$ denote the set of finite binary strings. When f and g are (possibly) partial functions, we write $f \subseteq g$ in order to denote that the domain of f is a subset of the domain of g and $f(n) = g(n)$ for all n in the domain of f. We say that f and g are incompatible if there exists some n such that $f(n)\downarrow$ and $g(n)\downarrow$ but $f(n) \neq g(n)$. We let $\langle \cdot, \cdot \rangle$ be a computable bijection $\omega \times \omega \to \omega$. For finite strings σ and τ, $\sigma * \tau$ denotes the concatenation of σ and τ. For a finite string σ, we let $|\sigma|$ denote the length of σ.

In fact the particulars of the oracle Turing machine definition will normally be of no importance to us. The only things that we need to be true of oracle Turing machines, at least for the vast majority of our proofs to go through, are the following:

1. The Turing functionals include and are closed under composition with all algorithmically calculable functions $2^\omega \times \omega \to \omega$;

[1] A notable omission is some of the recent work of Shore on natural definability, see for example [SH2]

[2] Throughout the course the word "effective" is used synonymously with "algorithmic", so by an effective listing of the Turing functionals we mean that there is an algorithm which, given any $i \in \omega$, produces the instructions for the ith functional in the list.

2. Turing functional computations take place in stages. If the computation $\Psi_i(A;n)$ does not converge in s stages then $\Psi_i(A;n)[s]\uparrow$, otherwise we define it to be equal to the value outputted by the computation. If $\Psi_i(A;n)[s]\downarrow$ then $\Psi_i(A;n)[t]\downarrow = \Psi_i(A;n)[s]$ for all $t \geq s$.
3. When a computation halts it does so in a finite number of stages and therefore only a finite number of bits of the oracle tape can be scanned. We write $\Psi_i(\sigma;n)\downarrow = m$ if $\Psi_i(A;n)\downarrow = m$ and this computation converges after scanning only bits of the oracle tape $< |\sigma|$. So if $\Psi_i(A;n) = m$ there exists $\sigma \in 2^{<\omega}$ which is an initial segment of A, such that $\Psi_i(B;n) = m$ for all $B \supset \sigma$.
4. There exists a universal machine, i.e. there exists i such that for all A,j,n, we have $\Psi_i(A;\langle j,n\rangle) \simeq \Psi_j(A;n)$ (where $\simeq$ denotes that either both values are undefined or else both are defined and equal).

2.2. Turing degrees

When there exists i such that $\Psi_i(B) = A$ we say that A is *Turing reducible* to B, denoted $A \leq_T B$. When we say that A is *computable* in B this is just another way of saying that A is Turing reducible to B.

- When $A \leq_T B$ and $B \leq_T A$ we say that A and B are *Turing equivalent*, denoted $A \equiv_T B$.
- The *Turing degree* of A is the set of B such that $B \equiv_T A$. Each Turing degree is therefore a countable collection of sets.
- For Turing degrees $\boldsymbol{a}$ and $\boldsymbol{b}$ we define $\boldsymbol{a} \leq \boldsymbol{b}$ if $A \leq_T B$ for all $A \in \boldsymbol{a}$ and $B \in \boldsymbol{b}$. Since $\leq_T$ is transitive, this latter condition will hold precisely when there exist *any* $A \in \boldsymbol{a}$ and $B \in \boldsymbol{b}$ with $A \leq_T B$.
- Given $X_0, X_1 \subseteq \omega$, we let $X_0 \oplus X_1$ denote the set whose characteristic function is defined as follows: for all n, $X_0 \oplus X_1(2n) = X_0(n)$ and $X_0 \oplus X_1(2n+1) = X_1(n)$. We write $\oplus_{i=0}^{k} X_i$ in order to denote $((\cdots(X_0 \oplus X_1) \oplus X_2)\cdots \oplus X_k)$.
  Since any set which computes A and computes B also computes $A \oplus B$, and since $A \oplus B$ clearly computes both of A and B, every pair of Turing degrees has a least upper bound.

2.3. The jump and c.e. sets

For any set A, A' is the *halting set* for A:

$$A' = \{i \ : \Psi_i(A;i)\downarrow\}.$$

. The basic facts are as follows:

- A' is always of degree *strictly above* the degree of A;
- If $A \leq_T B$ then $A' \leq_T B'$;
- For any degree $\boldsymbol{a}$ we can therefore define $\boldsymbol{a}'$, the *jump* of $\boldsymbol{a}$, to be the degree of A' for $A \in \boldsymbol{a}$. In particular, $\boldsymbol{0}'$ denotes the jump of $\boldsymbol{0}$ and for all $\boldsymbol{a}$ we have $\boldsymbol{a}' \geq \boldsymbol{0}'$.
- A set is *computably enumerable* (c.e.) if there is an algorithm which enumerates the elements of the set (but not necessarily in increasing order). Formally, $A \subseteq \omega$ is c.e. if there exists i such that $A = \{n : \Psi_i(n)\downarrow\}$. We let W_i denote the ith c.e. set, so $W_i = \{n : \Psi_i(n)\downarrow\}$. We also let $W_{i,s}$ denote the set of numbers enumerated into W_i by stage s, i.e. $W_{i,s} = \{n : \Psi_i(n)[s]\downarrow\}$, and we assume that if $\Psi_i(n)[s]\downarrow$ then $s \geq n$.
- The halting set $\emptyset'$ is an example of a c.e. set which is not computable.
- The computably enumerable (c.e.) degrees are those containing computably enumerable sets. Since any c.e. set is computable in $\emptyset'$, all c.e. degrees are below $\boldsymbol{0}'$.
- The sets of degree below $\boldsymbol{0}'$ are precisely those that can be computably approximated, i.e. those A for which there exists a computable sequence of finite sets $\{A_s\}_{s\in\omega}$ such that, for all n there exists t, $A_s(n) = A(n)$ for all $s \geq t$.
- Various important sets are of degree $\boldsymbol{0}^{(2)}$ or $\boldsymbol{0}^{(3)}$ (where $\boldsymbol{a}^{(n)}$ denotes the nth jump of $\boldsymbol{a}$, i.e. the result when we start with $\boldsymbol{a}$ and apply the jump operator n times). Examples are Tot=$\{i : (\forall n)\Psi_i(n)\downarrow\}$ which is of degree $\boldsymbol{0}^{(2)}$, and Comp which is the set of all i such that W_i is computable and is of degree $\boldsymbol{0}^{(3)}$.

3. Chains, antichains and independent sets

The fact that $\boldsymbol{a}'$ is always strictly above $\boldsymbol{a}$ suffices to show us that there does not exist a greatest Turing degree. Perhaps the next natural question is whether there exists a pair of incomparable Turing degrees.

Theorem 3.1 ([KP]) *There exist incomparable Turing degrees.*

Proof. The basic format of the proof is one that very soon becomes familiar to anybody who works in computability theory. Proofs in computability theory very often involve constructing a number of subsets of ω – in this case we need to construct two sets A and B. We write down a countable list of *requirements*, satisfaction of which will suffice to give the theorem:

$\mathscr{P}_{2s}$: $\Psi_s(A) \neq B$;
$\mathscr{P}_{2s+1}$: $\Psi_s(B) \neq A$.

The even requirements suffice to ensure that A does not compute B and the odd requirements suffice to ensure that B does not compute A.

In order to construct A and B so as to satisfy these requirements it suffices, in this case, to define a *finite extension argument*. This means that we define a construction which proceeds in stages, and we decide finite initial segments $\alpha_s \subset A$ and $\beta_s \subset B$ at each stage. Initially we define $\alpha_0 = \beta_0 = \emptyset$ (the string of length 0). At stage $s+1$, given α_s and β_s, we define $\alpha_{s+1} \supset \alpha_s$ and $\beta_{s+1} \supset \beta_s$ in such a way as to ensure that the next requirement in our list is satisfied. Then ultimately we define A to be the infinite string extending all α_s and we define B to be the infinite string extending all β_s. The precise instructions are as follows:

Stage 0: Define $\alpha_0 = \beta_0 = \emptyset$.
Stage $2s+1$: We are given α_{2s} and β_{2s}. We look to satisfy $\mathscr{P}_{2s}$. Let n be the least such that $\beta_{2s}(n)\uparrow$. We ask,

"does there exist any string $\alpha \supset \alpha_{2s}$ such that $\Psi_s(\alpha;n)\downarrow$?"

If NOT: then since A will extend α_{2s}, $\Psi_s(A;n)\uparrow$, so that requirement $\mathscr{P}_{2s}$ will be satisfied. In this case we can just define α_{2s+1} to be any finite extension of α_{2s} and we can define β_{2s+1} to be any finite extension of β_{2s}.

If SO: then let α_{2s+1} be the least such α (we may consider the finite binary strings ordered according to length and then from left to right). Define β_{2s+1} to be the one element extension of β_{2s} which disagrees with $\Psi_s(\alpha_{2s+1})$ on argument n. Since A will extend α_{2s+1} and B will extend β_{2s+1}, we have ensured that $\Psi_s(A) \neq B$.

Stage $2s+2$: Now we look to satisfy $\mathscr{P}_{2s+1}$. We proceed exactly as at stage $2s+1$ but with the roles of A and B reversed. □

An analysis of this proof allows us to deduce a stronger result. Which oracle do we need in order to carry out the construction, and so compute the characteristic functions of A and B? It suffices to be able to answer the following question for any $\sigma \in 2^{<\omega}$ and any $n,s \in \omega$:

"does there exist any string $\tau \supset \sigma$ such that $\Psi_s(\tau;n)\downarrow$?"

We can easily devise an algorithm which, given any $\sigma \in 2^{<\omega}$ and any $n,s \in \omega$, searches for $\tau \supset \sigma$ such that $\Psi_s(\tau;n)\downarrow$, and which terminates if and only if it finds such a string. An oracle for $\emptyset'$ can tell us whether or not this algorithm terminates, and so suffices to answer the question above. We therefore get:

Theorem 3.2 ([KP]) *There exist incomparable degrees below* $\mathbf{0}'$.

Now we look to extend this result.

Definition 3.1 *For* $i \in \{0,1\}$, *let* $\bar{i}$ *denote* $1-i$. *A partial function* $T : 2^{<\omega} \to 2^{<\omega}$ *is a* ***tree*** *if, for all* $\sigma \in 2^{<\omega}$ *and* $i \in \{0,1\}$, *when* $T(\sigma * i)\downarrow$:

1. $T(\sigma)\downarrow \subset T(\sigma * i)$;
2. $T(\sigma * \bar{i})\downarrow$ *and is incompatible with* $T(\sigma * i)$.

If T *is a total function of this kind, we also call it a* ***perfect*** *tree. The following abuse of terminology is standard: for any string* τ, *we say* τ ***is in*** T *if* τ *is in the range of* T. *Then we say* $A \subset \omega$ *is a* ***path*** *through* T *if there exist infinitely many* $\tau \subset A$ *in* T *(sometimes, for the sake of emphasis, we may also refer to* A *as an* infinite *path in this case). If* A *is a path through* T *we may also say that* A ***lies on*** T.

We say that T_1 *is a* ***subtree*** *of* T_0, *denoted* $T_1 \subseteq T_0$, *if the range of* T_1 *is a subset of the range of* T_0.

Definition 3.2 *A sequence of sets* $\{X_i\}_{i\in\omega}$ *is* ***independent*** *if, for all finite* $F \subseteq \omega$ *and all* $j \notin F$, $X_j \not\leq_T \oplus_{i\in F} X_i$. *A sequence of degrees* $\{\boldsymbol{a}_i\}_{i\in\omega}$ *is independent if there exists a sequence* $\{X_i\}_{i\in\omega}$ *which is independent with* $X_i \in \boldsymbol{a}_i$.

It is natural to ask next, what are the largest antichains we can find in the Turing degrees? By using the same techniques we used in order to prove Theorem 3.1 it is not difficult to construct a perfect tree T such that any two distinct paths through T are of incomparable degree. This gives us an antichain of cardinality the continuum, which is clearly the best we can do since there are only continuum many Turing degrees. Precisely the same techniques can also be used to show:

Theorem 3.3 ([KP]) *There exists an independent sequence of degrees.*

Corollary 3.1 ([KP]) *Every finite poset can be embedded in the Turing degrees.*

Proof. Let $\mathscr{M} = \langle M, \leq \rangle$ be a finite poset, and let $M = \{x_i : i < n\}$. Let $\{\boldsymbol{a}_i\}_{i\in\omega}$ be an independent sequence of degrees and let A_i be of degree $\boldsymbol{a}_i$. For each $k < n$ let $F(k)$ be the set of all i such that $x_i \leq x_k$, put $B_k = \oplus_{i\in F(k)} A_i$ and define $\boldsymbol{b}_k$ to be the degree of B_k.

We define an embedding as follows: $g(x_i) = \boldsymbol{b}_i$ for every $i < n$. In order to verify that this is indeed an embedding we must show that for all $i,j < n$, $x_i \leq x_j \Leftrightarrow B_i \leq_T B_j$. Suppose first that $x_i \leq x_j$. Then $F(i) \subseteq F(j)$ so the result follows immediately. Next suppose that $B_i \leq_T B_j$ and $x_i \not\leq x_j$ in order to derive a contradiction. Then $A_i \leq_T B_i \leq_T B_j$, so $A_i \leq_T \oplus_{k\in F(j)} A_k$ and $i \notin F(j)$, which contradicts the choice of $\{A_i\}_{i\in\omega}$ as an independent sequence. □

By the $\exists_1$-theory of the Turing degrees, we mean the set of sentences in the language of partial orders which are true of the Turing degrees, and are of the form $\exists x_1 \exists x_2 \cdots \exists x_k R(x_1, \cdots, x_k)$ for some k, where $R(x_1, \cdots, x_k)$ is a quantifier free expression with free variables $x_1, \cdots, x_k$.

Corollary 3.2 *The* $\exists_1$*-theory of the Turing degrees is decidable.*

Proof. An $\exists_1$ statement asserts the existence of finitely many degrees $\boldsymbol{a}_1, .., \boldsymbol{a}_k$, and for some pairs $i < j$ it asserts that $\boldsymbol{a}_i \leq \boldsymbol{a}_j$, while for other pairs it asserts that $\boldsymbol{a}_i \not\leq \boldsymbol{a}_j$. By Corollary 3.1, this will hold iff there is a finite poset satisfying these conditions. In order to test whether this is the case involves running through a finite number of possibilities, and so is decidable. □

We have already seen that there does not exist a largest Turing degree. How large can a chain of Turing degrees be? Given any countable set of Turing degrees, there exists a Turing degree strictly above every member of the set – given $\{A_i\}_{i\in\omega}$, consider B such that $B(\langle i,j \rangle) = A_i(j)$ and then take B'. Therefore there exist chains of length ω_1 (the

least uncountable ordinal). On the other hand, there cannot exist chains of length greater than ω_1 since each Turing degree has only countably many predecessors.

4. The Turing degrees as an upper semi-lattice

Definition 4.1 *A partial order $\mathscr{P}$ is an **upper semi-lattice** if, for all $x,y \in \mathscr{P}$, there exists a least degree which is above both x and y, denoted $x \vee y$. We call $x \vee y$ the **join** of x and y.*

*We say $\mathscr{P}$ is a lattice if it is an upper semi-lattice and for all $x,y \in \mathscr{P}$, there exists a greatest degree which is below both x and y, denoted $x \wedge y$. We call $x \wedge y$ the **meet** of x and y.*

As was mentioned previously, it is clear that any set which computes both A and B also computes $A \oplus B$ and the Turing degrees are therefore an upper semi-lattice. In order to show that they are **not** a lattice we need to consider countable ideals of degrees.

Definition 4.2 *If $\mathscr{P}$ is an upper semi-lattice then $\mathscr{I} \subseteq \mathscr{P}$ is an **ideal** if:*

1. *Whenever $x,y \in \mathscr{I}$, their join $x \vee y$ is in $\mathscr{I}$;*
2. *Whenever $x \in \mathscr{I}$ and $y \leq x$, y is in $\mathscr{I}$.*

A set of Turing degrees $\mathscr{X}$ is said to be *definable with parameters* if there is a finite set of Turing degrees $\boldsymbol{a}_1,\cdots,\boldsymbol{a}_k$ and a formula in the language of partial orders $F(x_0,\cdots,x_k)$ such that, for all degrees $\boldsymbol{a}$, $\boldsymbol{a} \in \mathscr{X}$ iff $\mathscr{F}(\boldsymbol{a},\boldsymbol{a}_1,\cdots,\boldsymbol{a}_k)$ is true in the Turing degrees. The next theorem shows that every countable ideal in the Turing degrees is definable with parameters – one just needs to specify an exact pair for the ideal.

Definition 4.3 *If $\mathscr{P}$ is a partial order and $\mathscr{I} \subseteq \mathscr{P}$, we say that (x,y) is an **exact pair** for $\mathscr{I}$ if, for all $z \in \mathscr{P}$, $z \in \mathscr{I}$ iff $z \leq x$ and $z \leq y$.*

Theorem 4.1 ([CS]) *Every countable ideal in the Turing degrees has an exact pair.*

Corollary 4.1 *The Turing degrees are not a lattice.*

Proof. Let $\{\boldsymbol{x}_i\}_{i\in\omega}$ be a strictly increasing sequence of degrees (we could set $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i'$, for example). Let $\mathscr{I}$ be the ideal generated by this sequence, i.e. any degree $\boldsymbol{c}$ is in $\mathscr{I}$ iff there exists $\boldsymbol{x}_i \geq \boldsymbol{c}$. Now let $(\boldsymbol{a},\boldsymbol{b})$ be an exact pair for $\mathscr{I}$. If $\boldsymbol{c} \leq \boldsymbol{a}$ and $\boldsymbol{c} \leq \boldsymbol{b}$ then $\boldsymbol{c} \in \mathscr{I}$ and there exists $\boldsymbol{x}_i \geq \boldsymbol{c}$. Then $\boldsymbol{x}_{i+1}$ is also below both $\boldsymbol{a}$ and $\boldsymbol{b}$ and is strictly above $\boldsymbol{c}$. Thus $\boldsymbol{a}$ and $\boldsymbol{b}$ have no greatest lower bound. □

Now we prove Theorem 4.1.

Proof. We suppose that we are given an enumeration $\{X_s\}_{s\in\omega}$ of all sets which are of degree in $\mathscr{I}$. We must construct A and B so as to satisfy the requirements:

$\mathscr{P}_s$: $X_s \leq_T A$ and $X_s \leq_T B$;
$\mathscr{Q}_s$: Let $s = \langle i,j\rangle$. $\Psi_i(A) = \Psi_j(B) = C \Longrightarrow$ there exists k, $C = X_k$.

The $\mathscr{P}_s$ requirements ensure that every degree in the ideal is below both $\boldsymbol{a} = \deg(A)$ and $\boldsymbol{b} = \deg(B)$. Then the $\mathscr{Q}_s$ requirements ensure that anything computable in both A and B is of degree in the ideal. Clearly these requirements suffice to prove the theorem.

This time the proof will not be a finite extension argument, we shall describe a **co-infinite extension argument**. This means that at each stage in the construction, we may define A and B on an infinite number of arguments, but at the end of each stage we will also have left each of these sets undefined on an infinite number of arguments.

In order to ensure that each X_s is computable in both A and B, the basic idea is that we divide these sets up into columns. The sth column is all numbers of the form $\langle s,j\rangle$. If $A(\langle s,j\rangle) = X_s(j)$ for all j, then A will compute X_s – and similarly for B of course. In fact, we need slightly less than this. If, for each s, $A(\langle s,j\rangle) = X_s(j)$ *for all but finitely many* j, then this suffices to show that A computes each X_s. At each stage $s+1$ we shall ensure that $X_s \leq_T A$ and $X_s \leq_T B$ by coding X_s into the sth columns of A and B in this way.

Suppose that at the end of stage s we have already coded X_k into the kth column of A and B for each $k < s$, and suppose also that:

$(*)$ Outside the finite set of columns we have already used for coding, we have decided only finitely many arguments of A and B.

At stage $s+1$ we now look to satisfy $\mathscr{Q}_s$. Let α_s be the partial function which specifies A on the arguments we have already decided, and let β_s be the corresponding partial function for B. Note that, unless $s = 0$, α_s and β_s will not be finite strings, but instead will be defined on an infinite number of arguments as well as being undefined on an infinite number of arguments. We ask whether there are any extensions $\alpha \supset \alpha_s$ and $\beta \supset \beta_s$ for which $\Psi_i(\alpha)$ and $\Psi_j(\beta)$ are incompatible (and where $s = \langle i,j\rangle$).

If so, then there are finite extensions which satisfy this property. We can take these finite extensions, code X_s into what remains of the sth columns of A and B (which will still be all but finitely many numbers), and maintain the condition $(*)$.

If there is no way in which we can extend so as to force disagreement, then we will be able to show that if $\Psi_i(A) = \Psi_j(B)$ and is total, then it is computable in the columns of A and B which we have already determined. This finite set of columns is essentially the join of a finite number of sets of degree in the the ideal, and so is of degree in the ideal.

We now formally describe the construction.

Stage 0. Define $\alpha_0 = \beta_0 = \emptyset$.
Stage $s+1$. Let $s = \langle i,j\rangle$. We ask: does there exist $\alpha \supset \alpha_s$ and $\beta \supset \beta_s$ for which $\Psi_i(\alpha)$ and $\Psi_j(\beta)$ are incompatible? If so then let α and β be finite extensions of α_s and β_s respectively, which satisfy this condition. If not then let $\alpha = \alpha_s$ and let $\beta = \beta_s$.

Now let α_{s+1} be the least extension of α which is defined and equal to $X_s(j)$ on all arguments of the form $\langle s,j\rangle$ for which α is undefined. Let β_{s+1} be defined similarly in terms of β.

This ends the formal description of the construction.

In order to show that the construction suffices to satisfy the requirements, it remains to consider what happens when, at stage $s+1$, there do not exist $\alpha \supset \alpha_s$ and $\beta \supset \beta_s$ for which $\Psi_i(\alpha)$ and $\Psi_j(\beta)$ are incompatible. In this case we claim that, if $\Psi_i(A)$ and $\Psi_j(B)$ are both total (and so equal) then this common value is computable in $D = \oplus_{k=0}^{s-1} X_k$, and so is of degree in the ideal. Certainly D can decide which arguments α_s and β_s are defined on, and can compute their values on all such arguments. If $\Psi_i(A) = \Psi_j(B)$ is total, then in order to compute $\Psi_i(A;n)$, an oracle for D can proceed as follows. Find *any* extension α of α_s such that $\Psi_i(\alpha;n)\downarrow$. Such an extension must exist since A extends α_s and $\Psi_i(A;n)\downarrow$. Then it must be the case that $\Psi_i(\alpha;n) = \Psi_i(A;n)$. In order to see this, suppose otherwise. Then let β be a finite extension of β_s which is compatible with B and such that $\Psi_j(\beta;n)\downarrow$. Since $\Psi_i(A) = \Psi_j(B)$ it must be the case that $\Psi_i(\alpha;n) \neq \Psi_j(\beta;n)$ which is impossible by hypothesis. □

5. Minimal Degrees

One of the first questions one is likely to ask as one tries to understand a partial order is whether or not it is dense. In the case of the Turing degrees, non-density is established through the existence of minimal degrees.

Definition 5.1 *A Turing degree* $\boldsymbol{a}$ *is* ***minimal*** *if* $\boldsymbol{a} > \boldsymbol{0}$ *and there does not exist* $\boldsymbol{b}$ *with* $\boldsymbol{0} < \boldsymbol{b} < \boldsymbol{a}$.

Theorem 5.1 ([CS]) *There exist minimal Turing degrees.*

Proof. We wish to build a set A of minimal Turing degree. In order to do so, once again we begin by drawing up a list of requirements:

$\mathscr{P}_s$: $A \neq \Psi_s(\emptyset)$;
$\mathscr{Q}_s$: If $\Psi_s(A)$ is total then either it is computable or $A \leq_T \Psi_s(A)$.

