Qual Prep Notes based on Lecture Notes by Joe and Uri

Karthik Ravishankar

September 27, 2022

Contents

1 Introductory Chapter

Basic properties that our ennumeration $\{\varphi_e\}_{e\in\omega}$ of partial computable functions satisfy:
i) **Padding Lemma**: Every φ_e has an infinite set of indices which can be found effectively.
ii) **Ennumeration Lemma**: There is an ennumeration of the partial computable functions i.e. there is a partial computable function of arity 2 such that $\psi(e,n) = \varphi_e(n) \forall e, n$.
iii) **Parameter Lemma/s-m-n theorem**: Inputs can be made into parameters i.e. given a partial computable function $\varphi_e(\bar{y}, \bar{z})$ there is an injective computable function g such that $\varphi_{g(e,\bar{y})}(\bar{z}) = \varphi_e(\bar{y}, \bar{z})$. (by 'hardcoding' the inputs)

Equivalent definitions of *c.e.* sets:
i) $W_e = dom(\varphi_e)$
ii) projection of computable relation - $X = \{x : \exists y R(x,y)\}$ where R is a computable relation.
iii)Range of a total computable function

Theorem: i) There is a Σ_1^0 formula $\psi(e,n,m)$ such that $\varphi_e(n) \downarrow = m \iff (\mathbb{N}, +, ., 0, 1) \models \psi(e,n,m)$
ii) There is a Σ_1^0 formula $\psi(e,n,m)$ such that $\varphi_e(n) \downarrow = m \implies PA \vdash \psi(e,n,m)$ and $PA \vdash \forall e, n \exists! m \psi(e,n,m)$

Formal Recursion Theorem: For a total computable function $f : \omega \to \omega$ there is a fixed point i.e. $\exists e \; \varphi_e = \varphi_{f(e)}$.
Proof Idea: Treat the φ_e's as functions on pairs $< e, n >$ and define a p.r. $\psi(< e, n >)$ whose e^{th} column is the e^{th} column of $\varphi_{f(e)}$ i.e $\psi(< e, . >) = \varphi_{f(e)}(< e, . >)$. If the index of ψ is i, then the i^{th} column of $\psi = \varphi_i$ is equal to the i^{th} column of $\varphi_{f(i)}$ by definition of ψ. Our f is 'fixing' this i^{th} column.
Intutive Recursion Theorem: We can use the index of an algorithm while describing the algorithm.
Proof of Intutive $\implies$ formal: Let the algorithm with index e compute $f(e)$ and simulate $\varphi_{f(e)}$. Then e is a fixed point for f.
Proof of formal $\implies$ intutive: Let a program guess its own index to be n. Then for each n let $\varphi_{f(n)}$ be the p.r function corresponding to the program which guesses its own index as n. If e is a fixed point for f then the program has the correct guess of its index.

Proposition Given a computable $g : \omega \to \omega$ there is an index e such that W_e is computable but the least index of $\overline{W_e}$ is $> g(e)$.

Theorem (Kleene Post): There exists a pair of $0'$ computable sets A, B which are turing incomparable i.e. $A|_T B$.
Proof: Initial segment construction using $0'$.

Theorem (Stronger Kleene Post): GIven $A >_T 0$ there is a $B \leq_T A'$ such that $A|_T B$.
Proof: Initial segment construction using e-splitting (A' oracle required for e-splitting).

2 The Arithmetical Hierarchy

Theorem: i) If B is computable from A then B c.e. $\implies$ A c.e.
ii) $B \leq_T A \iff B' \leq_1 A'$.

Theorem (limit lemma): $B \leq_T A'$ if and only if B can be approximated using A i.e. $\exists f(x,s) \leq_T A$ such that $B(x) = lim_s f(x,s)$ for any x.

Theorem (Post): i) X is $\Sigma^0_{k+1} \iff X \leq_1 0^{(k+1)} \iff X$ is $\Sigma^0_1[0^k]$.
ii) X is $\Delta^0_{k+1} \iff X \leq_T 0^k$.

Remark $Fin - \Sigma^0_2 . Inf, ToT - \Pi^0_2 . Cof - \Sigma^0_3$ complete. (Moveable marker construction for Cof using projection of Inf as our Σ^0_3 set)
Proposition If $g \leq_T 0''$ then there is an e such that W_e is computable and the least index of $\overline{W_e}$ is greater than $g(e)$.
Proof Idea: Show that otherwise Cof is $0''$ computable which violates its Σ^0_3 completeness.