The $\mathscr{P}_s$ requirements suffice to ensure that A is noncomputable. We already know a very simple technique for satisfying one of these requirements – find some argument n for which we have not yet specified $A(n)$, and if $\Psi_s(\emptyset;n)\downarrow$ then make $A(n)$ different to it. The $\mathscr{Q}_s$ requirements suffice to ensure that every set whose characteristic function is computable in A is either computable or else is of the same degree as A. In order to satisfy these requirements we need to consider splitting trees.

Definition 5.2 *Two strings σ,τ are Ψ_i-**splitting** if $\Psi_i(\sigma)$ and $\Psi_i(\tau)$ are incompatible. We may refer to a Ψ_i-splitting pair of strings just as a "Ψ_i-splitting".*

Definition 5.3 *A tree $T : 2^{<\omega} \to 2^{<\omega}$ is a Ψ_i-**splitting tree** if any two strings in T which are incompatible are also Ψ_i-splitting.*

Definition 5.4 *A tree $T : 2^{<\omega} \to 2^{<\omega}$ is a Ψ_i-**nonsplitting tree** if no pair of strings in T are Ψ_i-splitting.*

Definition 5.5 *We say τ is of **level** n in a tree T if $\tau = T(\sigma)$ for some σ of length n. We say τ is a **leaf** of T if $\tau \in T$ and there do not exist any proper extensions of τ in T. We say T is of level n if it is finite and all leaves are of level n.*

Note that a tree which is not Ψ_i-splitting will not necessarily be Ψ_i-nonsplitting. The following two lemmas are precisely what we need in order to develop a technique for satisfying the $\mathscr{Q}_s$ requirements.

Lemma 5.1. *If A lies on T which is a partial computable Ψ_i-splitting tree and $\Psi_i(A)$ is total then $A \leq_T \Psi_i(A)$.*

Proof. Suppose that A lies on T which is a partial computable Ψ_i-splitting tree and $\Psi_i(A)$ is total. Now we suppose that we have an oracle for $\Psi_i(A)$ and we show how one can compute A. We start at the base of T and work our way upwards.

We start with the knowledge that $T(\emptyset)\downarrow \subset A$. Since A lies on T, one of $T(0)$ and $T(1)$ must be an initial segment of A, so compute $T(0)=\tau_0$ and $T(1)=\tau_1$ – the fact that A lies on T means that these values must be defined. Since T is Ψ_i-splitting one of $\Psi_i(\tau_0)$ and $\Psi_i(\tau_1)$ must be incompatible with $\Psi_i(A)$. Determine which this is, and therefore which of τ_0 and τ_1 is an initial segment of A.

In the process just described, we started by knowing which string of level 0 in T is an initial segment of A (since there is only one), and we worked out which string of level 1 is an initial segment of A. Now we can proceed in exactly the same way working above this string in order to work out which string of level 2 in T is an initial segment of A, and so on. □

Lemma 5.2. *If A lies on T which is partial computable and Ψ_i-nonsplitting, then $\Psi_i(A)$ is computable if total.*

Proof. Suppose that A lies on T which is partial computable and also Ψ_i-nonsplitting, and suppose further that $\Psi_i(A)$ is total. In order to compute $\Psi_i(A;n)$ simply search until $\sigma \in T$ is found with $\Psi_i(\sigma;n)\downarrow$. Such a string σ must exist since A lies on T and $\Psi_i(A;n)\downarrow$. It must be the case that $\Psi_i(\sigma;n)=\Psi_i(A;n)$ since T is Ψ_i-nonsplitting. □

Lemmas 5.1 and 5.2 provide the key to ensuring minimality. We must ensure for each i that either:

1. A lies on a partial computable T which is Ψ_i-splitting. Then the conditions of Lemma 5.1 are satisfied, so that if $\Psi_i(A)$ is total then $A \leq_T \Psi_i(A)$,
   OR
2. A lies on a partial computable T which is Ψ_i-nonsplitting. Then the conditions of Lemma 5.2 are satisfied, so that if $\Psi_i(A)$ is total it is computable.

Recall that by $\sigma * \tau$ we mean the string which is σ concatenated with τ, i.e. the string τ' such that $|\tau'| = |\sigma| + |\tau|$ and such that $\tau'(n) = \sigma(n)$ for all $n < |\sigma|$ and $\tau'(n + |\sigma|) = \tau(n)$ for all $n < |\tau|$. We need a couple more simple definitions before we begin describing the actual construction.

Definition 5.6 *If T is a tree and $\tau = T(\sigma)$, then we define T_τ, the **subtree of** T **above** τ in the obvious way: for all σ_0, $T_\tau(\sigma_0) = T(\sigma * \sigma_0)$.*

Definition 5.7 *Given a partial computable tree T and τ in T, we define T_1 which is the **Ψ_i-splitting subtree of** T **above** τ by induction as follows. Set $T_1(\emptyset) = \tau$. Suppose $T_1(\sigma)$ is defined. Search (in some fixed algorithmic fashion) for two strings extending $T_1(\sigma)$ in T which are Ψ_i-splitting. For τ_0, τ_1 which are the first such pair of strings found, define $T_1(\sigma * 0) = \tau_0$, $T_1(\sigma * 1) = \tau_1$. In the case that no such strings are found, T_1 is undefined on all strings properly extending σ.*

The construction will proceed in stages. At each stage s we shall define a finite binary string α_s and also a perfect tree T_s, and α_s will be a string in T_s. Making these definitions at stage s amounts to agreeing to the restriction that A will extend α_s and A will lie on T_s. We begin by defining $\alpha_0 = \emptyset$ and $T_0 = \mathrm{id}$ (where id is the identity function $2^{<\omega} \to 2^{<\omega}$). So far, this amounts to no restriction at all: A will extend the empty string and will be an infinite binary string. At stage $s+1$, given α_s and T_s, we further restrict the choices for A in two ways. We define α_{s+1} to be a string extending α_s and we define T_{s+1} to be some perfect subtree of T_s. Of course we choose T_{s+1} in such a way that the restriction that A lies on T_{s+1} suffices to ensure satisfaction of $\mathscr{Q}_s$.

We now formally describe the construction.

Stage 0. Define $\alpha_0 = \emptyset$ and $T_0 = \mathrm{id}$.
Stage $s+1$. First we satisfy $\mathscr{P}_s$. Since T_s is total, there exists $\alpha \supset \alpha_s$ in this tree such that $\alpha(n)$ is different than $\Psi_s(n)$ for some n (where being defined counts as different than being undefined). Let α be the least such string.

Next we satisfy $\mathscr{Q}_s$. We ask the following question: does there exist τ in T_s extending α, such that no two strings in T_s extending τ are Ψ_s-splitting?

If NOT: Then the Ψ_s-splitting subtree of T_s above α is total, let us call it T. We can just define $\alpha_{s+1} = \alpha$ and $T_{s+1} = T$. Since A will lie on T_{s+1}, the conditions of Lemma 5.1 are satisfied, and so is $\mathscr{Q}_s$.

If SO: Then we can define $\alpha_{s+1} = \tau$ and we can define T_{s+1} to be the subtree of T_s above τ. Since T_{s+1} is Ψ_s-nonsplitting, and A will lie on this tree, the conditions of Lemma 5.2 are satisfied and so is $\mathscr{Q}_s$. □

The proof just described suffices to show that minimal degrees exist, but analyzing the proof further can tell us something about *where* they exist. What oracle do we require in order to run this construction? During the first part of each stage, where we act in order to satisfy $\mathscr{P}_s$, we need only to find the least n such that there are two strings extending α_s in T_s of length greater than n and which disagree on argument n – and then to decide whether $\Psi_s(\emptyset; n)\downarrow$. This last task requires only an oracle for $\emptyset'$.

During the second part of each stage we need to be able to answer the following question: does there exist τ in T_s extending α, such that no two strings in T_s extending τ are Ψ_s-splitting? In order to see that a $\emptyset''$-oracle suffices to answer this question, consider the following $\emptyset'$-oracle search procedure, which terminates iff the answer is yes. Starting with α and proceeding through all the strings $\tau \in T_s$ extending α (ordered by length and then from left to right), this search procedure asks whether there exist two strings in T_s extending τ which are Ψ_s-splitting, and terminates if the answer is no.

This proof, then, gives the existence of minimal degrees below $\mathbf{0}''$. Do there exist minimal degrees below $\mathbf{0}'$? In fact there do, but no proof like the one just described, which uses perfect splitting trees, can be used in order to construct one. In order to see this, we need to take a look at the hyperimmune-free degrees. The definition we give here is not the original, but is provably equivalent to it.

Definition 5.8 *A is of **hyperimmune-free** degree if, for every function $f \leq_T A$, there exists a computable function g which majorizes f, i.e. such that $g(n) > f(n)$ for all n.*

So if A is of hyperimmune-free degree then it is not an ability A has to compute quickly growing functions – for every function A computes there is a computable function which grows at least as quickly. It is not immediately obvious that any hyperimmune-free degrees other than $\mathbf{0}$ should exist but, in fact, an analysis of the proof of Theorem 5.1 will tell us that any minimal degree constructed in this way will automatically be of hyperimmune-free degree.

Suppose that $f \leq_T A$. Then $f = \Psi_i(A)$ for some i, and in fact this holds for some i such that, for any σ, $\Psi_i(\sigma;n)\downarrow \Longrightarrow \Psi_i(\sigma;n')\downarrow$ for all $n' < n$. If A lies on a perfect computable Ψ_i-splitting tree T, then it is easily seen by induction on n that $\Psi_i(\sigma;n)\downarrow$ for every σ of level $n+1$ in T. Since T is a total function we can compute the set of values $\Psi_i(\sigma;n)$ such that σ is of level $n+1$ in T and then we can define $g(n)$ to be greater than all these values. In this way we compute g which majorizes f.

A degree is hyperimmune iff it is not hyperimmune-free.

Definition 5.9 *When $n \leq |\sigma|$, $\sigma \upharpoonright n$ denotes the initial segment of σ of length n (and this definition is also extended in the natural way to the case of infinite strings).*

Theorem 5.2 *If $\boldsymbol{a} \leq \mathbf{0}'$ then there exists f of degree $\boldsymbol{a}$ such that any function majorizing f is of degree above $\boldsymbol{a}$. Thus, every non-zero degree below $\mathbf{0}'$ is hyperimmune.*

Proof. If A is of degree below $\mathbf{0}'$ then it has a computable approximation $\{A_s\}_{s\in\omega}$. For all n, let $f(n)$ be the least $s > n$ such that $A_s \upharpoonright n = A \upharpoonright n$. Clearly f is computable in A. Now suppose that g majorizes f. We show that in this case, A is computable in g. In order to compute $A \upharpoonright n$ given g search for $m > n$ such that $A_s \upharpoonright n = A_{g(m)} \upharpoonright n$ for all s such that $m \leq s \leq g(m)$. Such an m exists because the approximation to A eventually settles on all arguments $< n$. Since $A_s \upharpoonright m = A \upharpoonright m$ for some s with $m < s \leq g(m)$ and $A_s \upharpoonright n$ *is the same* for all such values of s, this common value must be $A \upharpoonright n$. □

So long as we use perfect trees in order to construct our set of minimal degree, the set we construct will be of hyperimmune-free degree, but all non-zero degrees below $\mathbf{0}'$ are hyperimmune. Therefore, in order to construct a minimal degree below $\mathbf{0}'$, some modification in our approach will be required. The solution is to use partial splitting trees.

Theorem 5.3 *[GS] There exists a minimal degree below $\mathbf{0}'$.*

Proof. The requirements are just as before, and other than the fact that we shall now use partial splitting trees the basic approach is the same as before. In order to satisfy each requirement $\mathscr{Q}_s$ we shall ensure that either A lies on a partial computable Ψ_s-splitting tree, or else A lies on some partial computable tree which is Ψ_s-nonsplitting. What we have to do is to replace the $\emptyset''$ question, "does there exist τ in T_s extending α, such that no two strings in T_s extending τ are Ψ_s-splitting?" with a question which can be decided using an oracle for $\emptyset'$.

When we asked this question in the proof of Theorem 5.1 what we were essentially asking was, "if we define T_{s+1} to be a Ψ_s-splitting tree, will there be dead-ends in this tree – will there be some point above which it is undefined?". The answer to this question allowed us to decide in one go how we should define T_{s+1}. Now we have to give up on the idea that this decision can be made in one step. We begin by assuming that T_{s+1} can be defined to be a splitting tree and that we will be able to construct A so as to lie on this tree. At each stage we ask the oracle for $\emptyset'$ whether there exists at least one more pair of strings in the splitting tree, above the initial segment of A that we have already defined. If so then we can continue with the idea that T_{s+1} can be built as a splitting tree. If not, then we can now begin to build T_{s+1} as a nonsplitting tree.

Thus we now have to *approximate* the nested sequence of splitting trees. We let T_k^s denote our guess as to how T_k should be defined at stage s. At any given stage $s+1$ we may change our guess as to T_k, and when we do so we have to abandon everything we previously believed about $T_{k'}$ for $k' > k$. Once we get to a stage, however, when we no longer change our mind about any $T_{k'}$ for $k' < k$, we shall only change our mind about T_k a maximum of one

further time (when and if we decide that it shouldn't be a splitting tree). Clearly this means we shall only change our mind as to how each tree should be defined a finite number of times.

We now formally describe the construction.

Stage 0. Define $\alpha_0 = \emptyset$ and $T_0^0 = \mathrm{id}$.
Stage $s+1$. We are given α_s and a finite sequence of nested trees

$$T_0^s \supseteq \cdots \supseteq T_k^s$$

for some $k \leq s$. Find the least k_0 such that either $k_0 = k+1$ or else there do not exist any strings in $T_{k_0}^s$ properly extending α_s. The rest of the required action for the stage is divided into two subcases.

If $k_0 \leq k$: Then define $T_{k_0}^{s+1}$ to be the subtree of $T_{k_0-1}^s$ above α_s. Define $T_{k_1}^{s+1} = T_{k_1}^s$ for all $k_1 < k_0$ and make $T_{k_1}^{s+1} \uparrow$ for all $k_1 > k_0$. Define α_{s+1} to be a proper extension of α_s in $T_{k_0}^{s+1}$ which is not an initial segment of $\Psi_s(\emptyset)$.

If $k_0 = k+1$: Define α_{s+1} to be a proper extension of α_s in T_k^s which is not an initial segment of $\Psi_s(\emptyset)$. Define $T_{k_1}^{s+1} = T_{k_1}^s$ for all $k_1 \leq k$, define T_{k+1}^{s+1} to be the Ψ_k-splitting subtree of T_k^s above α_{s+1}, and leave $T_{k_1}^{s+1}$ undefined for all $k_1 > k+1$.

This ends the formal description of the construction.

The verification. We only give a quick sketch. It is clear that each requirement $\mathscr{P}_k$ is satisfied. In order to show that each requirement $\mathscr{Q}_k$ is satisfied we must show that our approximation to each T_k eventually settles, either to some partial computable Ψ_k-splitting tree or to some partial computable Ψ_k-nonsplitting tree, and that A lies on this tree. This follows easily by induction. □

What we have given here is just a very brief introduction to the techniques of minimal degree construction. In the 60s and 70s, as computability theorists looked to answer various global questions concerning the theory of the Turing degrees, such as the decidability or otherwise of various fragments of the theory, the path originally taken in order to achieve this was through a detailed analysis of the initial segments of the structure (where an initial segment is a downward closed set of degrees). The techniques we have introduced here were greatly extended by Lachlan, Thomason, Lerman and others, through the use of *lattice tables* and other such methods, in order to give us a great deal of information about what the initial segments of the structure look like. It was eventually established that the isomorphism types of countable ideals of the Turing degrees are exactly the isomorphism types of countable upper semi-lattices with least element. For a thorough account we refer the reader to Lerman's book [ML]. From the latter result it can be proved that the two quantifier theory of the Turing degrees is decidable, and that the three quantifier theory is undecidable (actually all that is required is that every finite lattice can be embedded as an initial segment). In fact, Simpson's original proof [SS2] in which he used coding methods in order to show that the theory of the Turing degrees is computably isomorphic to true second order arithmetic, also relied upon initial segment techniques. Later Slaman and Woodin [SW], however, gave a simpler (and more generally applicable) proof, which does not use techniques of this kind.

6. Some order theoretic properties of $0'$

In the following sections we shall define the local and generalized **jump classes** and we shall then be looking to establish which order theoretic properties are satisfied by the degrees in each of these classes – the aim being that we might eventually be able to establish some fairly natural order theoretic properties which suffice to **define** the degrees in each class. As we discussed before, by a **natural** order theoretic property we mean, roughly speaking, a property which can be described by a formula in the first order language for the Turing degrees which has $\leq$ as the only non-logical symbol, which is not too long and doesn't have too many alternations of quantifiers. A simple example would be the **cupping property**; a degree $\boldsymbol{a}$ satisfies the cupping property if $\forall \boldsymbol{b} > \boldsymbol{a}$ there exists $\boldsymbol{c} < \boldsymbol{b}$ with $\boldsymbol{a} \vee \boldsymbol{c} = \boldsymbol{b}$.

By way of preparation, we consider first some simple order theoretic properties satisfied by $\mathbf{0}'$. This will allow us to introduce some standard techniques.

Theorem 6.1 $\mathbf{0}'$ *satisfies the cupping property.*

Proof. We give a proof using perfect trees, which will be easily modified later on in order to give the result for larger classes of degrees.

Definition 6.1 *We say that* $\boldsymbol{a}$ *is* ***perfectly cone avoiding*** *if it computes a perfect tree* T *such that no path through* T *is of degree above* $\boldsymbol{a}$*.*

If $\boldsymbol{a}$ is perfectly cone avoiding then it is easily observed that it satisfies the cupping property as follows. Suppose that the perfect tree T is of degree below $\boldsymbol{a}$ and that no path through T is of degree above $\boldsymbol{a}$. Given B of degree $\boldsymbol{b} > \boldsymbol{a}$ let $C = T(B)$, i.e. let C be the infinite string which extends $T(\sigma)$ for all $\sigma \subset B$. Then C is a path through T and so is of degree $\boldsymbol{c} < \boldsymbol{b}$. Given an oracle for T and an oracle for C we can determine B from the fact that $T(B) = C$. Thus $\boldsymbol{a} \vee \boldsymbol{c} = \boldsymbol{b}$.

So in order to show that $\mathbf{0}'$ satisfies the cupping property, it suffices to show that it is perfectly cone avoiding. We can construct the required tree T using an oracle for $\emptyset'$ as follows. Suppose $T(\sigma)$ is already defined and let $|\sigma| = n$. If there exist two strings extending $T(\sigma)$ which are Ψ_n-splitting then let $\tau \supset T(\sigma)$ be such that $\Psi_n(\tau)$ is incompatible with $\emptyset'$ and let $T(\sigma * 0)$ and $T(\sigma * 1)$ be incompatible extensions of τ. If there do not exist two such strings then $\Psi_n(A)$ is either partial or computable for all $A \supset T(\sigma)$, so we can just define $T(\sigma * 0)$ and $T(\sigma * 1)$ be any incompatible extensions of $T(\sigma)$. □

It is easy to see that the degrees which satisfy the cupping property are upward closed. The join property, however, is not so simple in this respect.

Definition 6.2 *A degree* $\boldsymbol{a}$ *satisfies the* ***join*** *property if, for all non-zero* $\boldsymbol{b} < \boldsymbol{a}$ *there exists* $\boldsymbol{c} < \boldsymbol{a}$ *with* $\boldsymbol{b} \vee \boldsymbol{c} = \boldsymbol{a}$*.*

Theorem 6.2 ([PR]) $\mathbf{0}'$ *satisfies the join property.*

Proof. We suppose that we are given B of non-zero degree below $\mathbf{0}'$. In fact, we can suppose not only that B is non-computable, but also that it is not c.e. (in any case, letting $\bar{B}$ be the complement of B, $B \oplus \bar{B}$ is not c.e. if B is non-computable). We use an oracle for $\emptyset'$ to construct A such that $\emptyset' \leq_T A \oplus B$ and:

$\mathscr{Q}_s$: $\Psi_s(A) \neq \emptyset'$.

The construction will be a finite extension argument. For all n let σ_n be the sequence of n 0s followed by a 1 (the point of this sequence of strings just being that they are pairwise incompatible). The basic idea behind the construction is that it should be possible to *retrace* each stage of the construction using an oracle for $A \oplus B$. At each stage we code another bit of $\emptyset'$ into A, so that retracing the construction means being able to compute $\emptyset'$. $A = \bigcup_s \alpha_s$ is constructed in stages as follows.

Stage 0. Define $\alpha_0 = \emptyset$.
Stage $s+1$. Let n be the least such that either:

Case 1. $n \in B$ and there do not exist two extensions of $\alpha_s * \sigma_n$ which are Ψ_s-splitting.
Case 2. $n \notin B$ and there exist two extensions of $\alpha_s * \sigma_n$ which are Ψ_s-splitting.

Such an n must exist, since otherwise $n \in B$ iff there exist two extensions of $\alpha_s * \sigma_n$ which are Ψ_s-splitting – in which case B would be c.e., contrary to assumption.

If case 1 applies then $\mathscr{Q}_s$ will be satisfied so long as we insist that $A \supset \alpha_s * \sigma_n$, since then $\Psi_s(A)$ is either partial or computable. So in this case we define $\alpha_{s+1} = \alpha_s * \sigma_n * \emptyset'(s)$ (so we code another bit of $\emptyset'$ into A with the last bit of α_{s+1}).

If case 2 applies then consider the first Ψ_s-splitting extending $\alpha_s * \sigma_n$ which is found by some fixed computable search procedure, choose α_s^* from this splitting such that $\Psi_s(\alpha_s^*)$ is incompatible with $\emptyset'$ and define $\alpha_{s+1} = \alpha_s^* * \emptyset'(s)$.

This ends the description of the construction.

The verification. Since it is clear that all the $\mathscr{Q}_s$ requirements are satisfied, it remains only to show that an oracle for $A \oplus B$ can compute the sequence $\{\alpha_s\}_{s\in\omega}$. Since the last bit of α_{s+1} is $\emptyset'(s)$, computing this sequence means being able to compute $\emptyset'$. So suppose that, using an oracle for $A \oplus B$ we have already been able to decide α_s. Then there exists a unique n such that $\alpha_s * \sigma_n \subset A$. Given n, we can then consult the oracle for B to decide whether $n \in B$. This tells us whether case 1 or case 2 applied at stage $s+1$, and this suffices for us to decide α_{s+1}. □

Definition 6.3 *A degree $\boldsymbol{a}$ satisfies the **meet** property if, for all $\boldsymbol{b} < \boldsymbol{a}$, there exists non-zero $\boldsymbol{c} < \boldsymbol{a}$ with $\boldsymbol{b} \wedge \boldsymbol{c} = \boldsymbol{0}$.*

The following proof is amongst the hardest that we shall give in this course. It is not necessary to understand this proof in order to follow the rest of the course, and so readers who feel inclined to do so may certainly omit it. In fact, the result can be proved more simply, but the advantage of making the effort to understand the proof given here is that it is easily modified in order to prove the *complementation theorem*: if $\boldsymbol{0} < \boldsymbol{a} < \boldsymbol{0}'$ then there exists $\boldsymbol{b} < \boldsymbol{0}'$ such that $\boldsymbol{a} \vee \boldsymbol{b} = \boldsymbol{0}'$ and $\boldsymbol{a} \wedge \boldsymbol{b} = \boldsymbol{0}$. It also serves as a good example of various techniques.

Theorem 6.3 ([JS]) *$\boldsymbol{0}'$ satisfies the meet property.*

Proof. We suppose that we are given $B <_T \emptyset'$ and we use an oracle for $\emptyset'$ to construct A which satisfies all requirements:

$\mathscr{P}_e$: $A \neq \Psi_e(\emptyset)$.
$\mathscr{Q}_e$: If $\Psi_e(A)$ is total and $\Psi_e(A) = \Psi_e(B) = C$ then C is computable.

Definition 6.4 *We say that a function f dominates a function g if for all but finitely many $n \in \omega$, $f(n) \geq g(n)$.*

Let's begin by considering a simple strategy that we might employ in order to satisfy an individual requirement $\mathscr{Q}_e$. Given the oracle for $\emptyset'$ we intend to construct A by a finite extension argument, so suppose that at stage $s+1$ of the construction the initial part of A we have already defined is α_s, and that now we wish to satisfy requirement $\mathscr{Q}_e$. Using the oracle for $\emptyset'$ we can ask whether there exist two strings extending α_s which are Ψ_e-splitting.
If not: then, since A will extend α_s, $\Psi_e(A)$ will either be partial or computable.
If so: then we can find τ_0, τ_1 extending α_s and m for which $\Psi_e(\tau_0; m)\downarrow \neq \Psi_e(\tau_1; m)\downarrow$. Now suppose that we knew $\Psi_e(B; m)\downarrow$ – then we could choose i such that $\Psi_e(\tau_i; m) \neq \Psi_e(B; m)$ and thus satisfy requirement $\mathscr{Q}_e$ in this case also by defining $\alpha_{s+1} = \tau_i$.

The obvious problem with this approach is that *we do not know whether* $\Psi_e(B; m)\downarrow$. In order to decide this for arbitrary e and m would require an oracle for B', which we do not have. The basic idea to overcome this obstacle is to use Theorem 5.2. We suppose we are given f of degree $\boldsymbol{0}'$ which is not dominated by any function of degree strictly below $\boldsymbol{0}'$ and so, in particular, is not dominated by any function computable in B.

Using this function f we now consider a slightly more sophisticated approach. At stage $s+1$, given α_s, we perform an iteration as follows. Initially we define $\tau_0 = \alpha_s$. At the n^{th} step in the iteration ($n \geq 1$) we are given τ_{n-1}. Check to see whether there exist two strings extending τ_{n-1} which are Ψ_e-splitting. If not then we say that $(\dagger_n)$ holds, and in this case we may define $\alpha_{s+1} = \tau_{n-1}$ before terminating the iteration. Otherwise let τ, τ' be two such strings. Let m_e^n be such that $\Psi_e(\tau; m_e^n)\downarrow \neq \Psi_e(\tau'; m_e^n)\downarrow$. Then check to see whether $\Psi_e(B; m_e^n)\downarrow$ in less than $f(n)$ steps. If so then we say that $(\ddagger_n)$ holds and in this case we can choose $\tau^* \in \{\tau, \tau'\}$ such that $\Psi_e(\tau^*; m_e^n) \neq \Psi_e(B; m_e^n)$, define $\alpha_{s+1} = \tau^*$ and terminate the iteration. If $(\ddagger_n)$ does not hold, then let τ^* be the leftmost of τ, τ', define $\tau_n = \tau^*$ and perform the next step in the iteration.

Now if $\Psi_e(B)$ is total we need to be able to argue that this iteration will come to an end, i.e. for some $n \geq 1$, either $(\dagger_n)$ or $(\ddagger_n)$ holds. So suppose otherwise and consider the function:

$$g(n) = \mu s.(\Psi_e(B;m_e^n)[s]\downarrow),$$

(where for any predicate $R(s)$, $\mu s.R(s)$ denotes the least s such that $R(s)$ holds and is undefined if $R(s)$ does not hold for any s). Since it is the case for all $n \geq 1$ that $(\ddagger_n)$ does not hold, g dominates f. This gives us the required contradiction because g is computable in B.

We are not through the woods yet though. The argument above sufficed to show that, *when* $\Psi_e(B)$ *is total* the iteration must terminate. When $\Psi_e(B)$ is not total, however, we have a problem because then the iteration may not terminate. To attempt to perform this iteration at a given stage might require an infinite number of steps, meaning that we never reach the next stage of the construction. The obvious way to deal with this is to *dovetail* the action required for each requirement. Rather than attempting to perform the entire iteration for a single $\mathscr{Q}_e$ at any given stage, we perform one step of the iteration for $\mathscr{Q}_0$ at stage 1, then another step of the iteration for $\mathscr{Q}_0$ and also one step of the iteration for $\mathscr{Q}_1$ at stage 2. Then at stage 3, we perform one step of each of the iterations for $\mathscr{Q}_0$, $\mathscr{Q}_1, \mathscr{Q}_2$, and so on.

This approach now brings with it a new problem. In order that g should be computable in B it was necessary that the sequence m_e^n should be computable (or at least computable in B). Previously this was the case because, at each step in the iteration, when we found that neither of $(\dagger_n)$ and $(\ddagger_n)$ held, τ_{n+1} was always defined to be the leftmost of τ and τ'. It was because we always went to the left that the sequence m_e^n was computable – had we gone sometimes to the left and sometimes to the right in a manner computable only in an oracle for $\emptyset'$ then we would only have been able to deduce that the sequence m_e^n (and so also the function g) was computable in $\emptyset'$. Now that we dovetail the action required for various requirements, this is precisely what will happen – sometimes we will find the chance to diagonalize for lower priority requirements and this may require us fixing either the leftmost or the rightmost string in some splitting as an initial segment of A.

What we need is a function $h \leq_T B$ which dominates g and which does not depend upon the values m_e^n in such a sensitive way, and we therefore consider the function h defined as follows:

$$h(1) = g(1)$$

$$h(n+1) = \mu s.(\Psi_e(B)[s] \upharpoonright h(n)\downarrow).$$

If we had that, for all $n \geq 1$, $m_e^{n+1} < g(n)$ then we could prove that for all $n \geq 1$, $g(n) \leq h(n)$ by a simple induction as follows. By definition $g(1) \leq h(1)$. Suppose that we know $g(n) \leq h(n)$. Then $m_e^{n+1} < h(n)$ and thus:

$$h(n+1) = \mu s.(\Psi_e(B)[s] \upharpoonright h(n)\downarrow) \geq g(n+1) = \mu s.(\Psi_e(B;m_e^{n+1})[s]\downarrow).$$

Now in order to ensure that $m_e^{n+1} < g(n)$ for all $n \geq 1$ we need only make a small change to the construction. Instead of checking to see whether the computation $\Psi_e(B;m_e^n)$ converges in less than $f(n)$ steps, we wait until we have defined m_e^{n+1} and then check to see whether the computation converges in at most $\max\{f(n), m_e^{n+1}\}$ steps. If so then we may define α_{s+1} so as to successfully diagonalize once and for all. Otherwise $m_e^{n+1} < g(n)$ as we required.

Now we must put all this together. First of all we establish some further notation and terminology. The splittings that are found during the iteration that we perform for $\mathscr{Q}_e$ will all be enumerated into a set T_e. So initially this set is empty, and then we enumerate strings into it during the course of the construction. Any strings that are enumerated into any T_e will also be enumerated into the set T – so T collects all of these splittings together into one set.

There are two ways in which we might get to irreversibly satisfy a requirement $\mathscr{Q}_e$. When we find a string τ above which there are no Ψ_e-splittings, we shall say that the requirement is γ-satisfiable via τ. When we find τ

with which we can diagonalize by forcing $\Psi_e(\tau;m)\downarrow \neq \Psi_e(B;m)\downarrow$ for some m, we shall say that the requirement is β-satisfiable via τ.

As described above, at stage 1 we perform one step in the iteration for $\mathscr{Q}_0$. Suppose that a splitting is found and that these strings, τ_0 and τ_1 say, are enumerated into T_0 and T. For the reasons described above, we do not immediately decide which of these strings will be an initial segment of A. At stage 2 we now look to perform a step of the iteration for $\mathscr{Q}_0$ and also a step in the iteration for $\mathscr{Q}_1$. We do this, however, *in reverse order*, i.e. we work for $\mathscr{Q}_1$ first and *then* $\mathscr{Q}_0$. The reason for this will subsequently become apparent. So first of all, we work for $\mathscr{Q}_1$. We do not yet know, however, which of the two strings already enumerated into T will be an initial segment of A, so we must work above both of these strings. We ask whether there exists a Ψ_1-splitting above τ_0 and we also ask whether there exists a Ψ_1-splitting above τ_1. Any splittings that exist are enumerated into T and T_1. If there doesn't exist a splitting above some τ_i then we record that $\mathscr{Q}_1$ is γ-satisfiable via this string. Rather than a single argument m_1^1 as in the iteration described before, we now have an argument corresponding to each splitting pair enumerated into T_1.

Having done this for $\mathscr{Q}_1$, we now work for $\mathscr{Q}_0$, and once again we work above every string enumerated into T. Again, rather than a single argument m_0^2 we now have an argument corresponding to each pair of strings enumerated into T_0. We use the maximum of these values and $f(1)$ as we test to see whether $\mathscr{Q}_0$ is β-satisfiable via either τ_0 or τ_1.

Then at the end of stage 2, we consider the highest priority requirement which is either β-satisfiable or γ-satisfiable via some string, and we declare that this string should be an initial segment of A. It is at this point that we can see why we considered the requirements in reverse order. Suppose that $\mathscr{Q}_0$ is not γ or β-satisfiable via any string at the end of stage 2, but that $\mathscr{Q}_1$ is γ-satisfiable via either τ_0 or τ_1. When we declare that this string, τ_i say, should be an initial segment of A at the end of stage 2, we have not lost our opportunity to find a string via which $\mathscr{Q}_0$ is β-satisfiable at the *next* stage – there are splittings which we have enumerated into T_0 extending τ_i at stage 2, and this is true only because we treated the requirements in reverse order. Thus we avoid the action we take for $\mathscr{Q}_1$ injuring our attempt to satisfy $\mathscr{Q}_0$.

The construction.
Stage $s=0$. Define $\alpha_0=\emptyset$. Initially T and all the T_e are empty.
Stage $s>0$.
Step 1. For each $e<s$, from greatest to least, proceed as follows in turn. For each leaf σ of T that extends α_{s-1} (if T is empty then it is convenient to consider $\emptyset$ a leaf of T) check to see whether there exist two strings extending σ which are Ψ_e-splitting. If not then record that $\mathscr{Q}_e$ is 'γ-satisfiable' at stage s via σ, unless the requirement $\mathscr{Q}_e$ has already been declared satisfied. Otherwise let τ and τ' be two such strings, enumerate them into T and T_e, and find m such that $\Psi_e(\tau;m)\downarrow \neq \Psi_e(\tau';m)\downarrow$. Since τ and τ' are both enumerated into T at the same stage and both extend the same string τ which was (prior to their enumeration) a leaf of T, we shall call these two strings a 'pair'. We shall call m the 'argument' corresponding to this pair of strings. Let $\tau_1,...,\tau_{2k}$ be all of those strings that we have enumerated into T_e at stage s, if there are any such, and let $m_1,..,m_k$ be those arguments corresponding to each pair of strings in this set. If this set is non-empty proceed as follows (and otherwise do nothing). Define $m(e,s)=\max\{f(s-1),m_1,..,m_k\}$. Check to see whether there exist any strings which we enumerated into T_e at stage $s-1$ and which properly extend α_{s-1}. If so then call these strings $\sigma_1,..,\sigma_{2k'}$ and let $n_1,..,n_{k'}$ be the arguments corresponding to each pair of strings in this set. Check to see whether there exists $i\leq k'$ such that $\Psi_e(B;n_i)$ converges in at most $m(e,s)$ steps. If so then choose σ_j such that $\Psi_e(B;n_i)\neq\Psi_e(\sigma_j;n_i)\downarrow$ and record that $\mathscr{Q}_e$ is 'β-satisfiable' at stage s via σ_j, unless the requirement $\mathscr{Q}_e$ has already been declared satisfied.

Step 2. If there does not exist $e<s$ which is either γ-satisfiable or β-satisfiable at stage s then let τ be the leftmost of all the strings extending α_{s-1} which were leaves of T at the end of stage $s-1$, and define $\alpha_s=\tau$ (if T was empty prior to stage s then define $\alpha_s=\emptyset$). Otherwise let e be the least such. We act to satisfy requirement $\mathscr{Q}_e$. If $\mathscr{Q}_e$ is γ-satisfiable via some string τ then choose such a string and define $\alpha_s=\tau$. Otherwise if $\mathscr{Q}_e$ is β-satisfiable via some string τ, choose such a string and define $\alpha_s=\tau$. Requirement $\mathscr{Q}_e$ is then declared to be satisfied.

Step 3. For each leaf τ of T enumerate into T a string τ' extending τ which diagonalizes against the s^{th} computable function.

Verification (Sketch). It is clear that for every $e \in \omega$ the requirement $\mathscr{P}_e$ is satisfied so we are left to show that all requirements of the form $\mathscr{Q}_e$ are satisfied. If we act to satisfy any requirement $\mathscr{Q}_e$ at some stage of the construction then obviously this requirement is (irreversibly) satisfied. So suppose that there is no stage of the construction at which we act in order to satisfy the requirement $\mathscr{Q}_e$ and that $\Psi_e(B)$ is total. Then we may take a stage s large enough such that at no stage $s' \geq s$ do we act in order to satisfy any requirement $\mathscr{Q}_{e'}$ for $e' \leq e$. At no stage $s' \geq s$ is the requirement $\mathscr{Q}_e$ γ-satisfiable. At every stage $s' > s$ we enumerate strings into T_e and it is the case that there exist strings which we enumerated into T_e at stage $s'-1$ which extend $\alpha_{s'-1}$. Consider the function g' defined as follows. If $s' < s$ then $g'(s') = 0$. Given any $s' \geq s$ let $\tau_1, .., \tau_{2k}$ be those strings which we enumerated into T_e at stage s' and which extend $\alpha_{s'}$. Let $m_1, .., m_k$ be the arguments corresponding to each pair of strings in this set. If $s' \geq s$ then $g'(s') = \min\{\mu s''.(\Psi_e(B; m_i)[s''] \downarrow) : 1 \leq i \leq k\}$. Then g' dominates f. We may also show by almost precisely the same inductive argument as before that g' is dominated by the function h' defined as follows. If $s' \leq s$ then $h'(s') = g'(s')$. If $s' > s$ then $h'(s') = \mu s''.(\Psi_e(B)[s''] \upharpoonright h'(s'-1) \downarrow)$. This gives us the contradiction required since $h' \leq_T B$. □

7. The jump hierarchy

We introduce the jump hierarchy first for the degrees below $\mathbf{0}'$. In this context the jump hierarchy provides a way of formalizing the notion of being close to $\mathbf{0}$ or close to $\mathbf{0}'$. The idea behind this hierarchy begins with the observation that there are degrees $\boldsymbol{a} > \mathbf{0}$ such that $\boldsymbol{a}' = \mathbf{0}'$.

Definition 7.1 *A degree $\boldsymbol{a}$ is **low** if $\boldsymbol{a}' = \mathbf{0}'$.*

Theorem 7.1 *There exist non-zero low degrees. In fact there exist c.e. degrees of this kind.*

Proof. To prove that there exist non-zero low degrees can be done very easily using a finite extension $\emptyset'$-oracle construction. We prove the stronger result that there exist non-zero c.e. degrees which are low, because it gives us the opportunity to describe a *finite injury* construction.

As is normally the case when working with the c.e. degrees, the construction we describe will be a *full approximation* construction, which means we shall not make use of any oracle but instead will describe a construction which can be effectively carried out. This is not surprising since the only general method we have for ensuring that a constructed set is c.e. is to describe an effective procedure for enumerating it.

First, let's consider the non-computability requirements. In order to ensure that A is non-computable, it suffices to ensure that it has infinite complement and that:

$$\mathscr{P}_e : \ W_e \text{ infinite } \implies W_e \cap A \neq \emptyset.$$

These requirements suffice because A is computable iff A and $\bar{A}$ are both c.e. – these requirements ensure that the complement of A is not equal to W_e for any e.

Next let's consider how to make the degree of A low. In order to do this, we act in order to satisfy the following requirements:

$$\mathscr{N}_e : (\exists^\infty s)[\Psi_e(A_s; e)[s] \downarrow] \implies \Psi_e(A; e) \downarrow,$$

where $(\exists^{\infty}s)Q(s)$ denotes "there exist infinitely many s such that $Q(s)$", A_s consists of the numbers enumerated into A by the end of stage s and $A = \bigcup_s A_s$. In order to see that satisfaction of the $\mathscr{N}_e$ requirements suffices to ensure that A is of low degree, consider the function g defined as follows; $g(e,s) = 1$ if $\Psi_e(A_s;e)[s]\downarrow$ and $g(e,s) = 0$ otherwise. Let $g^*(e) = \lim_s g(e,s)$ – satisfaction of $\mathscr{N}_e$ means that this limit exists. Then g^* is the characteristic function of A' and, since g is computable, g^* is computable given an oracle for $\emptyset'$.

Definition 7.2 *The **use** function is defined as follows; $u(B;e,x,s)$ is 1 plus the maximum element of the oracle used in the computation $\Psi_e(B;x)[s]$ if this computation converges, and is 0 otherwise.*

In order to satisfy the requirements $\mathscr{N}_e$ we consider a *restraint* function for each e;

$$r_e(s) = u(A_s;e,e,s).$$

We shall say that the restraint function r_e is **injured** at stage $s+1$ of the construction if $n < r_e(s)$ is enumerated into A at this stage. The crucial point about the restraint function is just this: if there exists a stage of the construction after which r_e is not injured then $\mathscr{N}_e$ is satisfied and $\lim_s r_e(s)$ is defined. In order to see this, suppose that r_e is not injured at any stage $\geq s_0$. If there doesn't exist any stage $t \geq s_0$ at which $\Psi_e(A_t;e)[t]\downarrow$ then $\mathscr{N}_e$ is satisfied, and $\lim_s r_e(s) = 0$. Otherwise let t be the least such. Since we do not enumerate any numbers into A less than $u(A_t;e,e,t)$ after stage t, this computation is preserved so that $\Psi_e(A;e)\downarrow$ and $r_e(s) = r_e(t)$ for all $s \geq t$.

In order to satisfy all of the requirements, then, we consider them given a priority ranking $\mathscr{N}_0, \mathscr{P}_0, \mathscr{N}_1, \mathscr{P}_1, \ldots$. So $\mathscr{N}_0$ is the highest priority requirement, then $\mathscr{P}_0$ is the requirement of next highest priority, and so on. Then we agree that any requirement $\mathscr{P}_e$ is not allowed to injure any requirement $\mathscr{N}_i$ of higher priority, i.e. $\mathscr{P}_e$ is only allowed to enumerate a number into A if this number is greater than the present values of all the restraint functions corresponding to requirements of higher priority. Once a requirement $\mathscr{P}_e$ enumerates a number into A it is irreversibly satisfied, so each requirement $\mathscr{P}_e$ needs only to enumerate at most one number into A.

For each requirement $\mathscr{N}_i$ there are only a finite number of requirements $\mathscr{P}_e$ which are of higher priority, and this immediately means that there will be a stage of the construction after which $\mathscr{N}_i$ is not injured. Therefore $\mathscr{N}_i$ is satisfied and $\lim_s r_i(s)$ is defined. This in turn means that we can satisfy each requirement $\mathscr{P}_e$ – if W_e is infinite then it will have a member greater than the limit values of all restraint functions of higher priority, and we can enumerate this number into A in order to satisfy the requirement. We now formally describe the construction.

Stage 0. Let $A_0 = \emptyset$.
Stage $s+1$. We are given A_s. Find the least $i \leq s$ (if any) such that:

$$W_{i,s} \cap A_s = \emptyset$$

and

$$\exists x[x \in W_{i,s} \ \& \ x > 2i \ \& \ (\forall e \leq i)[r_e(s) < x]],$$

where $W_{i,s}$ is the domain of $\Psi_i(\emptyset)[s]$. If such an i exists, then choose the least x satisfying the second clause above and enumerate this x into A, i.e. define $A_{s+1} = A_s \cup \{x\}$. If no such i exists then do nothing, so $A_{s+1} = A_s$.

This ends the construction.

The verification. That the complement of A is infinite follows since each requirement $\mathscr{P}_e$ enumerates at most one x into A, and if it does enumerate some x into A then $x > 2e$. That every requirement is satisfied follows by induction on the priority ranking according to the arguments already described above. □

We define the jump hierarchy first for the degrees below $\mathbf{0}'$.

Definition 7.3 *We define $\boldsymbol{a}^{(n)}$ inductively as follows; $\boldsymbol{a}^{(0)} = \boldsymbol{a}$ and $\boldsymbol{a}^{(n+1)}$ is the jump of $\boldsymbol{a}^{(n)}$.*
A degree $\boldsymbol{a}$ is $\boldsymbol{low}_n$ if $\boldsymbol{a} < \mathbf{0}'$ and $\boldsymbol{a}^{(n)} = \mathbf{0}^{(n)}$. A degree $\boldsymbol{a}$ is $\boldsymbol{high}_n$ if $\boldsymbol{a} \leq \mathbf{0}'$ and $\boldsymbol{a}^{(n)} = \mathbf{0}^{(n+1)}$.

In other words, a degree below $\mathbf{0}'$ is low_n if its nth jump is as low as it possibly could be – the same as the nth jump of $\mathbf{0}$. On the other hand, a degree is high_n if its nth jump is as high as it possibly could be – the same as the nth jump of $\mathbf{0}'$. A crucial point here is that when $\boldsymbol{a} \leq \boldsymbol{b}$ we always have $\boldsymbol{a}' \leq \boldsymbol{b}'$. This means that if $\boldsymbol{a} \leq \boldsymbol{b}$ and $\boldsymbol{b}$ is low_n then $\boldsymbol{a}$ is also low_n. If $\boldsymbol{a} \leq \boldsymbol{b}$ and $\boldsymbol{a}$ is high_n then $\boldsymbol{b}$ must be high_n also.

The next question which arises is whether these jump classes are all distinct. Is it the case that for every n, there exist degrees which are low_{n+1} but not low_n, and degrees which are high_{n+1} but not high_n? How about when we restrict to the c.e. degrees? In order to answer this question we consider **hop** operators, which are of considerable interest in their own right.

Definition 7.4 *W_e^A is the domain of $\Psi_e(A)$.*

Definition 7.5 *For $e \in \omega$ the eth hop operator J_e is defined as follows: for all $Y \subseteq \omega$, $J_e(Y) = Y \oplus W_e^Y$.*

In the definition above we take the $\oplus$ with Y simply in order to ensure that the degree of $J_e(Y)$ is above the degree of Y.

Definition 7.6 *We say that $A \equiv_T B$ **via** $\langle k_0, k_1 \rangle$ if $A = \Psi_{k_0}(B)$ and $B = \Psi_{k_1}(A)$.*

Theorem 7.2 ([Jsh]) *For every e there exists a c.e. set A such that $J_e(A) \equiv_T \emptyset'$. Moreover, A can be found uniformly in e and this result can be relativized to any Y, i.e. there exist computable functions f and g such that for all e, Y;*

$$J_e(J_{f(e)}(Y)) \equiv_T Y' \text{ via } g(e).$$

Proof. The basic techniques are somewhat similar to the proof of Theorem 7.1. First of all, we must make sure that $W_e^A \leq_T \emptyset'$. In order to achieve this we act in order to satisfy the requirements:

$$\mathscr{N}_x : \ (\exists^\infty s)[x \in W_{e,s}^{A_s}] \implies x \in W_e^A,$$

where $W_{e,s}^{A_s}$ is the domain of $\Psi_e(A_s)[s]$. That these requirements suffice to ensure $W_e^A \leq_T \emptyset'$ can be argued in the same way that we argued that the requirements for the proof of Theorem 7.1 sufficed to show $A' \leq_T \emptyset'$. The way in which we look to satisfy these requirements is also almost identical. We define a restraint function r_x for each x:

$$r_x(s) = u(A_s; e, x, s)$$

and we require that, for each x, there exists a stage of the construction after which the restraint function is not *injured*, i.e. after which we do not enumerate any numbers into A less than the present value of the restraint function for x. If we do this, then $\mathscr{N}_x$ will be satisfied and the value $r_x(s)$ will reach a limit.