Theorem: There is a non computable low set.
Proof Idea: Initial segment construction using $0'$ oracle and forcing the jump to ensure lowness.
Theorem (Martin's High Domination): A set is high $\iff$ it computes a dominant function (a function which dominates every computable function)
Proof Idea: If A is high there is a $A-$ approximation to $ToT \equiv_T 0''$. Now define $f(s)$ to try to beat all φ_e for $e < s$ which we 'think' are total (think in the sense of our approximation says they are total). Since our approximation converges, we will eventually beat all total functions.
If the set computes a dominant function f, we use f to approximate ToT using the fact that if a function φ_e is total, the function taking n to $\mu s \varphi_{e,s}(n) \downarrow$ is total computable (and so is dominated by f).

3 The Method of Forcing

Theorem (Friedberg Jump Inversion): Given an $S \geq_T 0'$ there is an A such that $A' \equiv_T A \oplus 0' \equiv_T S$.
Proof Idea: Initial segment construction using oracle S. Encode S into A at odd stages and force the jump of A at even stages i.e. try to make $\varphi_e^A(e) \downarrow$ if possible.
Definition: i) A is GL_1 if $A' \leq_T A \oplus 0'$ i.e. it is as low as possible.
ii) Turing degrees a, b form a minimal pair if they are both non computable and their meet is 0.
iii) Given a non ascending chain of Turing degrees $\{d_n\}_{n\in\omega}$, b, c is called an exact pair for the chain if $\forall n\ d_n \leq_T b, c$ and $a \leq b, c \implies \exists n\ a \leq_T d_n$.

Theorem: There exists a $0'$ computable minimal pair a, b.
Proof Idea: At the $2e$ stage, try to make $\varphi_e^A \neq \varphi_e^B$ if possible. Now if $g \leq_T A, B$ and g is total then it will be computable since there was no possible '$e-$ splitting.
Theorem(Exact pairs) Every non ascending chain of Turing degrees has an exact pair.
Proof idea: Make $d_n \leq b$ by in the $2n+1$ stage filling in the as yet undefined entries in the nth column of b whith values of d_n. (Only finitely many entries are filled in already so d_n can be computed by the nth column).
Make $\varphi_n^b \neq \varphi_n^c$ if possible in the $2n$th stage. This will ensure that anything which is total and computable by both b and by c will be computable by the first n columns of b due to lack of $e-$ splitting.

Definition:i) A is $1-generic$ if it forces every c.e. set of strings V i.e. $\forall e, \exists \sigma \prec A\ \sigma \in W_e$ or $\forall \tau \succ \sigma\ \tau \notin W_e$.
ii) A function f is hyperimmune if it is not dominated by any computable function. A set is hyperimmune if its principal function is hyperimmune. A degree is hyperimmune if it computes a hyperimmune function/set.

Theorem: A is $1-generic \iff$ it forces the jump i.e $\forall e, \exists \sigma \prec A\ \varphi_e^\sigma(e) \downarrow$ or $\forall \tau \succ \sigma \varphi_e^\tau(e) \uparrow$.
Theorem: i) A non computable c.e. set has hyperimmune degree
ii) A non computable Σ_2^0 set has hyperimmune degree.
iii) There is a non computable Δ_3^0 set ($0''$ computable) of hyperimmune free degree. Moreover this set is also low_2.
Proof Idea: i) If the function taking $n \to$ least number of steps needed to decide if n is in the c.e. set was dominated by a computable function, then the c.e. set would be computable
ii) If it is hyperimmune free, then it boils down to the c.e. case which was handled above.
iii) Forcing with $0''$ oracle using f-trees. Force φ_e^A to be total if we can. Then if φ_e^A ends up being total it will be majorized by a computable function which takes $n \to$ max value in the n^th level of the tree constructed at stage corre-

sponding to φ_e^A. This is low_2 since the construction can decide ToT^A.

Definition: i) A degree a is minimal if $b \leq_T a \implies b \equiv_T 0 or b \equiv_T a$.
ii) A string σ on a computable f-tree is said to $e-split$ if it has two extensions τ_1, τ_2 on the tree which make $\varphi_e^{\tau_1}(x) \neq \varphi_e^{\tau_2}(x)$ on some input x. iii) A tree is $e-splitting$ if every string on it is $e-split$ by its immediate descendant.