Definition 7.7 *We let $A^{[i]}$ denote the ith column of A, i.e. the set of all numbers in A of the form $\langle i,j\rangle$. Given finite $F \subset \omega$, let $F = \{a_0 < \cdots < a_k\}$, and for each $i \leq k$ let X_i be $A^{[a_i]}$. We define $A^{[F]} = \bigcup_{i=0}^{k} X_i$.*

Our second task is to code $\emptyset'$ into $A \oplus W_e^A$. In order to do this, we agree first of all, that we shall put a number into the xth column of A iff $x \in \emptyset'$, i.e. we shall ensure that $A \cap \omega^{[x]}$ is non-empty iff $x \in \emptyset'$. On its own this would achieve very little, we would have only that $\emptyset'$ is c.e. given an oracle for A and this would be true anyway! If, however, we can ensure that there exists a function $h \leq_T A \oplus W_e^A$ such that, whenever there exists a number in the xth column of A, there exists one less than $h(x)$, then we shall have that $\emptyset' \leq_T A \oplus W_e^A$ as required. For each x, then, we have the requirement:

$$\mathscr{P}_x : \ x \in \emptyset' \iff (\exists y \leq h(x))[y \in \omega^{[x]} \cap A].$$

The pleasing observation now is that, in fact, these requirements are easily satisfied. What is it, after all, which is limiting how small are the numbers that we can enumerate into the xth column? This is just the restraints associated with the requirements $\mathscr{N}_{x'}$. If we prioritize the requirements $\mathscr{N}_0, \mathscr{P}_0, \mathscr{N}_1, \mathscr{P}_1, ...$ then there will only be a finite number of requirements $\mathscr{N}_{x'}$ of higher priority than $\mathscr{P}_x$. Given an oracle for W_e^A we know which of these x' are in W_e^A, and then we can use the oracle for A to compute the limit values of each corresponding $r_{x'}$. This means that, using the oracle for $A \oplus W_e^A$ we can find a position in the xth column after which we will always be allowed to enumerate numbers without injuring requirements of higher priority.

The precise instructions for the construction are as follows.

Stage 0. Let $A_0 = \emptyset$. For all x we define $h^*(x,0) = \langle x,0\rangle$.
Stage $s+1$. For each x we define:

$$h^*(x,s) = (\mu y)[y \in \omega^{[x]} \ \& \ h^*(x-1,s) < y \ \& \ h^*(x,s-1) \leq y$$

$$\& \ (\forall j \leq x)[r_j(s) < y]].$$

For each $x \in \emptyset'_{s+1} - \emptyset'_s$ enumerate $h^*(x,s)$ into A_{s+1}.

This ends the description of the construction.

We define $h(x) = \lim_s h^*(x,s)$.

We leave it to the reader to verify that h is computable in $A \oplus W_e^A$, and that the proof relativizes and remains uniform. □

Theorem 7.3 ([GS2]) *For every n, there exist c.e. degrees which are low_{n+1} but not low_n, and degrees which are $high_{n+1}$ but not $high_n$.*

Proof. First note that Theorem 7.1 relativizes to arbitrary Y. There exists an index e_1, in other words, such that $Y <_T J_{e_1}(Y)$ and $Y' \equiv_T (J_{e_1}(Y))'$ for all Y. So $J_{e_1}(Y)$ is low_1 relative to Y but not low_0 relative to Y. Applying Theorem 7.2 to e_1 produces an index j_1 such that $J_{j_1}(Y)$ is high_1 relative to Y but not high_0 relative to Y. Now, applying Theorem 7.2 to j_1, produces e_2 such that $J_{e_2}(Y)$ is low_2 relative to Y but not low_1 relative to Y. Continue this pattern using the fact that Y' is low_n relative to C iff C is high_n relative to Y. □

7.1. Generalizing the jump classes

When discussing the Turing degrees in general, rather than just the degrees below $\mathbf{0}'$, it turns out that the most fruitful way to proceed is to consider the relationship between the iterated jump of the given degree and its join with $\mathbf{0}'$.

Definition 7.8 *For $n \geq 1$, a degree $\boldsymbol{a}$ is **generalized low$_n$** (GL_n) if $\boldsymbol{a}^{(n)} = (\boldsymbol{a} \vee \mathbf{0}')^{(n-1)}$. A degree is **generalized high$_n$** (GH_n) if $\boldsymbol{a}^{(n)} = (\boldsymbol{a} \vee \mathbf{0}')^{(n)}$. If a degree is generalized low$_1$ then we also say that it is generalized low, and if a degree is generalized high$_1$ then we say that it is generalized high.*

To rephrase – a degree $\boldsymbol{a}$ is generalized low$_n$ if its nth jump is as low as it possibly could be in relation to $\boldsymbol{a} \vee \mathbf{0}'$, and is generalized high$_n$ if its nth jump is as high as it possibly could be in relation to $\boldsymbol{a} \vee \mathbf{0}'$. This might initially seem a little arbitrary. The fact that this is a useful definition comes from the fact that this is normally precisely the definition which is needed in order to carry local results through to the generalized case. If we work first with the degrees below $\mathbf{0}'$ and we find that all non-low$_2$ degrees satisfy a certain structural property, for example, it will then normally be the case that, in fact, the proof can be played with in order to give the same result for all non-GL$_2$. There is one consequence of this definition, however, which it is important to note. The generalized jump classes do not respect the ordering relation on the Turing degrees in the way that one might initially hope. In fact, every degree which is not above $\mathbf{0}'$ is bounded by a generalized low.

Theorem 7.4 *Every degree which is not above $\mathbf{0}'$ is bounded by a generalized low degree.*

Proof. First of all we note that the proof of Theorem 6.2 can easily be modified in order to give the stronger result, that for every non-zero degree $\boldsymbol{b} \leq \mathbf{0}'$ there is a *low* degree $\boldsymbol{a}$ such that $\mathbf{0}' = \boldsymbol{a} \vee \boldsymbol{b}$. We only sketch the modification since it is not complicated.

Now, rather than simply building $\boldsymbol{a}$ to be non-computable, we must satisfy lowness and in order to do this, we decide whether or not $\Psi_s(A;s)\downarrow$ at stage $s+1$ of the construction. Since the construction requires an oracle for $\emptyset'$ and we compute A' as the construction progresses, this suffices to show that $A' \leq_T \emptyset'$.

So now, at stage $s+1$, we perform a modified version of the search previously described. We let n be the least such that:

Case 1. $n \in B$ and there does not exist $\alpha \supset \alpha_s * \sigma_n$ such that $\Psi_s(\alpha;s)\downarrow$.
Case 2. $n \notin B$ and there does exist $\alpha \supset \alpha_s * \sigma_n$ such that $\Psi_s(\alpha;s)\downarrow$.

Such an n must exist, for precisely the same reasons as before –since otherwise B would be c.e., contrary to assumption.

If case 1 applies then $s \notin A'$ so long as we insist that $A \supset \alpha_s * \sigma_n$. So in this case we define $\alpha_{s+1} = \alpha_s * \sigma_n * \emptyset'(s)$ (recall that we code another bit of $\emptyset'$ into A with the last bit of α_{s+1}).

If case 2 applies then consider the first such α which is found by some fixed computable search procedure, and define $\alpha_{s+1} = \alpha * \emptyset'(s)$.

Now we are ready to prove the theorem. Given $\boldsymbol{d}$ which is not above $\mathbf{0}'$, let $\boldsymbol{b} = \boldsymbol{d} \vee \mathbf{0}'$ and observe that $\boldsymbol{d} < \boldsymbol{b} \leq \boldsymbol{d}'$. Now apply the result we just proved, relativized to $\boldsymbol{d}$ (one can easily verify that the proof relativizes). This gives $\boldsymbol{a} > \boldsymbol{d}$ such that $\boldsymbol{a}' = \boldsymbol{d}'$ and $\boldsymbol{a} \vee \mathbf{0}' = \boldsymbol{a} \vee (\boldsymbol{d} \vee \mathbf{0}') = \boldsymbol{d}'$, so $\boldsymbol{a} \vee \mathbf{0}' = \boldsymbol{a}'$ and $\boldsymbol{a}$ is generalized low. □

Now that we have defined the jump hierarchies, we shall go on to consider what structural properties are satisfied by the degrees in each of the jump classes. Beginning with very simple properties such as the cupping property and gradually moving to consider properties which are more complex, we shall look to answer questions of the form, "do all high degrees satisfy this property?", "do all non-GL$_2$ degrees satisfy this property?", and so on. We shall go on to consider such matters shortly, but first it is necessary to consider some classes of degrees that will be useful in this analysis. We shall consider the PA degrees, and then we shall briefly introduce the a.n.r. degrees and the 1-generic degrees.

8. PA degrees and Π^0_1 classes

Since Turing functionals can take elements of 2^ω as well as natural numbers as inputs, it makes sense to talk about computable subsets of 2^ω, or more generally, computable subsets of $\omega^k \times (2^\omega)^l$. A set $\mathscr{P} \subseteq \omega^k \times (2^\omega)^l$ is computable if there is a Turing functional which terminates given any element of $\omega^k \times (2^\omega)^l$, and which outputs 1 if this element is in $\mathscr{P}$ and outputs 0 otherwise. Then $\mathscr{P} \subseteq 2^\omega$ is Π^0_1 if it is defined by a Π^0_1 formula, i.e. if there exists some computable R such that:

$$\mathscr{P} = \{A : (\forall n)R[A,n]\}.$$

Similarly $\mathscr{P}$ is Σ^0_1 if there exists some computable R such that:

$$\mathscr{P} = \{A : (\exists n)R[A,n]\}.$$

One of the reasons that Π^0_1 classes are very important is that they crop up all over the place. Very often one will be working with a subset of 2^ω which will turn out to be a Π^0_1 class and one will then immediately be able to apply all of the results we have for Π^0_1 classes. By way of example, the set of complete and consistent extensions of any axiomatizable theory can be seen as a Π^0_1 class (where an axiomatizable theory is the deductive closure of a c.e. set of sentences). In order to see this fix an effective bijection between sentences in the language and ω. Then any string can be seen as a set of sentences. Consider an algorithm which, given any string σ and the set of sentences $\mathscr{T}'$ that it codes, outputs 1 unless it finds after searching for $|\sigma|$ many steps that any of the following conditions are satisfied, in which case it outputs 0:

(i) that there exists some sentence such that both it and its negation correspond to numbers less than $|\sigma|$ but neither it nor its negation is in $\mathscr{T}'$;
(ii) there exists some sentence in $\mathscr{T}$, whose code is strictly less than $|\sigma|$, which is not in $\mathscr{T}'$;
(iii) $\mathscr{T}'$ is not consistent.

Often $\mathscr{P} \subseteq 2^{<\omega}$ which is a Π^0_1 set is referred to as a Π^0_1-*class* – this is simply in order to emphasize the fact that it is a set of sets of natural numbers being considered, rather than just a set of natural numbers. Now we want to analyze these classes, to see something of what they look like. Our first aim is to find a very concrete way of picturing these objects. The following observations are easily verified.

- A Σ^0_1 class is a set $\mathscr{P}$ for which there exists a c.e. set of finite strings W such that elements of $\mathscr{P}$ are precisely those infinite strings which extend some element of W.
- A Π^0_1 class is a set $\mathscr{P}$ for which there exists some downward closed and computable set of finite strings Λ, such that $\mathscr{P}$ is the set of all infinite paths through Λ.

A set of strings is downward closed if all initial segments of any member of the set are also in the set. In this context, it is useful to consider a simple topology on 2^ω. We take as the basic open sets, those of the form $[\sigma]$ where σ is a finite binary string, and where $[\sigma]$ denotes the set of all infinite strings extending σ. According to this topology any Π^0_1 class is a closed set and any Σ^0_1 class is open. Compactness for this space (Cantor space) is given by the following weak form of König's lemma.

Lemma 8.1 (König's Lemma). *If Λ is a downward closed set of finite binary strings and is infinite, then there exists an infinite path through Λ.*

Proof. We define $A = \bigcup_s \alpha_s$ which is an infinite path through Λ one bit at a time.
Stage 0. Define $\alpha_s = \emptyset$.

Stage $s+1$. We are given α_s such that there exist infinitely many strings in Λ extending α_s. If there exist infinitely many strings in Λ extending $\alpha_s * 0$ then define α_{s+1} to be this string, otherwise define $\alpha_{s+1} = \alpha_s * 1$. □

8.1. Basis theorems

A basis theorem is a theorem which says that every set of a particular kind has a member of a particular kind. We shall establish three basis theorems for Π^0_1 classes.

Theorem 8.1 *Every non-empty Π^0_1 class has a member of c.e. degree.*

Proof. Let us suppose that $\mathscr{P}$ is a non-empty Π^0_1 class, and let Λ be a downward closed and computable set of finite binary strings such that $\mathscr{P}$ is the set of all infinite paths through Λ.

For each n let σ_n be the leftmost of all the strings of length n which has an extension in $\mathscr{P}$. Since $\mathscr{P}$ is a closed set, $A = \bigcup_n \sigma_n$ is an element of $\mathscr{P}$. Now it follows from König's lemma that if σ does not have any infinite extension in $\mathscr{P}$ then there exists m such that σ does not have any extension in Λ of length m. Therefore the set of finite strings to the left of A is computably enumerable, and is of the same degree as A. □

Definition 8.1 *If $\Lambda \subseteq 2^{<\omega}$ then we let $[\Lambda]$ denote the set of all infinite paths through Λ, i.e. the set of all $A \subset \omega$ which have an infinite number of initial segments in Λ.*

Probably the most cited theorem in the theory of Π^0_1 classes, is the low basis theorem of Jockusch and Soare:

Theorem 8.2 (The low basis theorem [JS]) *Every non-empty Π^0_1 class contains a member of low degree.*

Proof. We suppose we are given $\mathscr{P}$ which is a non-empty Π^0_1 class, and also a downward closed and computable set of strings Λ such that $\mathscr{P} = [\Lambda]$. We construct $A \in [\Lambda]$ of low degree. To construct A , we define a sequence of finite strings $\{\alpha_s\}_{s\in\omega}$ such that $A = \bigcup_s \alpha_s$. In order to help us define this sequence, we define also a sequence $\{\Lambda_s\}_{s\in\omega}$ such that $\Lambda_0 = \Lambda$ and each $\Lambda_{s+1} \subseteq \Lambda_s$.

We do all this in a sequence of stages, so that at stage s we define α_s and Λ_s. At stage s, what we have done is to decide that $\alpha_s \subset A$ and that $A \in [\Lambda_s]$. So at each stage we further restrict the Π^0_1 class of which A must be a member.

We run this construction using an oracle for $\mathbf{0}'$ in such a way that at stage $s+1$ we get to decide whether $s \in A'$. If we do this it is clear that A' will be computable in the halting problem.

The construction is defined precisely as follows.

Stage 0. Define $\alpha_0 = \emptyset$. Define $\Lambda_0 = \Lambda$. So far we know that $\alpha_0 \subset A$ and that $A \in [\Lambda_0]$.
Stage $s+1$. We have already defined α_s and Λ_s. We ask the following question (the fact that we can answer this question using an oracle for $\emptyset'$ follows from König's Lemma):

Does there exist $B \in [\Lambda_s]$ extending α_s such that $\Psi_s(B;s)\uparrow$?

If SO: we define Λ_{s+1} so that $[\Lambda_{s+1}]$ is the set of all B in $[\Lambda_s]$ of this kind. Since A will be in $[\Lambda_{s+1}]$ we know that $A'(s) = 0$. We define α_{s+1} to be some extension of α_s which has an infinite extension in $[\Lambda_{s+1}]$.

If NOT: we just define $\Lambda_{s+1} = \Lambda_s$ and define α_{s+1} to be some extension of α_s which has an infinite extension in $[\Lambda_{s+1}]$. Since A will be in $[\Lambda_{s+1}]$ we know that $A'(s) = 1$. □

Corollary 8.1 *There exists a complete and consistent extension of PA which is of low degree.*

Proof. We saw before that the complete and consistent extensions of any axiomatizable theory can be seen as a Π^0_1 class. The result then follows from the low basis theorem. □

Almost as important as the low basis theorem is a hyperimmune-free basis theorem.

Theorem 8.3 (The hyperimmune-free basis theorem, [JS]) *Every non-empty Π^0_1 class contains a member of hyperimmune-free degree.*

Proof. The basic format of the proof is very similar to the proof of the low basis theorem. Once again, we suppose we are given $\mathscr{P}$ which is a non-empty Π^0_1 class, and also a downward closed and computable set of strings Λ such that $\mathscr{P} = [\Lambda]$. We construct $A \in [\Lambda]$ which is of hyperimmune-free degree. Once again, to construct A, we define a sequence of finite strings $\{\alpha_s\}_{s\in\omega}$ such that $A = \bigcup_s \alpha_s$. In order to help us define this sequence, we define also a sequence $\{\Lambda_s\}_{s\in\omega}$ such that $\Lambda_0 = \Lambda$ and each $\Lambda_{s+1} \subseteq \Lambda_s$.

We do all this in a sequence of stages, so that at stage s we define α_s and Λ_s. At stage s, what we have done is to decide that $\alpha_s \subset A$ and that $A \in [\Lambda_s]$. So at each stage we further restrict the Π^0_1 class of which A must be a member.

A major difference is that, this time around, we do not have to worry about which oracle is required in order to run the construction. We run the construction in such a way that at stage $s+1$ we either force $\Psi_s(A)$ to be partial, or we define some computable g which majorizes $\Psi_s(A)$.

The construction is defined precisely as follows.

Stage 0. Define $\alpha_0 = \emptyset$. Define $\Lambda_0 = \Lambda$. So far we know that $\alpha_0 \subset A$ and that $A \in [\Lambda_0]$.
Stage $s+1$. We have already defined α_s and Λ_s. We ask:

Does there exist m and $B \in [\Lambda_s]$ extending α_s, such that $\Psi_s(B;m)\uparrow$?

If SO: we fix such an m and define Λ_{s+1} so that $[\Lambda_{s+1}]$ is the set of all B in $[\Lambda_s]$ of this kind. Since A will be in $[\Lambda_{s+1}]$ we know that $\Psi_s(A)$ is partial. We define α_{s+1} to be some extension of α_s which has an infinite extension in $[\Lambda_{s+1}]$.

If NOT: we just define $\Lambda_{s+1} = \Lambda_s$ and define α_{s+1} to be some extension of α_s which has an infinite extension in $[\Lambda_{s+1}]$. It is then easy to define some computable g which majorizes every $\Psi_s(B)$ such that $B \in [\Lambda_{s+1}]$. For any $m \in \omega$, in order to define $g(m)$ proceed as follows. Since there does not exist $B \in [\Lambda_s]$ extending α_s, such that $\Psi_s(B;m)\uparrow$, it follows from König's Lemma that there must exist some l such that $\Psi_s(\sigma;m)\downarrow$ for all strings σ in Λ_s of length l. We can search in a computable fashion until such an l is found, and then simply choose $g(m)$ to be greater than all of the values $\Psi_s(\sigma;m)$ such that σ is a string in Λ_s of length l.

This ends the construction. □

Corollary 8.2 *There exists a complete and consistent extension of PA of hyperimmune-free degree.*

Definition 8.2 *A Turing degree is PA if it contains a set which effectively codes a complete and consistent extension of PA.*

The following theorem, which we state without proof, means that the study of the PA degrees and the study of degrees of members of Π^0_1 classes are intimately related.

Theorem 8.4 (Solovay, extending Scott's basis theorem [DS]) *A degree $\boldsymbol{a}$ is PA iff every non-empty Π^0_1 class contains a member of degree below $\boldsymbol{a}$.*

The following theorem is very often useful.

Definition 8.3 *We say f is **diagonally non-recursive** (DNR) if, for all n, $f(n) \neq \Psi_n(n)$. If in addition, $f(n) \in \{0,1\}$ for all n, then we say that f is $0,1$-valued DNR. We say f is **fixed point free** (FPF) if, for all n, $\Psi_n(\emptyset) \neq \Psi_{f(n)}(\emptyset)$.*

Jockusch observed that the DNR degrees (those containing DNR functions) are precisely the FPF degrees (those containing FPF functions).

Theorem 8.5 *A degree is PA iff it contains a* $0,1$*-valued DNR function.*

Proof. On the one hand, the fact that any PA degree computes a $0,1$-valued DNR function follows from Theorem 8.4 and the fact that the set of all $0,1$-valued DNR functions is a Π^0_1 class. It is not difficult to see that the degrees containing $0,1$-valued DNR functions are upward closed.

On the other hand, if $\boldsymbol{a}$ contains a $0,1$-valued DNR function f, then this can be used to compute a path through any Π^0_1 class as follows. Let $\mathscr{P} = [\Lambda]$ for some downward closed computable Λ.

Stage 0. Define $\alpha_0 = \emptyset$.
Stage $s+1$. We are given α_s which has an infinite extension in $[\Lambda]$. Consider the algorithm which searches for $j \leq 1$ and l such that $\alpha_s * j$ does not have any extensions in Λ of length l, and which outputs the first such j found (and which does not terminate if no such j and l are found). Let i be such that the output of this algorithm is equal to $\Psi_i(\emptyset;i)$. Then $\alpha_s * f(i)$ has an infinite extension in $[\Lambda]$, so define α_{s+1} to be $\alpha_s * f(i)$. □

So far then, what do we know about what the PA degrees look like? Theorem 8.4 makes it clear that the PA degrees are upward closed. We know that there are PA degrees which are low, so certainly all degrees above $\mathbf{0}'$ are PA. There are PA degrees which are hyperimmune-free. Theorem 8.5 makes it clear that all PA degrees are fixed-point-free. The fact that the PA degrees are a proper subset of the fixed-point-free degrees can be seen through an analysis of the measure of these sets – the PA degrees are of Lebesgue measure 0, while the fixed-point-free degrees are of measure 1. Since we are not dealing with measure in this course, however, another way to see this is from the fact that there exists a minimal degree which is fixed-point-free, together with the following theorem.

Theorem 8.6 *Every PA degree strictly bounds a PA degree.*

The following property of the PA degrees will also be used later.

Theorem 8.7 ([AK]) *Every PA degree satisfies the cupping property.*

Proof. (Sketch) The proof we sketch here is an alternative proof due to the author. For the duration of this proof we assume that, for any i, τ, n, $\Psi_i(\tau;n)\downarrow$ only if this computation converges in $< |\tau|$ many steps and $\Psi_i(\tau;n')\downarrow$ for all $n' < n$.

Definition 8.4 *A total function* $T : n^{<\omega} \mapsto n^{<\omega}$ *is an* n***-branching tree*** *if, for all* $\sigma \in n^{<\omega}$ *and* $i < j < n$*:*

1. $T(\sigma) \subset T(\sigma * i)$, $T(\sigma) \subset T(\sigma * j)$*;*
2. $T(\sigma * j)$ *is incompatible with* $T(\sigma * i)$*.*

We move freely between thinking of trees as sets of strings or as functions from strings to strings—so we may specify a tree by describing its range.

What we aim to do is to construct a downward closed and computable set of finite binary strings Λ such that there exist infinite paths through Λ and such that if A is an infinite path through Λ then A computes some non-empty 2-branching T^A (let's say) such that no set lying on T^A computes A. In order to ensure that no set lying on T^A computes A it is convenient to construct a Turing functional Ψ such that no set lying on T^A computes $\Psi(A)$. In order to define T^A for any A which is an infinite path through Λ we shall define values T^τ for τ in Λ and then T^A will be defined to be the union of all T^τ such that $\tau \subset A$. Thus there are three different kinds of object under construction: Λ, T^A for A which is an infinite path through Λ and Ψ, and we must define these values in such a way that there exist infinite paths through Λ and so that the following requirements are satisfied:

$$\mathscr{N}_i : (A \in [\Lambda] \wedge C \in [T^A]) \to (\Psi_i(C;i) \neq \Psi(A;i)).$$

In fact, what we shall do here is just to consider how to satisfy a single requirement $\mathscr{N}_0$. We shall therefore only be concerned with the values $\Psi_0(C;0)$ and with defining Ψ on argument 0.

The most primitive form of the intuition runs as follows: if we are given four strings and we colour those four strings using two colours then there exists some colour such that at least two strings are not that colour (actually we only need three strings but it is convenient here to do everything in powers of two). Now we extend this idea. First we define a certain set of strings T. The role of T is that it is the set of all strings that could possibly be in T^τ for some $\tau \in \Lambda$. In the case that we are only looking to satisfy a single requirement T is rather trivial, we just define T to be the set of all strings of even length. The important thing about T is that it is 4-branching. We let $T(n)$ denote the set of strings in T of level $\leq n$ in T. Next, we consider a certain form of finite subset of T:

Definition 8.5 *We say that finite $T' \subset T$ is $(T,2)$-compatible if the strings of level n in T' are of length $2n$ and every string in T' which is not a leaf of T' has precisely two successors.*

The role of these $(T,2)$-compatible T' is that when we define T^τ for $\tau \in \Lambda$ actually we shall define this value to be some $(T,2)$-compatible T'. Recall that T' is said to be of level n if it is finite and all leaves are of level n. The following lemma is what we need in order to satisfy the first requirement:

Lemma 8.2. *For any finite $T' \subseteq 2^{<\omega}$ a 2-colouring of T' is an assignment of some $col(\sigma) \in \{0,1\}$ to each leaf σ of T'. For any n and any 2-colouring of $T(n)$ there exists T' which is $(T,2)$ compatible of level n and $d \in \{0,1\}$ such that no leaf σ of T' has $col(\sigma) = d$.*

Proof. This is easily seen by induction. The case $n = 0$ is trivial and, in fact, we have already seen the case $n = 1$. If we are given four strings and we colour those four strings using two colours then there exists some colour such that at least two strings are not that colour. Those two strings then define some $(T,2)$-compatible T' of level 1. In order to see the induction step suppose we are given a 2-colouring of $T(n+1)$. First we use this 2-colouring in order to define a 2-colouring of $T(n)$ as follows. Consider each leaf σ of $T(n)$. Such σ has precisely four successors in $T(n+1)$. If more than two of those successors are coloured 0 then colour σ with 0. If more than two of those successors are coloured 1 then colour σ with 1, and otherwise colour σ with 0. What this means is that if σ is not coloured d then at least two successors of σ are not coloured d. By the induction hypothesis there exists some $(T,2)$-compatible T' of level n and there exists d such that no leaf of T' is coloured d. In order to define T'' of level $n+1$ sufficient to complete the induction step, all we need do is to choose two successors of each leaf of T' which are not coloured d. □

Now we see how to use this lemma in order to satisfy $\mathscr{N}_0$ while also satisfying the condition that $[\Lambda]$ should be non-empty. Before defining Λ we define a set of strings $\Lambda^\star$. These are strings which may or may not be in Λ. We do not insist that $\Lambda^\star$ should be downward closed, in order to form Λ we shall later add strings in so that Λ is downward closed. For every n we let $\Lambda^\star(n)$ denote the set of strings in $\Lambda^\star$ which are of level n in $\Lambda^\star$. Thus for every n we must define the set of strings which are in $\Lambda^\star(n)$, for each such τ we must define a value T^τ which will be some $(T,2)$-compatible T' of level n (if $\tau \subset \tau'$ then we must also have that $T^\tau \subseteq T^{\tau'}$), and we must also ensure that if $n > 0$ then $\Psi(\tau)$ is defined on argument 0. In order to satisfy this latter condition we can just ensure that $\Psi(\tau;0)$ is defined for all τ of level 1 in $\Lambda^\star$ and then this task is done once and for all. We shall not describe here precisely how to define these values, but hopefully it is clear that we can do so in such a way that the following lemma is satisfied:

Lemma 8.3. *For any $n > 0$, any $(T,2)$-compatible T' of level n and any $d \in \{0,1\}$ there exists $\tau \in \Lambda^\star(n)$ such that $T^\tau = T'$ and $\Psi(\tau;0) = d$.*

The fact that we can satisfy Lemma 8.3 is really completely obvious. We are not insisting that $\Lambda^\star$ should be downward closed, so in order to ensure satisfaction of the lemma all we need do is to put enough strings into each $\Lambda^\star(n)$ so that all possibilities can be realised.

What this means is that if we define Λ by taking the strings in $\Lambda^\star$, adding in strings in order to make it downward closed and then removing any string τ (together with all extensions) for which it is the case that there exists $\sigma \in T^\tau$ with $\Psi_0(\sigma;0)\downarrow = \Psi(\tau;0)$ then for every n we must be left with strings in $\Lambda^\star(n)$ which are in Λ. In order to see this suppose we are given $n>0$. Then we can consider the values $\Psi_0(\sigma;0)$ for those $\sigma \in T(n)$ to define a 2-colouring of $T(n)$—where values are not defined to be either 0 or 1 we need not be concerned with them. Then Lemma 8.2 tells us that for this 2-colouring of $T(n)$ there exists some $(T,2)$-compatible T' of level n and there exists $d \in \{0,1\}$ such that no leaf of T' is coloured d. Fixing such T' and such d we may then apply Lemma 8.3 which tells us that there exists $\tau \in \Lambda^\star(n)$ with $T^\tau = T'$ and $\Psi(\tau;0)=d$. Then τ is a string in $\Lambda^\star(n)$ which is in Λ. By König's Lemma the fact that there exist an infinite number of strings in Λ suffices to ensure that $[\Lambda]$ is nonempty.

In order to satisfy all requirements we must become a little more sophisticated —we need more colours and bushier trees—but the basic idea remains the same. □

9. The a.n.r. degrees

We shall see later that the non-GL_2 degrees turn out to be an important class when we come to consider structural properties of degrees which are likely to to satisfied by degrees that are fairly high up in the Turing degree structure. Part of the reason for this, is that Theorem 10.4 often suffices to make uninteresting the question as to whether a property is satisfied by all degrees which are GL_2 or all degrees which are GL_2 and not GL_1.

When working with the degrees which are non-GL_2, it will invariably be a *domination* property of these degrees which we shall use in order to prove our results. The domination property in question, comes from a relativization of the following result.

Theorem 9.1 ([DM]) *A computes f which dominates all total computable functions iff $A' \geq_T \emptyset''$.*

Proof. Recall that $\mathrm{Tot}=\{i:\ (\forall n)\Psi_i(\emptyset;n)\downarrow\}$ is of degree $\mathbf{0}''$. First of all suppose that f dominates all total computable functions. Then we show how to compute g using an oracle for f, such that for all i, $\lim_s g(i,s) = \mathrm{Tot}(i)$. In order to decide $g(i,s)$ run all computations $\Psi_i(\emptyset;s')$ for $f(s)$ many steps, and for all $s' \leq s$. If all of these computations converge within $f(s)$ many steps then output 1, otherwise output 0. If $i \in \mathrm{Tot}$ then let $h(s)$ be the number of steps it takes for all computations $\Psi_i(\emptyset;s')$ to converge for $s' \leq s$. Then h is a computable function and so is dominated by f.

On the other hand, suppose that $A' \geq_T \emptyset''$. Then A can approximate Tot, and so computes g as just described. In order to compute the required f on argument s, proceed as follows for each $i \leq s$. Run the computation $\Psi_i(\emptyset;s)$ until either the computation converges or else we find $s' \geq s$ such that $g(i,s')=0$. In the first case let $n_i = \Psi_i(\emptyset;s)$, and in the second case define $n_i = 0$. Choose $f(s)$ greater than all n_i for $i \leq s$. □

Now we can relativize:

Theorem 9.2 *A is of non-GL_2 degree iff there is no function computable in $A \oplus \emptyset'$ which dominates all A computable functions.*

Proof. This follows by relativizing the proof of Theorem 9.1, since it follows directly from the definition that A is of degree which is GL_2 iff $A \oplus \emptyset'$ is high relative to A. □

A proof that all non-GL_2 degrees satisfy a certain property typically follows the following pattern. We begin by observing that $\mathbf{0}'$ satisfies the property in question, and then observe that, in fact, it is the ability of $\emptyset'$ to compute a sufficiently fast growing function f which allows us to prove this (when relativizing to A one will often find that it is now the ability of $A \oplus \emptyset'$ to compute a sufficiently fast growing function which is required). The final step then requires showing that the previous proof can be modified. This modification involves showing that, actually, it wasn't necessary to be able to compute f. In fact, it suffices to compute g not dominated by f.

Described out of context the paragraph above might seem rather abstract, so let us give an example.

Theorem 9.3 ([JP]) *All non-GL_2 degrees satisfy the cupping property.*

Proof. For the duration of this proof we assume that $\Psi_i(\tau;n)$ is defined only if the computation converges in $< |\tau|$ many steps. First of all, recall the proof that $\mathbf{0}'$ satisfies the cupping property, given in Section 6. In that proof an oracle for $\mathbf{0}'$ sufficed to build the required tree, essentially because, given any i and any σ, the oracle is able to decide whether or not there exist Ψ_i-splittings above σ. Now we see that, to put this another way, it sufficed to be able to compute some sufficiently fast growing function. For any τ we define $f(\tau)$ as follows. For each $i \leq |\tau|$ let s_i be the least such that there exist τ_0, τ_1 extending τ of length s_i and which are a Ψ_i-splitting and if there exist no such τ_0, τ_1 then let $s_i = 0$. Define $f(\tau) = \max\{s_i : i \leq |\tau|\}$. For every s define $f^\star(s) = \max\{f(\tau) : |\tau| = s\}$. Then $f^\star \leq_T \emptyset'$ and computing any function which grows as quickly as $f^\star$ does, allows one to decide when splittings exist.

Now suppose we are given A of non-GL_2 degree. Then there exists $g \leq_T A$ with $g(s) \geq f^\star(s)$ for infinitely many s. We can assume that g is an increasing function.

In order to show that A is perfectly cone avoiding we define $T = \bigcup_s T_s$ as follows. We define T as a set of finite binary strings, but it will be clear how to convert this into the appropriate function $2^{<\omega} \to 2^{<\omega}$.

Stage 0. We define T_0 to be $\{\emptyset\}$.
Stage $s+1$. For each leaf τ of T_s and each $i \leq |\tau|$ such that $\Psi_i(\tau)$ is compatible with A, check to see whether there exists a Ψ_i-splitting τ_0, τ_1 such that both these strings extend τ and are of length $\leq g(s)$.
If so (for some $i \leq |\tau|$): then let i be the least such, let $\tau' \supset \tau$ be as short as possible such that $\Psi_i(\tau')$ is incompatible with A and if $|\tau'| \leq s$ then enumerate the two one element extensions of τ' into T_{s+1}.
If not: then enumerate the two one element extensions of τ into T_{s+1}.

This ends the description of the construction.

It is a crucial detail that, in the first case above, we only enumerate the two one element extensions of τ' into T_{s+1} *if* $|\tau'| \leq s$. This means that, by the end of stage s, all strings enumerated into T are of length at most s. We know that there exist infinitely many s with $g(s) \geq f^\star(s)$, suppose that s is one of these stages. Let τ be a leaf of $\bigcup_{s' \leq s} T_{s'}$. Then at stage $s+1$ we search for enough steps to see all of the relevant splittings that exist—for every $i \leq |\tau|$ such that there exists a Ψ_i-splitting above τ there exists a splitting in which the strings are of length $\leq g(s)$. If there exists no such i, then we shall enumerate the two one element extensions of τ into T at stage $s+1$. Otherwise, let i be the least such, let $\tau' \supset \tau$ be as short as possible such that $\Psi_i(\tau')$ is incompatible with A, and let $|\tau'| = s'$. Then we shall not enumerate any proper extensions of τ into T until stage $s'+1$, when we shall enumerate in the two one element extensions of τ'. □

Now we are ready to define the a.n.r. degrees and to explain one of the reasons for their usefulness.

Definition 9.1 *We say that $A \leq_{wtt} B$ if $A \leq_T B$ via a Turing functional which has use on argument n bounded by $g(n)$ for some (total) computable function g.*

Definition 9.2 *A is **array non-recursive** (a.n.r.) if there is no function $f \leq_{wtt} \emptyset'$ which dominates every function computable in A. A degree is a.n.r. if its members are. A set A is array recursive if it is not array non-recursive.*

Since in this course we are using the terminology "computable" rather than "recursive", one might reasonably ask why we do not refer instead to the "array non-computable" degrees. The reason is that that degrees are standardly referred to by their acronym "a.n.r." rather than their full name "array non-recursive". To call them the a.n.c. degrees would probably lead to confusion.

Theorem 9.4 ([DJS]) *All degrees which are non-GL_2 are a.n.r..*

Proof. If A is of non-GL_2 degree then $A \oplus \emptyset'$ is not high relative to A. The result then follows from Theorem 9.2 – for *every* function f which is Turing reducible to $\emptyset'$ there is $g \leq_T A$ which is not dominated by f, so certainly this is also true for all those functions wtt reducible to $\emptyset'$. □

So the a.n.r. degrees are a superset of the degrees which are non-GL_2. In fact they are a proper superset, since it follows directly from the definition that they are upward closed. The key point here is that, when proving results concerning $\mathbf{0}'$ or the non-GL_2 degrees, it is often the existence of some quickly growing function $f \leq_T \emptyset'$ which allows us to put the proof through, and this function is very often wtt-reducible to $\emptyset'$. In this case we may well be able to put the same proof through in order to give the result for the a.n.r. degrees. Theorem 9.3 is an example.

Theorem 9.5 ([DJS]) *All a.n.r. degrees satisfy the cupping property.*

Proof. The function f in the proof of Theorem 9.3 is actually wtt reducible to $\emptyset'$, so the same proof suffices to give this stronger result. □

We state the following theorem without proof.

Theorem 9.6 ([DJS]) *There exist low degrees which are a.n.r..*

The class of a.n.r. degrees has many interesting interactions with other properties in various contexts. Perhaps the most impressive example is the following characterization of those c.e. degrees which have a strong minimal cover, due to Ishmukhametov.

Definition 9.3 *A degree $\boldsymbol{b}$ is a **minimal cover** for $\boldsymbol{a}$ if $\boldsymbol{b} > \boldsymbol{a}$ and there does not exist $\boldsymbol{c}$ with $\boldsymbol{a} < \boldsymbol{c} < \boldsymbol{b}$.*

Definition 9.4 *A degree $\boldsymbol{b}$ is a **strong minimal cover** for $\boldsymbol{a}$ if the degrees strictly below $\boldsymbol{b}$ are precisely the degrees below and including $\boldsymbol{a}$.*

Theorem 9.7 ([IS]) *A c.e. degree has a strong minimal cover iff it is array recursive.*

Proof. For the duration of this proof it is convenient to assume that, for any i, τ, n, $\Psi_i(\tau; n) \downarrow$ only if the computation converges in $|\tau|$ many steps and $\Psi_i(\tau; n') \downarrow$ for all $n' < n$. We give a proof which is a little more informative than the original and proceeds via a sequence of lemmas.

Definition 9.5 *We say $\boldsymbol{a}$ is a **tree basis** if, whenever it computes a perfect tree T, it computes a perfect pointed subtree of T (a tree is pointed if all paths compute the tree).*

Lemma 9.1 ([AL]). *If $\boldsymbol{a}$ is a tree basis then it has a strong minimal cover.*

Proof. First we need to consider how we can relativize the minimal degree construction, and what happens when we do so. Suppose we are given A. We can relativize the minimal degree construction to A by starting with a tree T_0 which contains all strings of the form $\sigma \oplus \tau$ such that $\sigma \subset A$, rather than the full binary tree. This ensures that the set we construct is of degree above A, but means that now we must work with A computable trees rather than partial computable trees. Suppose that the set we are constructing is B. When we force B to lie on a Ψ_i-nonsplitting tree, this now means that $\Psi_i(B)$ is either partial or computable in A. When we force B to lie on a Ψ_i-splitting tree, this now means that $A \oplus \Psi_i(B)$ computes B. What results, then, is a degree which is a minimal cover for $\boldsymbol{a} = \deg(A)$, but not necessarily a strong minimal cover. Now, however, suppose that we know $\boldsymbol{a}$ is a tree basis. In order to construct a strong minimal cover for $\boldsymbol{a}$ we begin just as if we were trying to construct a minimal cover for $\boldsymbol{a}$. At stage $s+1$, given T_s and β_s, we ask, "does there exist any string $\tau \in T_s$ such that no two strings extending τ in T_s are Ψ_s-splitting?"

If SO: then we define T_{s+1} to be the subtree of T_s above τ.

If NOT: then we can let T^0 be a perfect A-computable Ψ_s-splitting subtree of T_{s+1}. Now let T^1 be such that $\tau \in T^1$ iff $\tau = \Psi_s(\sigma)$ for some $\sigma \in T^0$, so T^1 is a perfect A-computable tree. Now we use the fact that $\boldsymbol{a}$ is a tree basis. Let $T^2 \subseteq T^1$ be a perfect A-computable tree such that all paths through the tree compute A. Then T^3 such that $\sigma \in T^3$ iff $\Psi_s(\sigma) \in T^2$ is a perfect A-computable tree. We can define $T_{s+1} = T^3$. If B lies on T_{s+1} then $B \leq_T \Psi_s(B)$. □

The next couple of lemmas we need concern the c.e. traceable degrees.

Definition 9.6 *$A \subseteq \omega$ is **c.e. traceable** if there is a computable function p such that for every function $f \leq_T A$ there is a computable function h such that $|W_{h(n)}| \leq p(n)$ and $f(n) \in W_{h(n)}$ for all $n \in \omega$.*

How should one understand this definition? The function p here can be thought of as a *bounding* function and the function h can be thought of as a *guessing* function. Thus A is c.e. traceable if there exists some computable bounding function p such that for every $f \leq_T A$ there exists some computable guessing function h which makes at most $p(n)$ guesses as regards each value $f(n)$ and one of these guesses is always correct. The next lemma shows that the choice of bounding function is fairly arbitrary.

Lemma 9.2 ([TZ]). *If A is c.e. traceable then, for any increasing and unbounded computable function g such that $g(n) \geq 1$ for all n, and for every function $f \leq_T A$, there is a computable function h such that $|W_{h(n)}| \leq g(n)$ and $f(n) \in W_{h(n)}$ for all n.*

Proof. We freely identify finite strings and natural number codes for them, for the sake of readability. Let g be any increasing and unbounded computable function such that $g(n) \geq 1$ for all n. Let p be an increasing computable function such that for every function $f \leq_T A$ there is a computable function h such that $|W_{h(n)}| \leq p(n)$ and $f(n) \in W_{h(n)}$ for all $n \in \omega$. We can assume $p(0) = 1$. Now let r be an increasing and unbounded computable function which grows sufficiently slowly that for all n:

$$p(r(n)) \leq g(n).$$

Let q be an increasing computable function such that for all n:

$$q(r(n)) > n.$$

Suppose we are given $f \leq_T A$ and let $f^*(n) = f \upharpoonright q(n)$. Let h be a computable function such that $|W_{h(n)}| \leq p(n)$ and $f^*(n) \in W_{h(n)}$ for all n. Let $h^*(n) = i$ such that $W_i = \{\tau(n) | \ \tau \in W_{h(r(n))}\}$. Then there are at most $p(r(n)) \leq g(n)$ many strings in $W_{h(r(n))}$ and (we can assume that) each is of length $q(r(n)) > n$. Also, $f \upharpoonright q(r(n)) \in W_{h(r(n))}$, so $f(n) \in W_{h^*(n)}$. □

Lemma 9.3 ([AL2]). *Every c.e. traceable degree is a tree basis.*

Proof. Suppose we are given A which is c.e. traceable. By the previous lemma this means that, for every function $f \leq_T A$ there is a computable function h such that $|W_{h(n)}| \leq 2^n$ and $f(n) \in W_{h(n)}$ for all $n \in \omega$.

Recall that for any finite tree T', we say T' is of level n if the domain of T' is all strings of length $\leq n$. In what follows we shall subdue mention of effective codings between strings and natural numbers and between finite sets of strings and natural numbers for the sake of readability. So now assume that A is c.e. traceable, $T \leq_T A$ and that T is perfect. In fact we may assume without loss of generality that that we are given Ψ such that:

- for any τ, the value $\Psi(\tau)$ is a finite tree of level n for some n and can be computed in $|\tau|$ many steps,
- $\Psi(A) = T$,
- for any $\tau \subset \tau'$, $\Psi(\tau')$ extends $\Psi(\tau)$ as a function from strings to strings.
- if $\Psi(\tau)$ is of level $n > 0$ then there exists $\tau' \subset \tau$ such that $\Psi(\tau')$ is of level $n-1$.

So Ψ is just a functional via which A computes T and which behaves in a nice tidy fashion. First we define the function $f \leq_T A$ as follows; for every n, $f(n)$ is the shortest initial segment of A, τ say, such that $\Psi(\tau)$ is of level $\Sigma_{i=0}^{n}(i+2)$. Since A is c.e. traceable we can then take computable h such that $|W_{h(n)}| \leq 2^n$ and $f(n) \in W_{h(n)}$ for all $n \in \omega$. We can assume that if $n > 0$ then τ is not enumerated into $W_{h(n)}$ unless some initial segment of τ has already

been enumerated into $W_{h(n-1)}$ and unless $\Psi(\tau)$ is of level $\Sigma^n_{i=0}(i+2)$ and this is not the case for any proper initial segment of τ.

Next we proceed to computably enumerate various values $T_0(\tau)$ and $T_1(\tau)$ and axioms for a Turing functional Φ such that $T_0(A)$ is a 2-branching subtree of T and for every set C lying on $T_0(A)$ we have $\Phi(C)=A$. We construct T_1 just as an auxiliary function which is useful in defining the construction.

The basic idea is just this. Each value $W_{h(n)}$ only has a small number of trees in it, while each of these trees has a large number of strings. This allows us to pick out strings in each tree which are chosen specifically for that tree, and which can then be mapped to the tree via Φ.

Initially all strings are *available for use* (which means that there isn't any τ such that they have been put into $T_0(\tau)$ yet). There can be only a single string enumerated into $W_{h(0)}$, τ say. We define $T_1(\tau)$ to be the strings of level 0 and 2 in $\Psi(\tau)$ and we define $T_0(\tau)$ to be the string, σ say, of level 0 in $\Psi(\tau)$. We declare σ to be *unavailable for use* and enumerate the axiom $\Phi(\sigma)=\tau$. Whenever some τ is enumerated into $W_{h(n)}$ for $n>0$ we shall already have enumerated a (unique and proper) initial segment of this string, τ' say, into $W_{h(n-1)}$. For each leaf σ of $T_0(\tau')$ there will be precisely 2^{n+1} successors in $T_1(\tau')$. Let σ_0 and σ_1 be the first two of these which are still available for use, enumerate these strings into $T_0(\tau)$ and enumerate all leaves σ' of $\Psi(\tau)$ which extend these two strings into $T_1(\tau)$ before enumerating the axioms $\Phi(\sigma_0)=\tau$, $\Phi(\sigma_1)=\tau$. Declare σ_0 and σ_1 to be unavailable for use.

In order to see that the axioms enumerated for Φ are consistent observe that when we enumerate an axiom $\Phi(\sigma)=\tau$, σ is a leaf of $T_0(\tau)$. The consistency of the axioms enumerated for Φ then follows from the fact that it is easily seen by induction on the stage of the construction that if τ and τ' are incompatible and we have defined values $T_0(\tau),T_0(\tau')$ then the leaves of these two sets of strings are pairwise incompatible. The fact that $T_0(A)$ is 2-branching and that for every $C\in T_0(A)$ we have $\Phi(C)=A$ then follows immediately from the description of the construction. □

We just need one more lemma:

Lemma 9.4 ([IS]). *Any c.e. degree is c.e. traceable iff it is array recursive.*

Proof. Suppose A is a c.e. set. If A is a.n.r. then its degree satisfies the cupping property, and so, by the previous lemmas A cannot be c.e. traceable. Now suppose that A is array recursive, and let $g\leq_{wtt}\emptyset'$ dominate every A computable function. Since g can be computed from $\emptyset'$ with use bounded by a computable function, we can take a computable function p and a computable function h^* such that $|W_{h^*(n)}|<p(n)$ and $g(n)\in W_{h^*(n)}$ for all n. Given an enumeration of A and $f\leq_T A$ let $f=\Psi_i(A)$ and let $k(n)$ be the least s such that $\Psi_i(A_s;n)[s]\downarrow$ and A_s below the use of this computation is actually an initial segment of A. Let $W_{h(n)}$ be the set of values $\Psi_i(A_s;n)[s]$ such that there exists $t\in W_{h^*(n)}$ and s is the first stage $s'\geq t$ at which $\Psi_i(A_{s'};n)[s']\downarrow$. There are less than $p(n)$ such values and, since g dominates k, for all but finitely many n one of these values is $f(n)$. □

Now we just need to put these lemmas together. Suppose A is a c.e. set. If A is a.n.r. then its degree satisfies the cupping property and so does not have a strong minimal cover. If A is not a.n.r. then it is c.e. traceable, and so is a tree basis and so has a strong minimal cover. □

We shall see more about the a.n.r. degrees later.

10. The 1-generic degrees

There is little one can say in general about $\mathscr{D}[\leq \boldsymbol{a}]$ which is true for all $\boldsymbol{a}$ which are low, or for all $\boldsymbol{a}$ which are generalized low. The 1-generic degrees, however, are a nice subset of the degrees which are GL_1, for which $\mathscr{D}[\leq \boldsymbol{a}]$ is relatively rich in structure.

Definition 10.1 *A set $A\subseteq\omega$ is 1-generic if for every c.e. set of strings W, either:*

(i) $(\exists\sigma\subset A)[\sigma\in W]$*; or*
(ii) $(\exists\sigma\subset A)(\forall\tau\supseteq\sigma)[\tau\notin W]$.

A degree is 1-generic if it contains a 1-generic set.

Theorem 10.1 *If A is 1-generic then its degree is GL_1.*

Proof. Consider $W = \{\tau |\ \Psi_e(\tau;e)\downarrow\}$. If A is 1-generic then either (i) or (ii) of definition 10.1 holds. Using an oracle for $A \oplus \emptyset'$ we can successively test initial segments σ of A to see whether either of (i) or (ii) hold for that σ. When we find σ for which this is true, this tells us whether $e \in A'$. Therefore $A' \leq_T A \oplus \emptyset'$. □

Theorem 10.2 *Every degree $\boldsymbol{b} \geq \mathbf{0}'$ is the jump of a 1-generic.*

Proof. Given an oracle for $B \geq_T \emptyset'$ we construct A which is 1-generic. Since all 1-generics are generalized low, in order to show that $A' \equiv_T B$, it suffices to ensure that $A \oplus \emptyset'$ can compute B. This will follow since we code one more bit of B into A at each stage of the construction, and an oracle for $A \oplus \emptyset'$ will suffice to retrace the construction and so determine the coded bits of B. We construct A as the union of a sequence of strings $\{\alpha_s\}_{s\in\omega}$.

Stage 0. Define $\alpha_0 = \emptyset$.
Stage $s+1$. We identify finite strings and natural numbers codes for them, so that W_s is the sth c.e. set of strings. Using an oracle for $\emptyset'$ determine whether there exists $\alpha \supseteq \alpha_s$ which extends some element of W_s. If not then define $\alpha_{s+1} = \alpha_s * B(s)$. Otherwise let α be the first such, and define $\alpha_{s+1} = \alpha * B(s)$. □

Theorem 10.3 *If $\boldsymbol{a}$ is 1-generic, then there exists an independent sequence of degrees below $\boldsymbol{a}$.*

Proof. Let A be 1-generic, we show that the sequence $\{A^{[i]}\}_{i\in\omega}$ is independent. For each i and each finite F such that $i \notin F$ consider the set:

$$W = \{\tau |\ (\exists x \in \omega)(\exists \sigma \subset \tau^{[F]})[\Psi_i(\sigma;x)\downarrow \neq \tau^{[i]}(x)]$$

Since A is 1-generic, either (i) or (ii) of Definition 10.1 holds. If (i) holds then $\Psi_i(A^{[F]}) \neq A^{[i]}$. So suppose (ii) holds for σ. Let x be such that $\sigma^{[i]}(x)\uparrow$. If it was the case that $\Psi_i(A^{[F]};x)\downarrow$ then it would be the case that (i) holds for some extension of σ, so it must be the case that $\Psi_i(A^{[F]};x)\uparrow$. □

Corollary 10.1 *If $\boldsymbol{a}$ is 1-generic, then $\exists_1 \cap Th(\mathscr{D}[\leq \boldsymbol{a}])$ is decidable.*

Having done the ground work we are now ready to study the structural properties satisfied by the degrees in each of the various jump classes. We begin by considering simple properties and then work our way gradually to consider properties which are more complex. The following theorem means that the right choice of upper semi-lattice can often be used in order to show that a given structural property is not satisfied by all degrees which are low or by all degrees which are low_2.

Theorem 10.4 ([ML]) *Let $\mathscr{P}$ be a finite upper semi-lattice. Then there is a low degree $\boldsymbol{a}$ such that $\mathscr{D}[\leq \boldsymbol{a}]$ is isomorphic to $\mathscr{P}$, and there also exists a degree which is low_2 non-low of this kind.*

11. Minimal degrees and the jump classes

By Theorem 8.6 no PA degree is minimal. We saw earlier that there exist low PA degrees, and that the PA degrees are upward closed. It's also true that every low degree is bounded by a degree in each jump class. This can be seen by relativizing Theorem 7.3, because it follows straight from the definition that (for $n > 0$) any degree which is low_n relative to a low degree is still low_n, and that any degree which is high_n relative to a low degree is still high_n. Putting

all these facts together, we see that every jump class has members which are not of minimal degree. The question which remains is, which jump classes do have members of minimal degree?

Theorem 11.1 ([JP]) *If $\boldsymbol{a}$ is non-GL$_2$ then $\boldsymbol{a}$ bounds a 1-generic.*

Proof. The proof follows the same structure that we discussed earlier in Section 9. First of all, one thinks about how one would prove that $\mathbf{0}'$ bounds a 1-generic, and then one thinks how to rephrase this proof so that it is the ability of $\emptyset'$ to compute a sufficiently fast growing function f which allows the proof to go through.

In this case we define the function f as follows. Given $\tau \in 2^{<\omega}$, for each $i \leq |\tau|$ let s_i be the least s such that there is a string $\sigma \supset \tau$ in $W_{i,s}$, putting $s_i = 0$ if there exists no such s (again we identify finite strings and natural numbers codes for them, so that W_i may be regarded as a set of finite strings). Let $f^*(\tau) = \max\{s_i |\ i \leq |\tau|\}$ and, for all n, define $f(n) = \max\{f^*(\tau) |\ |\tau| \leq n\}$.

Given A which is of non-GL$_2$ degree, choose an increasing function $g \leq_T A$ which is not dominated by f. We define $B = \bigcup_s \sigma_s$ as follows.

Stage 0. Define $\sigma_0 = \emptyset$.
Stage $s+1$. For each $i \leq |\sigma_s|$ such that there does not exist any initial segment of σ_s in $W_{i,s}$, check to see whether there exists a proper extension of σ_s in $W_{i,g(s)}$.

If SO (for some i): then let i be the least such. Let τ be the first extension of σ_s enumerated into W_i. If $|\tau| \leq s+1$ then define $\sigma_{s+1} = \tau$, otherwise define $\sigma_{s+1} = \sigma_s$.

If NOT: then define $\sigma_{s+1} = \sigma_s * 0$.

In order to see that the construction does what it is supposed to, consider any s such that $g(s) > f(s)$. The key point is that $|\sigma_s| \leq s$. At stage $s+1$ there are two possibilities. The first possibility is that for each $i \leq |\sigma_s|$ the following holds: there is already a string in $W_{i,s}$ which is an initial segment of σ_s, or else there are no strings in W_i properly extending σ_s. In this case we define $\sigma_{s+1} = \sigma_s * 0$. The second possibility is that there is some least i for which this does not hold – so there is a first string $\tau \supset \sigma_s$ enumerated into W_i. Let $|\tau| = s'$. At stage s' we shall define $\sigma_{s'} = \tau$. □

By Theorem 10.3 no 1-generic degree is minimal. By Theorem 11.1, it therefore follows that all minimal degrees are GL$_2$. The fact that there are minimal degrees which are low follows from the fact that any c.e. degree bounds a minimal degree. In order to prove this, however, requires being able to construct a minimal degree by full approximation, and this is a technique which we will not cover in this course. We restrict ourselves to showing that there are minimal degrees which are GL$_2$ but not GL$_1$.

Theorem 11.2 ([LS]) *There exists a minimal degree which is GL$_2$ but not GL$_1$.*

Proof. Since we are not worried about whether or not the minimal degree we construct is below $\mathbf{0}'$, we use a modification of the construction which builds minimal degrees using perfect trees. In order to ensure that A is not GL$_1$ it suffices to satisfy the requirements:

V_e : $A' \neq \Psi_e(A \oplus \emptyset')$.

Now we intersperse the action required to satisfy these requirements with the original construction, so we might act to satisfy these requirements at odd stages for example. Suppose that at some stage of the construction it is already decided that A will lie on the perfect tree T. In order to satisfy the requirement V_e we consider the **narrow** subtree of T:

Definition 11.1 *If T is a perfect tree, then the **narrow subtree** of T, T^* say, is defined as follows. For σ of length n, $T^*(\sigma) = T(\sigma \oplus 0^n)$, where 0^n is the sequence of n zeros.*

The crucial point about the narrow subtree T^* is that, for every $\sigma \in T^*$ there exists $\tau \supset \sigma$ which is in T but not in T^* – at any point in a finite extension argument it is still possible to leave T^* and remain in T. Note that if T is computable then its narrow subtree is computable too. Given an index for T^* we can effectively find i such that $i \in A'$ iff A does NOT lie on T^*.

In order to satisfy V_e, we just have to ask, does there exist $\sigma \in T^*$ such that $\Psi_e(\sigma \oplus \emptyset'; i) \downarrow = 0$?

If NOT: then we satisfy the requirement simply by having A lie on T^*. If we do this, then either $\Psi_e(A \oplus \emptyset'; i) \uparrow$ or $\Psi_e(A \oplus \emptyset'; i) \downarrow = 1$ but $i \notin A'$.

If SO: then let $\sigma \in T^*$ be such that $\Psi_e(\sigma \oplus \emptyset'; i) \downarrow = 0$. There exists $\tau \supset \sigma$ which is in $T - T^*$ so now we just have to insist that A lies on the subtree of T above τ. Then $i \in A'$ but $\Psi_e(A \oplus \emptyset'; i) \downarrow = 0$. □

12. The cupping property

All relations between the cupping property and the jump classes are fully understood. We have seen already that all a.n.r. degrees (and so all degrees which are non-GL_2) satisfy the cupping property. We saw that all PA degrees satisfy the cupping property, that there are PA degrees which are low and PA degrees which are low_2 non-low (since the PA degrees are upward closed and a low degree is bounded by members of each and every jump class). So we know that there are low degrees and also low_2 non-low degrees which satisfy the cupping property. On the other hand there are also low degrees and low_2 non-low degrees which have a strong minimal cover, and no degree with a strong minimal cover satisfies the cupping property.

There are, however, many open questions concerning the cupping property which remain. We list a few here.

Question 12.1 *Does every degree either have a strong minimal cover or satisfy the cupping property?*

Question 12.2 *Is every degree either a tree basis or else perfectly cone avoiding?*

A positive answer to Question 12.2 would give a positive answer to Question 12.1 and would also give a positive solution to an old question of Yates which is one of the longstanding questions of degree theory: does every minimal degree have a strong minimal cover? This follows because:

Theorem 12.1 ([AL3]) *Every perfect and non-computable tree computes one of its non-computable paths.*

Theorem 12.1 suffices to show that no minimal degree is perfectly cone avoiding. In order to see this suppose that A is of minimal degree and computes the perfect tree T. Then A computes a non-computable path of T, and since A is of minimal degree this path must therefore be of the same degree as A. A positive solution to Question 12.2 would therefore mean that the degree of A must be a tree basis, and so have a strong minimal cover. A positive solution to Question 12.2 would also give a positive solution to the following question:

Question 12.3 *Are the degrees which satisfy the cupping property precisely those which are perfectly cone avoiding?*

13. The join property

This is a property which is more interesting in this context, basically because it is not upward closed. First consider the non-GL_2 degrees:

Theorem 13.1 ([DGLM]) *All non-GL_2 degrees satisfy the join property.*

Proof. The proof follows the format for non-GL_2 proofs that we described in Section 9. So we look at how the proof works for $\mathbf{0}'$, and then consider how to modify this proof. In Section 6 we gave a proof that $\mathbf{0}'$ satisfies the join property. The reader is invited to verify for themselves that the proof given there is easily modified to show that for all non-zero $\mathbf{a} < \mathbf{0}'$ there exists $\mathbf{b} < \mathbf{0}'$ which 1-generic with $\mathbf{a} \vee \mathbf{b} = \mathbf{0}'$. In the present proof we shall also build a joining partner which is 1-generic.

So we suppose we are given D of degree which is not GL_2, and A which is of non-zero degree strictly below that of D. We want to construct $B <_T D$ such that $A \oplus B \equiv_T D$. As before, we define

$$\sigma_n = \underbrace{000\cdots}_{n \text{ zeros}}1.$$

If $s = \langle m,n \rangle$ let $\text{left}(s) = m$. Once again, we can assume that A is not computably enumerable. First of all, we have to define the appropriate fast growing function.

Given any $\sigma \in 2^{<\omega}$ and any $e,s \in \omega$, let $g(\sigma,e,s)$ be defined in the following way. First, let n be the least such that:

$n \in A \Rightarrow \nexists \tau \in W_{e,s}$ with $\tau \supseteq \sigma * \sigma_n$;
$n \notin A \Rightarrow \exists \tau \in W_{e,s}$ with $\tau \supseteq \sigma * \sigma_n$.

If $n \in A$ define $g(\sigma,e,s) = \sigma * \sigma_n$. If $n \notin A$ then let $g(\sigma,e,s)$ be the first string enumerated into $W_{e,s}$ with $\tau \supseteq \sigma * \sigma_n$.

So $g(\sigma,e,s)$ basically tells us how the join construction (modified to build a 1-generic joining partner) would proceed at some stage if σ was the initial segment we have already constructed, if we were looking to satisfy the eth genericity requirement at this stage, but providing we only search for s many steps in order to try to ascertain the correct way of extending σ. Let $g^*(\sigma,e)$ be the least s such that either $g(\sigma,e,s)$ extends a string in $W_{e,s}$, or there doesn't exist any string in W_e extending $g(\sigma,e,s)$. Then, roughly speaking, $g^*(\sigma,e)$ tells us the least s such that $g(\sigma,e,s)$ is the "correct" extension.

Given any $f : \omega \to \omega$ we define a set $B_f = \bigcup_s \tau_{f,s}$ as follows:

Stage 0. Define $\tau_{f,0} = D(0)$.

Stage $t+1$. Define $\tau_{f,t+1} = g(\tau_{f,t}, \text{left}(t), f(t)) * D(t+1)$.

So B_f is the set that the join construction will build, if we use f in order to bound how many steps we search for at each stage in order to try to determine the correct way to extend.

Now for any t the set $\Pi_t = \{\tau_{f,t} \mid f : \omega \to \omega\}$ is finite (since once you search for "enough" steps at a certain stage you will never subsequently change your mind about how to extend at this stage), so let f_0 be an increasing function computable in $A \oplus \emptyset'$ such that, for all t and all $\tau \in \Pi_t$, $f_0(t)$ is greater than $g^*(\tau, \text{left}(t))$. Then f_0 is a function which bounds how long we have to search at stage s of the join construction in order to proceed correctly if we wish to satisfy the $(\text{left}(t))$th genericity requirement, *no matter how we have proceeded at previous stages*, i.e. even if we did not search for long enough to proceed "correctly" at previous stages. One last little detail we have to be concerned with, is that when we choose a function f_2 not dominated by f_0, we cannot be sure for which arguments t we will have that $f_2(t) > f_0(t)$—we want to be sure that for each e this happens for some t with $\text{left}(t) = e$. Let h be a computable and increasing function such that, for all t and all $e \leq t$, there exists t' with $t < t' < h(t)$ and $\text{left}(t') = e$. Define $f_1(t) = f_0(h(t))$ for all t. Since D is not GL_2, we may let f_2 be an increasing function computable in D which is not dominated by f_1 and then define $B = B_{f_2}$.

That D is computable in $A \oplus B$ follows using precisely the same argument as we used when proving that $\mathbf{0}'$ satisfies the join property. It remains to show that B is 1-generic. In order to see this, let $t > e$ be such that $f_2(t) > f_1(t)$. Then there exists t' such that $t < t' < h(t)$ and $\text{left}(t') = e$. We have that $f_2(t') > f_2(t) > f_1(t) = f_0(h(t)) > f_0(t')$, so that $f_2(t') > g^*(\tau_{f_2,t'}, e)$ as required. □

Theorem 10.4 then suffices to show that there are low degrees and also low_2 non-low degrees which satisfy the join property, as well as degrees of these kinds which do not.

The situation becomes more interesting, however, when we consider the upward closure – for which degrees is it the case that all degrees above satisfy the join property? In looking to answer this question, it becomes natural to consider upward closed classes of degrees which we are used to working with. Do all PA degrees satisfy the join property? Do all a.n.r. degrees satisfy the join property? Another natural question is as to whether satisfaction of the cupping property is equivalent to all degrees above satisfying join. We can answer all of these questions with the following result:

Theorem 13.2 ([AL4]) *All low fixed point free degrees fail to satisfy the join property.*

The techniques used to prove this theorem are a combination of Kučera's fixed point free permitting below $\emptyset'$ [AK2] and the techniques developed independently by Cooper [BC4], and also Slaman and Steel [SS] in order to show that there exist c.e. degrees which do not satisfy the join property (when considered as elements of the Δ^0_2 degrees). The proof of the latter result we shall sketch later in this section. We shall not cover the fixed point free permitting technique in this course, an account is given in Nies' book [AN].

Corollary 13.1 *Above each low degree, there is a low degree which doesn't satisfy join.*

Proof. Each low degree is bounded by a degree which is low and fixed-point-free. To see this, apply the low basis theorem relative to a low degree **a**, in order to deduce that there is a degree **b** which is PA relative to **a** and low over **a**. The degree **b** is then PA by Theorem 8.5, and is also low, since any degree which is low over a low degree is still low. □

The following corollary concerns Martin-Löf randomness, which we don't define here. For an introduction to the study of algorithmic randomness (and, in particular, Martin-Löf randomness), we refer the reader to [AN] and [DH].

Corollary 13.2 *There are PA/a.n.r./Martin-Löf random degrees which don't satisfy join.*

Proof. All PA degrees and all Martin-Löf random degrees are fixed-point-free, and it follows from the low basis theorem that there exist PA degrees which are low and also Martin-Löf random degrees which are low. For the a.n.r. degrees, the corollary follows from Corollary 13.1 and the fact that the a.n.r. degrees are upward closed. □

Corollary 13.3 *Satisfaction of the cupping property is not equivalent to all degrees above satisfying join.*

Proof. By Theorem 8.6 any PA degree properly bounds another PA degree. The result then follows from Corollary 13.2 and the fact that all PA degrees satisfy the cupping property. □

There is another theorem along the same lines:

Theorem 13.3 ([AL4]) *Above every low c.e. degree, there is a low c.e. degree which doesn't satisfy join.*

This leaves open various questions. Working below $\mathbf{0}'$ first of all, we have already managed to separate the non-low_2 degrees from the low degrees, and it seems natural to ask if this can be extended to give a definition of low_2:

Question 13.1 *In $\mathscr{D}[\leq \mathbf{0}']$, are the low_2 degrees precisely those for which it is not the case that all degrees above satisfy join?*

It remains open, in fact, whether $\mathbf{0}'$ can be given an extremely simple definition using the join property, although one would expect the following question to be answered in the negative:

Question 13.2 *Can $\mathbf{0}'$ be defined as the least degree such that all degrees above satisfy join?*

There is an interesting observation to be had here. Let us suppose for a moment that question 13.2 receives a negative response, and that, in fact, any degree which bounds a high degree satisfies join. This moves us towards a definition of $\mathbf{0}'$ in another direction. Immediately from this we get a formula sufficient to distinguish $\mathbf{0}'$ from all degrees comparable with it, because then all degrees above $\mathbf{0}'$ satisfy the following formula, while no degree strictly below $\mathbf{0}'$ does so:

$$\mathbf{a} \text{ satisfies mincup, and } \forall \mathbf{b} \geq \mathbf{a},\ \forall \text{ minimal degrees } \mathbf{m} < \mathbf{b}, \mathbf{b} \text{ satisfies join}(\mathbf{m}),$$

where $\mathbf{a}$ satisfies mincup if for all $\mathbf{b} \geq \mathbf{a}$ there exists a minimal degree $\mathbf{m} < \mathbf{b}$ with $\mathbf{a} \vee \mathbf{m} = \mathbf{b}$, and where join$(\mathbf{m})$ means join above $\mathbf{m}$, i.e. $\mathbf{b}$ satisfies join$(\mathbf{m})$ if, for all $\mathbf{c}$ with $\mathbf{m} < \mathbf{c} < \mathbf{b}$ there exists $\mathbf{d}$ with $\mathbf{m} < \mathbf{d} < \mathbf{b}$ and $\mathbf{d} \vee \mathbf{c} = \mathbf{b}$. Let us see first why degrees above $\mathbf{0}'$ satisfy the formula above. All such degrees satisfy mincup – this follows from the fact that $\emptyset'$ computes a perfect tree of sets of minimal degree. If $\mathbf{b} \geq \mathbf{0}'$ bounds a minimal degree $\mathbf{m}$ then, since all minimal degrees are GL_2, $\mathbf{b}$ bounds a degree which is high relative to $\mathbf{m}$ and so satisfies join$(\mathbf{m})$ according to our hypothesis (given relativization).

Now suppose that $\mathbf{a} < \mathbf{0}'$. Then $\mathbf{a}$ may not satisfy mincup, but if it does then let $\mathbf{b} \geq \mathbf{a}$ be GL_1 (such a degree will exist by Theorem 7.4), and let $\mathbf{m}$ be a minimal degree such that $\mathbf{m} \vee \mathbf{a} = \mathbf{b}$. Then $\mathbf{m} \vee \mathbf{0}' \geq \mathbf{b} \vee \mathbf{0}' = \mathbf{b}'$, so $\mathbf{m} \vee \mathbf{0}' = \mathbf{b} \vee \mathbf{0}' = \mathbf{b}' = \mathbf{m}'$, so that $\mathbf{b}$ is actually low over $\mathbf{m}$. By Corollary 13.1, above every low degree there exists a low degree which doesn't satisfy join. This proof relativizes, meaning that we may choose $\mathbf{c} \geq \mathbf{b}$ which doesn't satisfy join$(\mathbf{m})$.

One may reasonably ask how useful this observation actually is, given that we are supposing something that may well not be true – it could well be the case that there are degrees which bound a high degree and which do not satisfy the join property. The point here, though, is that we don't have to consider just the join property. Any property which suffices to separate high and low cones will suffice:

Question 13.3 *Can we find a natural order theoretic property P, which suffices to separate the high and low cones, in the sense that above any low degree there is a degree which does not satisfy P, while any degree which bounds a high degree satisfies P?*

Question 13.4 *If $\mathbf{a} \not\geq \mathbf{0}'$ does there necessarily exist $\mathbf{b} \geq \mathbf{a}$ and a minimal degree $\mathbf{m} < \mathbf{b}$ such that $\mathbf{b}$ is low over $\mathbf{m}$?*

These two questions are interesting because positive answers to both would give a natural definition of the jump (assuming necessary relativizations hold). This follows because then $\mathbf{0}'$ would be the least degree such that:

$$\forall \mathbf{b} \geq \mathbf{a}, \forall \text{ minimal degrees } \mathbf{m} < \mathbf{b},\ \mathbf{b} \in \mathrm{P}(\mathbf{m}),$$

where $P(\mathbf{m})$ denotes P above $\mathbf{m}$, in just the same way that we let join$(\mathbf{m})$ denote join above $\mathbf{m}$ previously.

We finish this section by considering the methods needed to prove Theorem 13.2. We shall not actually prove this theorem, but will prove instead the theorem mentioned previously due to Slaman and Steel, and independently due to Cooper, that there exist non-zero c.e. degrees $\mathbf{b} < \mathbf{a}$ such that no Δ^0_2 degree $\mathbf{d} < \mathbf{a}$ joins $\mathbf{b}$ up to $\mathbf{a}$. This proof illustrates the basic framework which has to be expanded in order to prove Theorem 13.2. We construct c.e. sets B and C such that the following requirements are satisfied:

$\mathscr{P}_i$: If W_i is infinite then $W_i \cap B \neq \emptyset$;
$\mathscr{Q}_i$: If $\Phi_i(B \oplus C)$ $(= D_i$ say) is total and $\Psi_i(B \oplus D_i) = C$ then $\Gamma_i(D_i) = B$;

where $\{(\Phi_i, \Psi_i)\}_{i \in \omega}$ is an effective listing of all pairs of Turing functionals, and each Γ_i is a functional that we build during the course of the construction. B will have infinite complement by construction. The reader is invited to verify for themselves that satisfaction of these requirements suffices to prove the theorem. In what follows we will often abuse notation by writing D_i in order to denote $\Phi_i(B \oplus C)$, even though this may actually be a partial function.

We shall have one strategy for each requirement, and the requirements and their corresponding strategies are prioritized $\mathscr{P}_0, \mathscr{Q}_0, \mathscr{P}_1, \mathscr{Q}_1, \ldots$. At each stage s of the construction we perform another step of the instructions for each of the first s strategies in turn. The construction will have finite injury. More precisely, what this means here, is that each strategy will be *initialized* a finite number of times by higher priority strategies. When a strategy is initialized, this means that it then begins again as if it had never been run before, and in particular that we discard any axioms for Turing functionals that it has previously enumerated. It is only the $\mathscr{Q}_i$ requirements which enumerate axioms for Turing functionals, and each enumerates axioms for a functional that is specific to the strategy, so if the strategy is only initialized finitely many times then the finitely many discarded axioms for that functional are not problematic.

Since the strategies are very simple and the interactions between them are not complicated, it is probably easiest just to describe the strategies directly and then verify that they work.
In what follows we adopt the convention that when discussing any point in the construction, we may write B, C etc in order to denote their present values. At any point during the construction we let δ_i denote the finite string $\Phi_i(B \oplus C)$ (which we can assume to be binary).

The $\mathscr{Q}_i$ strategy. The basic idea behind this strategy is extremely simple – at each stage at which it is run the strategy considers the least n such that $\Gamma_i(\delta_i; n) \uparrow$ and it looks to define this value. There are a few small considerations that have to be made, however.

(a) Whenever we enumerate an axiom defining a value $\Gamma_i(\delta; n)$, we wish to ensure that it is already the case $\Gamma_i(\delta; n') \downarrow$ for all $n' < n$.
(b) When we enumerate such an axiom, we wish to ensure that δ and the present value B already map via Ψ_i to an initial segment of C which is longer than n. The motivation for this is in order to co-ordinate with the $\mathscr{P}$ strategies.
(c) We also have to make sure that the axioms enumerated are consistent. We do not wish to enumerate an axiom $\Gamma_i(\delta; n) = 1$, for example, when there is already some $\delta' \supseteq \delta$ for which we have enumerated the axiom $\Gamma_i(\delta'; n) = 0$. The precise instructions are therefore as below.

At any point in the construction the *length of agreement* for $\mathscr{Q}_i$ is the greatest m such that $C \upharpoonright m \subseteq \Psi_i(B \oplus \delta_i)$. At each stage at which it is run the strategy considers the least n such that $\Gamma_i(\delta_i; n) \uparrow$ and if n is less than the length of agreement then it proceeds as follows. Let β and δ be, respectively, the initial segments of B and δ_i used in the computation $\Psi_i(B \oplus \delta_i; n) = C(n)$. If there exists a shortest δ' with $\delta \subseteq \delta' \subseteq \delta_i$ such that $\Gamma_i(\delta'; n') \downarrow$ for all $n' < n$ and for which it is consistent with all axioms previously enumerated to enumerate the axiom $\Gamma_i(\delta'; n) = \beta(n)$, then enumerate this axiom.

If $\Phi_i(B \oplus C) = D_i$ is total and $\Psi_i(B \oplus D_i) = C$ then it is not hard to see that these instructions ensure that Γ_i is total. In order to ensure that $\Gamma_i(D_i) = B$ we just have to make sure that:

$(\star_i)$ when $n \in B$, D_i does not extend any string δ for which we enumerate an axiom $\Gamma_i(\delta; n) = 0$.

The $\mathscr{P}_i$ strategy. When this strategy enumerates some n into B it must be able to ensure for each $\mathscr{Q}_{i'}$ of higher priority, that if $D_{i'}$ is total and $\Psi_{i'}(B \oplus D_{i'}) = C$, then condition $(\star_{i'})$ is not violated, i.e. $D_{i'}$ cannot extend any string δ for which we have enumerated an axiom $\Gamma_{i'}(\delta; n) = 0$. In order to achieve this the strategy uses a standard

agitator technique, which works as follows. It will not enumerate n into B, unless there exists $\beta = B \upharpoonright n'$ for some $n' < n$ and there exists m such that, for all δ and $i' < i$ for which we have enumerated an axiom $\Gamma_{i'}(\delta;n) = 0$, it is the case $\Psi_{i'}(\beta \oplus \delta;m) = 0$. We call m the agitator for this strategy. Then, upon enumerating n into B, the strategy enumerates m into C and looks to preserve β as an initial segment of B. It attempts to achieve this latter task simply by initializing all lower priority strategies – these strategies will only be allowed to enumerate numbers into B or C which are greater than the last stage at which they were initialized. So long as β is preserved as an initial segment of B, then for each $i' < i$ and each δ for which we have enumerated the axiom $\Gamma_{i'}(\delta;n) = 0$, it *cannot* be the case that $D_{i'}$ is total, $\Psi_{i'}(B \oplus D_{i'}) = C$ and $\delta \subset D_{i'}$ because $\Psi_{i'}(\beta \oplus \delta;m) = 0$, but $m \in C$. Any higher priority strategy $\mathscr{P}_{i''}$ may subsequently enumerate a number into B which is less than $|\beta|$, but then *this* strategy will similarly ensure that no problematic δ can be an initial segment of $D_{i'}$ if it is to be the case that $D_{i'}$ is total and $\Psi_{i'}(B \oplus D_{i'}) = C$ (and if $i'' > i'$), since it will also enumerate some m' into C and preserve a (shorter) initial segment of B. The instructions for the strategy are as follows.

The instructions for $\mathscr{P}_i$. When the strategy is first run (subsequent to any initialization) it chooses a large agitator m and then a large *marker* $n > m$. At every subsequent stage s at which it has not yet been declared satisfied it performs the following steps:

1. Check to see whether there exist any $i' < i$ for which the strategy has not previously *seen convergence* (this will initially be all $i' < i$) and for which $\Gamma_{i'}(\delta_{i'};m)\downarrow$. If so then initialize all lower priority strategies, redefine n to be large, declare that the strategy has seen convergence for each $i' < i$ for which $\Gamma_{i'}(\delta_{i'};m)\downarrow$, and perform no further action at this stage. If not then proceed to (2).
2. If there exists n' such that $n < n'$ and $n' \in W_{i,s} - W_{i,s-1}$, then enumerate n' into B, enumerate m into C, declare the strategy to be satisfied and initialize all lower priority strategies.

The verification. It is clear that each strategy is initialized a finite number of times, and since each marker for a strategy $\mathscr{P}_i$ is only redefined a finite number of times it follows that each $\mathscr{P}_i$ requirement is satisfied. That the complement of B is infinite follows since markers are chosen to be large. In order to show that the $\mathscr{Q}_i$ requirement is satisfied, let s_0 be such that the $\mathscr{Q}_i$ strategy is never initialized after stage s_0. It suffices to show that if D_i is total and the length of agreement is unbounded, then D_i is not compatible with any δ for which we enumerate an axiom subsequent to stage s_0, of the form $\Gamma_i(\delta;n) = 0$ for some n which is enumerated into B. This suffices, because then the instructions for $\mathscr{Q}_i$ clearly ensure that $\Gamma_i(D_i)$ is total.

So suppose that n is enumerated into B by a strategy $\mathscr{P}_{i'}$ for $i' > i$ at a stage $s_1 > s_0$ and that, prior to this enumeration, we have enumerated an axiom $\Gamma_i(\delta;n) = 0$. Let i'' be the least such that $\mathscr{P}_{i''}$ enumerates a number into B at a stage $s_2 \geq s_1$, and such that $\mathscr{P}_{i''}$ is not initialized at any stage in the interval $[s_1,s_2]$ (so that $i'' \leq i'$). Then $\mathscr{P}_{i''}$ has already seen convergence for i when the axiom $\Gamma_i(\delta;n) = 0$ is enumerated, since its agitator m is less than or equal to that of $\mathscr{P}_{i'}$. Let β and δ' be, respectively, the initial segments of B and δ_i used in the computation $\Psi_i(B \oplus \delta_i;m)$ when $\mathscr{P}_{i''}$ saw convergence for i. Since $\mathscr{P}_{i''}$ initialized all lower priority strategies when it saw convergence for i, $\delta' \subseteq \delta$, and β is an initial segment of the final value B – in order to see this note that δ_i cannot change below any given length unless B or C change below the relevant use and that once lower priority strategies are initialized, they will subsequently choose markers and agitators larger than all previously observed uses. It follows that if the length of agreement is unbounded then δ cannot be an initial segment of D_i, because $\Psi_i(\beta \oplus \delta';m) = 0$ and $\mathscr{P}_{i''}$ enumerates m into C.

14. Degrees which bound minimal degrees

Cooper showed that every high degree bounds a minimal, and this was extended by Jockusch who showed that all GH_1 degrees bound minimal degrees.

Theorem 14.1 ([CJ]) *Every generalized high degree bounds a minimal degree.*

Proof. The proof is a modification of the proof that there exists a minimal degree below $\mathbf{0}'$, which makes use of the recursion theorem.

Theorem 14.2 ***The recursion theorem***. *If f is computable then there exists i such that $\Psi_i(\emptyset) = \Psi_{f(i)}(\emptyset)$.*

The main use we make of the recursion theorem is that it allows us to perform the following trick. If we define a construction which takes any index i, and produces a computable function $\Psi_{f(i)}$ (or a c.e. set $W_{f(i)}$) then, in fact, for the right choice of i, we shall have that $\Psi_{f(i)} = \Psi_i$ ($W_i = W_{f(i)}$). So long as we don't make any assumptions about i in advance – so long as we don't assume while defining $\Psi_{f(i)}$ for example, that Ψ_i is total – we may assume given an index for the partial computable function or the c.e. set that we are constructing *in advance*.

Posner's trick. There is another trick which we can make use of when defining minimal degree constructions, which is due to Posner, and which means that generally speaking we do not need to explicitly work in order to satisfy non-computability requirements. Suppose that we construct a set B so that for every i, B either lies on a Ψ_i-splitting tree, or else lies on a Ψ_i-nonsplitting tree. Now suppose that B is computable and consider Ψ_i which is defined as follows; $\Psi_i(\sigma) = \emptyset$ if $\sigma \subset B$, $\Psi_i(\sigma) = \sigma$ otherwise. Since it is not possible for B to lie on a Ψ_i-nonsplitting tree, it must lie on a Ψ_i-splitting tree, which gives a contradiction.

Now suppose that A is GH_1. We are going to construct a set $B \leq_T A$, so by the recursion theorem (which relativizes) we may assume given d such that $B = \Psi_d(A)$. Now since A is GH_1, it can approximate $(B \oplus \emptyset')'$. This means we can assume given a function f such that, for all i,j, $\lim_s f(i,j,s) = 1$ if there exists a Ψ_i-splitting above every initial segment of B amongst the strings in W_j, and is equal to 0 otherwise.

Now recall the construction of a set B of minimal degree below $\mathbf{0}'$. At each stage $s+1$ we are given β_s which is the initial segment of B constructed so far, together with a nested sequence of trees $T^s_0 \supseteq \cdots \supseteq T^s_k$. Since we have an oracle for $\emptyset'$ we can ask whether or not there exists at least one more pair of strings extending β_s in each of these trees. For the least k' for which this is not the case (if there exists such) we change our mind about how $T_{k'}$ should be defined – we define $T^{s+1}_{k'}$ to be some subtree of $T^s_{k'-1}$.

Now that we don't have an oracle for $\emptyset'$ we cannot tell for sure whether or not there exist any strings in each of these trees extending β_s. What we can do, however, is to keep searching for such strings until the function f gives us new evidence that there doesn't exist any such pair. For a given tree, this may cause us to stop searching too early sometimes, but once all trees of higher priority settle down and then the corresponding approximation given by f settles down also, we will not stop searching too early while searching for splittings in this tree anymore. Thus, it will easily follow by induction that the approximation to each tree settles and is defined correctly. The precise construction is as follows.

Stage 0. Define $\beta_0 = \emptyset$ and define T_0 to be the identity function $2^{<\omega} \to 2^{<\omega}$.
Stage $s+1$. We are given β_s and a nested sequence of trees:

$$T^s_0 \supseteq T^s_1 \supseteq \cdots \supseteq T^s_k,$$

together with an index i_j for each $j \leq k$ which specifies the range of T^s_j as a c.e. set of strings. First of all we have to check that there isn't new evidence that we stopped searching too early for one of the trees at a previous stage. We shall define precisely what it means for some $j \leq k$ to *require attention* below, but basically this will just mean that there is now evidence that we stopped searching too early for splittings in the corresponding tree at some earlier stage, because a splitting has now been found. If there is some least $j \leq k$ which requires attention, then proceed as follows:

- Define T^{s+1}_j to be the Ψ_{j-1}-splitting subtree of T^s_{j-1} above β_s.
- Define $T^{s+1}_{j'} = T^s_{j'}$ for all $j' < j$.

- Make $T_{j'}^{s+1}$ undefined for all $j' > j$.
- Declare that j has *received attention*.
- Define $\beta_{s+1} = \beta_s$.

If no $j \leq k$ required attention then for each $0 < j \leq k$ such that T_j^s is not defined simply to be the subtree of the previous tree in the sequence above a certain string, search until either:

Case 1: a pair of strings is found in T_j^s extending β_s, or else;

Case 2: we find $s' > s$ such that $f(j-1, i_{j-1}, s') = 0$.

If there exists a least j such that case 2 applies then define T_j^{s+1} to be the subtree of T_{j-1}^s above β_s. Define $T_{j'}^{s+1} = T_{j'}^s$ for all $j' < j$ and make $T_{j'}^{s+1}$ undefined for all $j' > j$. Define β_{s+1} to be a proper extension of β_s in T_j^{s+1}. We say that j *requires attention* at any stage $s' > s+1$ if $T_{j-1}^{s''} = T_{j-1}^s$ for all s'' with $s < s'' \leq s'$, j has not received attention at any of these stages s'', and there exists a Ψ_{j-1} splitting with both strings extending β_s which is found within s' steps of some fixed effective search procedure.

Otherwise define $T_j^{s+1} = T_j^s$ for all $j \leq k$, define β_{s+1} to be a proper extension of β_s in T_k^{s+1}, and define T_{k+1}^{s+1} to be the Ψ_k-splitting subtree of T_k^{s+1} above β_{s+1}.

The reader is invited to verify the construction. Note that Posner's trick can be used in order to help verify that the set constructed is of minimal degree. □

The fact that this result is sharp in terms of the jump hierarchy follows from the following result of Lerman's.

Theorem 14.3 *[ML2] There exists a high$_2$ degree which doesn't bound any minimal degrees.*

15. The joins of minimal degrees

Which degrees are the join of two minimal degrees? The strongest possible result in terms of the jump hierarchy is the following, which settles a conjecture of Posner's from the 70's.

Theorem 15.1 ([EL2]) *Every generalized high degree is the join of two minimal degrees.*

In order to demonstrate the basic idea behind the proof, we give here a proof of the weaker result, due to Posner, that all degrees above $\mathbf{0}'$ are the join of two minimal degrees.

Proof. In this context it is useful to think of trees as sets of strings rather than functions from strings to strings. It is also useful to restrict our attention to trees of a very particular kind. So, for the duration of this proof, we say that T is a *c.e. Ψ-splitting tree* if every pair of incompatible strings in T are Ψ-splitting and T has a computable enumeration $\{T_s\}_{s\in\omega}$ such that:

Conv A: $|T_0| = 1$ and each $T_{s+1} - T_s$ is finite and consists only of strings extending leaves of T_s.

Conv B: If any string in T has successors in T, then it has precisely three. Three strings of this kind, all of which are successors to the same string, will be called a splitting triple.

Corresponding to each Ψ-splitting tree that we consider, we shall assume that there is some fixed computable enumeration satisfying Conv A and Conv B. If T is any tree which is computably enumerated according to Conv A, and if $\sigma \in T$, then by the *c.e. Ψ-splitting subtree* of T above σ, we just mean any (canonically chosen) c.e. Ψ-splitting tree $T' \subseteq T$ which has σ as the unique string of level 0 and which is maximal, in the sense that if $\tau \in T'$ and there exist three extensions of τ in T every pair of which are a Ψ-splitting, then τ has successors in T'. We assume that any string $\tau \in T'$ is enumerated into this tree at a stage $s \geq s'$, where s' is the stage at which τ is enumerated into T.

We shall use a notion of thin subtrees, which works very much like in the proof of Theorem 11.2. If T is a c.e. Ψ-splitting tree, then the *thin subtree* of T above $\tau \in T$ is the smallest T' satisfying:

(i) $\tau \in T'$;
(ii) if $\tau' \in T'$ and is of even level in T', then its leftmost successor in T (if there exists such) is in T';
(iii) if $\tau' \in T'$ and is of odd level in T', then every successor of τ' in T is in T'.

We assume T' to be given the computable enumeration in which each string in T' is enumerated into this tree at the same stage it is enumerated into T.

Recall that , for any two finite binary strings τ and τ', if there exists some least n such that $\tau(n)\downarrow \neq \tau'(n)\downarrow$, then we say that τ is to the left of τ' (and that τ' is to the right of τ) if $\tau(n)=0$. If T is tree and $\tau \in T$ has three successors in T, then we shall refer to these three successors as the *leftmost*, the *second* and the *rightmost* successor respectively when ordered from leftmost to rightmost. We write λ to denote the string of length 0. By a 3-fold Ψ-splitting, we mean three strings, every pair of which are Ψ-splitting. From this point on, for the duration of this proof, by a splitting tree we shall mean a tree which is a c.e. Ψ-splitting tree for some Ψ, and by a splitting we shall actually mean a 3-fold splitting—but no confusion will result from these abuses of terminology. We assume the full binary tree to be given the enumeration in which all strings of length s are enumerated at stage s.

15.1. The basic idea

Suppose we are given C of degree above $\mathbf{0}'$, and consider running Sacks' oracle construction of a set A of minimal degree below $\mathbf{0}'$, as described in the proof of Theorem 5.3, but using an oracle for C and using splitting trees of the modified form just described in which each string in the tree has three successors if any, rather than just two. At each stage we are given a sequence of trees $T_0,\cdots,T_i$ and a finite binary string α which is the initial segment of A that we have constructed so far. We find the greatest $j \leq i$ such that there exists a splitting triple in T_j above α, and then redefine α to be the leftmost string in this splitting triple, before defining the sequence of trees to be considered at the next stage of the construction.

Of course, we did not *have* to redefine α to be the leftmost string in the splitting. We could consider trying to code C into A at stage $s+1$ by redefining α to be the second string if $C(s)=0$ and the rightmost string otherwise. A constructed in this way would fail to compute C *because* it is unable to retrace the construction to see how the sequence of trees is defined at each stage. We shall show how to define two sets A and B in this way, however, so that $A \oplus B$ is able to compute the sequence of trees defined at each stage of the construction, and thereby compute C.

The basic idea is very simple. We shall construct a sequence of trees $T_0,T_1,\cdots$ such that A is a path through T_j when j is even, and B is a path through T_j when j is odd. For each j we shall also consider a tree S_j which will be a thin subtree of T_j. Suppose for a moment that j is even—the following discussion will also hold for odd j, but with B in place of A. Initially we try to construct A lying on S_j. T_{j+2}, then, is constructed as a splitting subtree of S_j. When we find at some stage that we must redefine T_{j+1}, however, we code this fact by forcing A to "step off" S_j (i.e. we decide that A should be a path through T_j which is not a path through S_j). If this happens at stage s and $A \oplus B$ has already been able to compute the set of trees that we are working with at stage s, then it will be able to see that A has stepped off S_j, and so will be able to discern how we redefined the trees at this stage of the construction.

Along these lines, the following terminology will be useful. Suppose that τ is a string of even level in S_j. Then we call the splitting triple in T_j whose strings are successors of τ (should such a triple exist), a *coding opportunity*. The second and rightmost strings of the coding opportunity are referred to as the *coding* strings. We shall define α_s and β_s at each stage s, and ultimately we define $A=\bigcup_s \alpha_s$ and $B=\bigcup_s \beta_s$. At every stage $s>0$ of the construction we shall find a coding opportunity in some T_j and define either α_s or β_s (depending on whether j is even or odd) to be one of the strings in this coding opportunity. Note that the set of coding opportunities in T_j depends also on S_j, which may be redefined at stages of the construction when T_j is not.

So we start with the following sort of procedure in mind. At the beginning of stage $s+1$ we are given $T_0,\cdots,T_i$, $S_0,\cdots,S_i$, α_s and β_s. Initially we define $\alpha=\alpha_s$ and $\beta=\beta_s$. These values α and β may be redefined a finite number of times during stage $s+1$, to be extensions of their previous values. At the end of the stage we will define $\alpha_{s+1}=\alpha$

and $\beta_{s+1} = \beta$. At stage $s+1$ we perform a finite iteration which moves down through the sequence of trees, starting with T_i and continuing until we find some T_j with a coding opportunity that can be used at this stage. Suppose for now that i is even.

We ask, does there exist a splitting triple above α in T_i? If so, there are now two subcases. If this splitting triple is not a coding opportunity in T_i, then we redefine α to be the leftmost string in the triple and go back to asking whether there exists a splitting triple in T_i above α. Otherwise, redefine α to be the second string in the coding opportunity if $C(s) = 0$ and the rightmost string otherwise, before redefining S_i, T_{i+1}, S_{i+1} and terminating stage $s+1$ of the construction. The use of this coding opportunity codes the fact that we did not have to redefine any of $T_0, \cdots, T_i$ at this stage.

If not, then let $j \leq i$ be the least which is even and such that there doesn't exist a splitting triple above α in T_j. We now try to code the fact that T_j must be redefined. In order to do so, we begin the iteration again but with T_{j-1} and β in place of T_i and α (and with "odd" in place of "even").

15.2. Further considerations

When $A \oplus B$ retraces the construction at stage $s+1$, having already computed the set of trees that we are working with at this stage together with α_s and β_s which are the initial segments of A and B which have been decided by the end of stage s, it will continue to enumerate the trees until it sees either A or B step off one of the S_j. We must ensure, however, that the first coding opportunity which appears in this way, with either A or B extending one of its coding strings as appropriate, is actually the coding opportunity that is used at stage $s+1$ and not one that is used at some subsequent stage. If we proceed simply as described in 15.1 it remains possible that at stage $s+1$ we use a coding opportunity in T_i, which is enumerated into this tree at stage t (say), and that at the next stage we use a coding opportunity in T_{i-1} which is enumerated into this tree at a stage $t' < t$. In this case $A \oplus B$ will retrace the construction incorrectly. For this reason we define a value t_s at each stage s, which is the stage at which the coding opportunity we use at stage s is enumerated into the respective tree. At stage $s+1$ we are restricted to using coding opportunities which are enumerated into the relevant tree at a stage $> t_s$. The following terminology is useful in this context. If T is a tree with a given computable enumeration, then for any $\tau \in T$, $t(T, \tau)$ is the stage at which τ is enumerated into T. We also use the variable ρ to range over the set of splitting triples, and let $t(T, \rho)$ be the stage at which ρ is enumerated into T.

Perhaps some words of caution are useful here. In the proofs of some previous theorems in this course and in the proof that follows, we often use computer science type conventions of choosing variables, and then allowing these variables to be redefined multiple times without any corresponding indication in the notation. The motivation here is that, in doing so, the argument should become much easier to follow – the alternative, indexing variables in terms of the stages and ultimately the sub-stages of the construction, would result in heavy notation. The danger is that ambiguity might result from referring to variables that take multiple values at various stages of the construction. During the construction there will be no ambiguity, because when we refer to the value of any given variable, we simply mean the value as defined *at that point in the construction*. During the verification, ambiguity will not result so long as we are very clear about exactly which point of the construction (and not just the stage, but which point of the stage) is being referred to.

It is also worth pointing out that the stages of the construction and the stages in the enumeration of the trees that are considered at any point of the construction, are treated entirely separately. Thus, at stage 5 we might enumerate T_2 until we see that, at stage 117 in the enumeration of this tree, a splitting triple ρ appears which we can use as a coding opportunity. If we use this coding opportunity at stage 5, then $t_5 = 117$, and, at the end of this stage, if we remark that $t(T_2, \rho) = 117$, then the value T_2 being referred to here is the value at the end of stage 5.

There is another (fairly trivial) way in which our construction will deviate from the Sacks construction. In that construction, when we find at stage $s+1$ that j is the least such that there do not exist any proper extensions of α_s in T_j, we then redefine this tree to be the subtree of T_{j-1} above α_{s+1} (i.e. the set of all strings in T_{j-1} which extend α_{s+1}). The redefined tree T_j, however, does not subsequently play any vital role in the construction. If T_j was

defined to be a Ψ_k-splitting tree at the end of stage s, then at stage $s+1$ we could alternatively just redefine T_j to be the Ψ_{k+1}-splitting subtree of T_{j-1} above α_{s+1}. It will be technically much more convenient to follow this approach here, and so we shall do so (although now that we are constructing *two* sets and since we are using thin subtrees, T_j will be defined to be a splitting subtree of S_{j-2} rather than T_{j-1}).

Before describing the construction precisely, let us describe in outline how $A \oplus B$ will be able to retrace it. So suppose that $A \oplus B$ has already been able to decide α_s, β_s, t_s and the sequence of trees $T_0, \cdots, T_i$ which are defined at the end of stage s, together with each S_j for $j \leq i$. In order to compute these values for stage $s+1$, $A \oplus B$ will then enumerate these trees until it sees an initial segment of A which is compatible with a coding string in some T_j such that j is even, or an initial segment of B which is compatible with a coding string in some T_j such that j is odd (and then it will consider the first such coding string which appears). Suppose for now, that the first such coding string τ is a coding string in T_j, and that j is even. Then α_{s+1} is that coding string, and we define the construction so as to ensure that β_{s+1} is the longest initial segment of B which is enumerated into T_{j-1} at a stage $\leq t(T_j, \tau)$.

15.3. The construction

We proceed in stages as follows.

Stage 0. It is convenient to assume that Ψ_0 is the identity functional $2^{<\omega} \to 2^{<\omega}$. Define T_0 and T_1 to be the Ψ_0-splitting subtree of the full binary tree above λ, and define S_0 and S_1 to be the thin subtree of T_0 above λ. Define $\alpha_0 = \beta_0 = \lambda$ and $t_0 = 0$.

Stage $s+1$. Let i be the greatest such that T_i is defined. We run a finite iteration which will terminate at the first step $n+1$ at which we find an appropriate opportunity to code $C(s)$. At step n in this iteration we define a value j_n such that $j_0 = i$ and each $j_{n+1} \leq j_n$. Initially we have $\alpha = \alpha_s$ and $\beta = \beta_s$ and we may redefine these values to be extensions of their previous values at various stages in the iteration.

Step 0. Define $j_0 = i$, $\alpha = \alpha_s$ and $\beta = \beta_s$.

Step $n+1$. We ask, does there exist a splitting triple in T_{j_n}, above α if j_n is even and above β if j_n is odd? If not, then let j be the least $\leq j_n$, which is even if j_n is even and odd if j_n is odd, such that there does not exist such a splitting triple in T_j (the way in which we define T_0 and T_1 means that $j \geq 2$). Put $j_{n+1} = j-1$ and perform step $n+2$.

Otherwise, consider the first such splitting triple ρ enumerated into T_{j_n}. Now there are two cases to consider.

Case (a): ρ is not a coding opportunity or else $t(T_{j_n}, \rho) \leq \max\{t_s, x\}$, where $x = t(T_{j_n-1}, \alpha)$ if j_n is odd and $x = t(T_{j_n-1}, \beta)$ otherwise. Then redefine α to be the leftmost string in the splitting triple if j_n is even, and redefine β to be the leftmost string in the splitting triple if j_n is odd. Put $j_{n+1} = j_n$ and perform step $n+2$.

Case (b): otherwise. Define $t_{s+1} = t(T_{j_n}, \rho)$. First we code $C(s)$.

If j_n is even then define α_{s+1} to be the second string in the splitting triple if $C(s) = 0$ and the rightmost string otherwise. In this case, define β_{s+1} to be the longest of all those leftmost extensions of β in S_{j_n-1} which are enumerated into this tree at a stage $\leq t_{s+1}$. If j_n is odd then define β_{s+1} to be the second string in the splitting triple if $C(s) = 0$ and the rightmost string otherwise. In this case, define α_{s+1} to be the longest of all those leftmost extensions of α in S_{j_n-1} which are enumerated into this tree at a stage $\leq t_{s+1}$.

Next we must redefine the trees. Let $j = j_n$.

- If $j = i$, with i defined as at the beginning of stage $s+1$, then suppose that T_{i-1} is presently defined to be a Ψ_k-splitting tree. Define T_{i+1} to be the Ψ_{k+1}-splitting subtree of S_{i-1} above β_{s+1} if i is even, and define T_{i+1} to be the Ψ_{k+1}-splitting subtree of S_{i-1} above α_{s+1} if i is odd.
- If $j < i$ then suppose that T_{j+1} was defined to be a Ψ_k-splitting tree at the end of stage s. Redefine T_{j+1} to be the Ψ_{k+1}-splitting subtree of S_{j-1}, above β_{s+1} if j is even or above α_{s+1} if j is odd.
- Redefine S_j to be the thin subtree of T_j, above β_{s+1} if j is odd or above α_{s+1} if j is even. Define S_{j+1} to be the thin subtree of T_{j+1}, above β_{s+1} if j is even or above α_{s+1} if j is odd. Make $T_{i'}$ and $S_{i'}$ undefined for all $i' > j+1$.
- Terminate the iteration and stage $s+1$ of the construction.

It is not difficult to show that $A = \bigcup_s \alpha_s$ and $B = \bigcup_s \beta_s$ are total and are of minimal degree. Now suppose that the oracle $A \oplus B$ has already been able to compute α_s, β_s, t_s and the sequence of trees $T_0, \cdots, T_i$ which are defined at the end of stage s, together with each S_j for $j \leq i$. In order to conclude that $A \oplus B$ can compute these values for stage $s+1$, it suffices to show that the coding opportunity we use at this stage is the first enumerated into any $T_j \in \{T_0, \cdots, T_i\}$ such that either A extends one of the coding strings and j is even or B extends one of the coding strings and j is odd. In order to see this, suppose that the iteration terminates at step $n+1$ at stage $s+1$ and let $j = j_n$, as defined at that stage. We consider each of the trees T_k such that $k \neq j$.

First consider $k > j$. Suppose that k is even, a similar argument will apply when k is odd. Let m be the greatest such that $j_m \geq k$, and let $\sigma = \alpha$ as defined at the beginning of step $m+1$ (so that σ is a string in S_k). If j_m is even then our action at step $m+1$ determines that there is no splitting triple in T_k above σ. So suppose otherwise. Then by induction on the steps after m, at the beginning of step $n+1$, for all strings $\tau \in T_k$ extending σ, α is either an initial segment of τ or lies to the left of τ. If j is even, then at step $n+1$ we define α_{s+1} to be incompatible with any coding strings in T_k. If j is odd, then at step $n+1$ we define α_{s+1} to lie to the left of any coding strings in T_k above σ which are enumerated into this tree at a stage $\leq t_{s+1}$.

Next consider $k < j$. If there does not exist any subsequent stage at which we elect to use a coding opportunity in some $T_{k'}$ for $k' \leq k$ then A lies on S_k if k is even and B lies on S_k if k is odd. So suppose otherwise and consider the first such stage s'. If $k' = k$ then the coding opportunity ρ in T_k which we use at stage s', has $t(T_k, \rho) > t_{s+1}$. If $k' < k$ then precisely the argument we used above for the case $k > j$, suffices to show that A does not extend any coding strings in T_k which are enumerated into this tree at a stage $\leq t_{s'}$ if k is even, and B does not extend any coding strings in T_k which are enumerated into this tree at a stage $\leq t_{s'}$ if k is odd. Since $t_{s'} > t_{s+1}$, the claim follows. □

Posner originally asked whether $\mathbf{0}'$ could be defined as the least degree such that all degrees above are the join of two minimal degrees. This was settled negatively by Shore, but the following question remains open:

Question 15.1 *Is $\mathbf{0}'$ minimal amongst the degrees such that all degrees above are the join of two minimal degrees?*

16. The minimal cupping property

We gave the definition earlier:

Definition 16.1 *A degree* $\mathbf{a}$ *satisfies the minimal cupping property (mincup) if for all* $\mathbf{b} \geq \mathbf{a}$ *there exists a minimal degree* $\mathbf{m} < \mathbf{b}$ *with* $\mathbf{a} \vee \mathbf{m} = \mathbf{b}$.

Let us consider the situation below $\mathbf{0}'$ first.

Theorem 16.1 ([EL2]) *In $\mathscr{D}[\leq \mathbf{0}']$, all high degrees satisfy mincup.*

This result is clearly sharp in terms of the jump hierarchy – again this follows from Lerman's result [ML2] that there exist high$_2$ degrees which don't bound minimal degrees.

Conjecture 16.1 *In $\mathscr{D}[\leq \mathbf{0}']$, a c.e. degree is high iff it satisfies mincup.*

If true this would give us a nice characterization of the high c.e. degrees (and a natural definition of the high c.e. degrees in terms of a definition for the c.e. degrees in $\mathscr{D}[\leq \mathbf{0}']$). Unfortunately it certainly can't be extended to give us a natural definition of the high degrees:

Theorem 16.2 ([EL2]) *There exists a low degree which satisfies mincup.*

Note that this theorem holds globally. In order to give this result, the approach taken is to establish the existence of a perfect tree which is of low degree, and such that every path through the tree is of minimal degree.

The first remaining questions here, also concern the global structure.

Question 16.1 *Do all GH_1 degrees satisfy mincup?*

17. The meet and complementation properties

We next consider the meet and complementation properties. The meet property was defined previously.

Definition 17.1 *A degree $\boldsymbol{a}$ satisfies the complementation property if, for all non-zero $\boldsymbol{b} < \boldsymbol{a}$, there exists $\boldsymbol{c} < \boldsymbol{a}$ such that $\boldsymbol{b} \wedge \boldsymbol{c} = \boldsymbol{0}$ and $\boldsymbol{b} \vee \boldsymbol{c} = \boldsymbol{a}$.*

First of all, let us establish the uninteresting cases – initial segment results can be used in order to show that all possibilities can be realized for the low and for the low_2 non-low degrees. On the positive side, Posner [DP2] showed that $\boldsymbol{0}'$ satisfies the complementation property using a non-uniform proof. This non-uniformity was subsequently shown not to be necessary by Slaman and Steel [SS]. The following is the strongest result known:

Theorem 17.1 ([GMS]) *All generalized high degrees satisfy the complementation property.*

The proof given for this stronger result is once again non-uniform, and it remains open as to whether the complement here can be constructed uniformly. Recently, James Riley has shown that this result fails for the high_2 degrees:

Theorem 17.2 ([JR]) *There exists a high_2 degree which does not satisfy the meet property.*

The following theorem settles a conjecture of Cooper's from the 80s:

Theorem 17.3 ([DLNR]) *All c.e. degrees satisfy the meet property i.e. if $\boldsymbol{a}$ is c.e. then for* all *$\boldsymbol{b} < \boldsymbol{a}$, there exists non-zero $\boldsymbol{c} < \boldsymbol{a}$ (which is not necessarily c.e.) such that $\boldsymbol{b} \wedge \boldsymbol{c} = \boldsymbol{0}$.*

The following question was asked by Slaman and Steel:

Question 17.1 ([SS]) *Can $\boldsymbol{0}'$ be defined as the least degree such that all degrees above satisfy complementation?*

As with Question 13.2, the expectation is presumably that this question will eventually be answered in the negative, but that the counter example may be difficult to construct.

18. The capping property

In this section, we briefly consider a property about which little is known:

Definition 18.1 *A degree $\boldsymbol{a}$ satisfies the capping property if, for all $\boldsymbol{b} > \boldsymbol{a}$ there exists non-zero $\boldsymbol{c} < \boldsymbol{b}$ such that $\boldsymbol{a} \wedge \boldsymbol{c} = \boldsymbol{0}$.*

It follows from Theorem 16.2, that there exists a low degree such that all degrees above satisfy the capping property. This suffices to show that there exist members of each jump class which satisfy the capping property. It also follows from Theorem 16.1 that, in $\mathscr{D}[\leq \boldsymbol{0}']$, all high degrees satisfy the capping property. Nothing else substantial is known at this point.

Question 18.1 *Do all GH_1 degrees satisfy the capping property?*

19. The minimal complementation property

In this final section, we consider the most complex property so far:

Definition 19.1 *A degree $\boldsymbol{a}$ satisfies the minimal complementation property if, for all non-zero $\boldsymbol{b} < \boldsymbol{a}$, there exists a minimal degree $\boldsymbol{m} < \boldsymbol{a}$ such that $\boldsymbol{b} \vee \boldsymbol{m} = \boldsymbol{a}$.*

The results here are as follows:

Theorem 19.1 (Lewis, Seetapun, Slaman [AL5]) *$\mathbf{0}'$ satisfies the minimal complementation property.*

Theorem 19.2 ([AL6]) *All degrees above $\mathbf{0}'$ satisfy the minimal complementation property.*

In fact, one can complement quite a lot of degrees simultaneously:

Theorem 19.3 ([AL7]) *There exists a minimal degree below $\mathbf{0}'$ which complements all (non-zero, incomplete) c.e. degrees.*

While it is immediately clear that no Δ^0_2 degree can complement all non-zero and incomplete Δ^0_2 degrees, one might hope that *some* version of this theorem might hold for the Δ^0_2 degrees in general – there might be two degrees such that each non-zero, incomplete degree is complemented by at least one of them, for example. In fact, however, this fails very strongly:

Theorem 19.4 ([AL8]) *For every degree $\boldsymbol{c}$ with $\mathbf{0} < \boldsymbol{c} \leq \mathbf{0}'$ and every uniformly Δ^0_2 sequence of degrees $\{\boldsymbol{b}_i\}_{i\geq 0}$ such that, for all i, $\boldsymbol{b}_i \not\geq \boldsymbol{c}$, there exists $\boldsymbol{a}$ with $\mathbf{0} < \boldsymbol{a} < \mathbf{0}'$ such that $\boldsymbol{a} \vee \boldsymbol{b}_i \not\geq \boldsymbol{c}$ for all i.*

The first remaining questions are these:

Question 19.1 *Do all/any incomplete high degrees satisfy the minimal complementation property?*

Question 19.2 *Can $\mathbf{0}'$ be defined as the least degree such that all degrees above satisfy minimal complementation?*

References

[BC] Cooper,S.B., Distinguishing the arithmetical hierarchy, preprint, Berkeley, 1972.

[BC2] Cooper,S.B., *Computability Theory*, Chapman & Hall, 2004.

[BC3] Cooper,S.B., Degrees of unsolvability complementary between recusively enumerable degrees, *Annals of Mathematical Logic*, 4 (1), 31-74, 1972.

[BC4] Cooper,S.B., The strong anticupping property for recursively enumerable degrees, *Journal of Symbolic Logic*, vol. 54, 527-539, 1989.

[DGLM] Downey,R., Greenberg,N., Lewis,A.E.M., Montalbán,A., Extensions of embeddings below computably enumerable degrees, *Transactions of the American Mathematical Society*, 365, 2977-3018, 2013.

[DH] Downey,R., Hirshfeldt,D., *Algorithmic Randomness and Complexity*, Springer-Verlag, 2010.

[DJS] Downey,R., Jockusch,C.G. and Stob,M., Array nonrecursive degrees and genericity, *Computability, Enumerability, Unsolvability*, vol. 224, London Mathematical Society Lecture Note Series, 93-104, 1996.

[DLNR] Durrant,B., Lewis-Pye,A., Ng, S,, and Riley,R., C.e. degrees and the meet property, *Proceedings of the American Mathematical Society*, 144, 1735-1744, 2016.

[EL] Ellison,P., Lewis,A.E.M., Joining up to the generalized high degrees, *Proceeding of the American Mathematical Society*, 138, 2949-2960, 2010.

[EL2] Ellison,P., Lewis,A.E.M., The minimal cupping property, to appear.

[GMS] Greenberg,N., Montalbán,A. and Shore,R.A., Generalized high degrees have the complementation property, *Journal of Symbolic Logic*, 69, 1200-1220, 2004.

[IS] Ishmukhametov,S. (1999) Weak recursive degrees and a problem of Spector, *Re- cursion Theory and Complexity*, Arslanov and Lempp eds., de Gruyter, 81-89.

[CJ] Jockusch,C.G., Simple proofs of some theorems on high degrees of unsolvability, *Canadian Journal of Mathematics*, 29, (1977), 1072-1080.

[Jsh] Jockusch,C.G., Shore,R., Pseudo jump operators I: the R.E. case, *Transactions of the American Mathematical Society*, 275, (1983), 599-609.

[JP] Joskusch,C.G., and Posner,D.B., Double jumps of minimal degrees, *Journal of Symbolic Logic*, 43, 715-724, 1978.

[JS] Jockusch,C.G., and Soare,R., Π^0_1 classes and degrees of theories, *Transactions of the American Mathematical Society*, 173, pp. 33-56, 1972.

[KP] Kleene,S.C and Post, E.L., The upper semi-lattice of degrees of recursive unsolvability, *Annals of Mathematics*, 59, (1954), 379-407.

[AK] Kučera,A., Measure, Π^0_1 classes and complete extensions of PA, in *Recursion Theory Week (Proceedings Oberwolfach 1984)*, Lecture Notes in Mathematics, vol 1141, pp. 245-259, 1985.

[AK2] Kučera,A., An alternative priority-free solution to Post's problem, *Twelfth Symposium held in Bratislava, Czechoslovakia, August 25-29*, 1986, edited by J. Gruska, B. Rovan, and J. Weiderman, Lecture Notes in Computer Science No. 233.

[ML] Lerman,M., *Degrees of Unsolvability*, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1983.

[ML2] Lerman,M., Degrees which do not bound minimal degrees, *Annals of Pure and Applied Logic*, 30, 249-276, 1986.

[AL] Lewis,A.E.M., Π^0_1 classes, strong minimal covers and hyperimmune-free degrees, *Bulletin of the London Mathematical Society*, 39, 892-910, 2007.

[AL2] Lewis,A.E.M., Strong minimal covers and a question of Yates: the story so far, *Proceedings of the Logic Colloquium 2006*.

[AL3] Lewis,A.E.M., On a question of Slaman and Groszek, *Proceedings of the American Mathematical Society*, 136, 3663-3668, 2008.

[AL4] Lewis,A.E.M., A note on the join property, *Proceedings of the American Mathematical Society*, 140, 707-714, 2012.

[AL5] Lewis,A.E.M., Minimal complements for degrees below 0', *Journal of Symbolic Logic*, 69 (4), 937-966, 2004.

[AL6] Lewis,A.E.M., The minimal complementation property above 0', *Mathematical Logic Quarterly*, 51 (5), 470-492, 2005.

[AL7] Lewis,A.E.M., A single minimal complement for the c.e. degrees, *Transactions of the American Mathematical Society*, 359, 5817-5865, 2007.

[AL8] Lewis,A.E.M., Finite cupping sets, *Archive for Mathematical Logic*, 43, 845-858, 2004.

[DM] Martin,D., Classes of recursively enumerable sets and degrees of unsolvability, *Zeit. Math. Log. Grund. Math.*, 12, (1966), 295-310.

[AN] Nies,A., *Computability and Randomness*, Oxford University Press, 2009.

[AN2] Nies,A. Coding Methods in Computability Theory and Complexity Theory. Habilitation thesis, Jan 1998.

[NSS] Nies,A., Shore,R.A., and Slaman,T.A., Interpretability and definability in the recursively enumerable degrees, *Proceedings of the London Mathematical Society*, (3), vol. 77, no. 2, 241291, 1998.

[DP] Posner,D., A survey of the non-r.e. degrees $\leq \mathbf{0}'$, *London Mathematical Society Lecture Note Series* 45, Recursion Theory: its Generalisations and Applications, Proceedings of the Logic Colloquium '79, Leeds, edited by F.R. Drake and S.S.Wainer.

[DP2] Posner,D., The uppersemilattice of degrees $\leq \mathbf{0}'$ is complemented, *Journal of Symbolic Logic*, 46, 705-713, 1981.

[PR] Posner,D. and Robinson,R., Degrees joining to $\mathbf{0}'$, *Journal of Symbolic Logic*, 46, 714-722, 1981.

[JR] Riley,J., PhD thesis, University of Leeds, 2016.

[GS] Sacks,G., A minimal degree less than $\mathbf{0}'$, *Bulletin of the American Mathematical Society*, 67, 416-419, 1961.

[GS2] Sacks,G., Recursive enumerability and the jump operator, *Transactions of the American Mathematical Society*, 108, (1963), 223-239.

[LS] Sasso,L., A minimal degree not realizing least possible jump, *Journal of Symbolic Logic*, 39, (1974), 571-574.

[DS] Scott,D., Algebras of sets binumerable in complete extensions of arithmetic, *Proc. Symp. Pure Appl. Math.*, 5, (1962), 117-121.

[JS] Shoenfield,J., A theorem on minimal degrees, *Journal of Symbolic Logic*, 31, 539-544, 1966.

[SSl] Shore,R. and Slaman,T., Defining the Turing jump, *Math. Res. Lett.*, 6(5-6):711722, 1999.

[SH] Shore,R., Natural definability in degree structures, *Computability Theory and Its Applications: Current Trends and Open Problems*, P. Cholak, S. Lempp, M. Lerman and R. A. Shore eds., Contemporary Mathematics, AMS, Providence RI, 255-272, 2000.

[SH2] Shore,R., Direct and local definitions of the Turing jump, *Journal of Mathematical Logic*, 7, 229-262, 2007.

[SS2] Simpson,S., First order theory of the degrees of recursive unsolvability, *Annals of Mathematics*, 105, 121-139, 1977.

[SS] Slaman,T.A. and Steel,J.R., Complementation in the Turing degrees, J. Symbolic Logic, vol. 54 no. 1, pp. 160-176, 1989.

[SW] Slaman,T.A. and Woodin,H., Definability in the Turing degrees, *Illinois Journal of Mathematics*, 30, 320-334, 1986.

[RS] Soare,R., Automorphisms of the lattice of recursively enumerable sets, *Bulletin of the American Mathematical Society*, 80, 53-58, 1974.

[RS2] Soare,R., *Recursively enumerable sets and degrees*, Springer, New York, (1987).

[CS] Spector,C., On degrees of recursive unsolvability, *Annals of Mathematics*, 64, (1956), 581-592.

[TZ] Terwijn,S.A. and Zambella,D., Computational randomness and lowness, *J. Symbolic Logic*, 66, (2001), 1199-1205.