Theorem(Spectors minimal degree): There is a $0''$ computable minimal degree. When relativized this says that every set has a minimal cover but it need not have a strong minimal cover (Eg: $0'$).
Proof Idea: Construct a sequence of computable $f-trees$ using $0''$ oracle. At stage $2e$ find an $e-splitting$ sub f-tree and if it is total add it to our sequence of trees. Otherwise add the subtree rooted at the point where $e-splitting$ failed. In the first case if φ_e^A is total, then it computes A since A is a path through a $e-splitting$ tree. In the second case if φ_e^A is total it is computable since the tree has no $e-splitting$.

Theorem If A is $nonGL_2$ (i.e. $A'' \not\leq_T (A \oplus 0')'$) it computes a $1-generic$.
Proof Idea:Given a listing of $c.e.$ sets, let f be the funcing taking n to the least stage s such that every a string of length at most n that has an extension in one of the first n $c.e.$ sets in our listing, has an extension within s stages of that $c.e.$ set. This f is $0'$ computable and so by Martins high domination (relativized) there must be a A computable function g which is not dominated by f. The $1-generic, B$ is constructed in stages, such that at stage s we have the first s bits of B.
At stage s look at the first s $c.e.$ sets within $g(s)$ steps. Pick the first $c.e.$ set which is not satisfied and which has some extension of b_{s-1} in it. Then extend b_{s-1} by 1 bit along this extension.

Theorem There is a set $A < 0'$ such that $A' \equiv_T 0''$
Using $0'$ oracle we construct A so that the limit of its columns is $0'$. Let $0'' = W_j^{0'}$. In the odd stage $2i+1$ we diagonalize against φ_e^A for $e \leq i$ to make it not equal to $0'$ by trying to find $e-splitting$ while not violating 'higher priority' columns $e' < e$ trying to respect $0''(e') = lim_y A(< e', y >)$.
At even stages $2e$ we fill in unfilled entries in the first $e \times e$ submatrix of A with our guess of $W_{j,e}^{0'}$.
Theorem(Shoenfeld Jump Inversion) Let $S \geq 0'$ be Σ_2^0. Then there is an $A \leq_T 0'$ such that $A' \equiv_T S$.
Proof Idea: Replace diagonalization step in previous proof with forcing the jump.

4 Trees and Paths

A Π^0_1 class is the complement of a Σ^0_1 class. It can be characterized as the set of paths through a computable tree or equivalently the set of paths through a Π^0_1 tree.
An infinite tree must have a path through it. Therefore $\{X\}$ is a Π^0_1 class $\iff$ X is computable. Similarly any isolated element of a Π^0_1 class must be computable.
Definition $DNC_2 = [T]$ where T is the set of strings σ such that $\forall s, e\varphi_{e,s}(e) \neq \sigma(e)$
No path through this tree can be computable, and DNC_2 is a Π^0_1 class.
Theorem If $X \in DNC_2$ and $C \neq \emptyset$ is a Π^0_1 class, then $\exists Y \in C$ such that X computes Y.
Proof Idea: Let T be a computable tree such that $C = [T]$. Then let $\psi(\sigma, y)$ take value $0, 1$ on whether the subtree of T above σ is taller to the right of σ or to its left respectively. If both are infinite $\psi(\sigma, y)$ diverges. Let $\varphi_{g(\sigma)}(y) = \psi(\sigma, y)$. Then since $X(e) \neq \varphi_{g(e)}(g(e))\forall e$, we have that if the tree is infinite above σ it is infinite above $\sigma \frown X(g(e))$.

Definitioni) Given disjoint c.e. sets A, B let $Sep(A, B) = [\{\sigma : \forall n < |\sigma| n \in A_{|\sigma|} \implies \sigma(n) = 1, n \in B_{|\sigma|} \implies \sigma(n) = 0\}]$. Note $Sep(A, B)$ is a Π^0_1 class and DNC_2 is a special instance of this.
ii) Given a computable consistent set of axioms T, and a computable list of all sentences in the language, all complete extensions of T can be encoded as paths through a tree where a finite string σ is in the tree if the set of sentences indexed by n such that $\sigma(n) = 1$ union the negation of sentences where $\sigma(n) = 0$ is consistent with T.
Theorem: The following are equivalent:
i) X computes a DNC_2 function
ii) X computes an element of any non empty Π^0_1 class.
iii) X computes a separating set for any pair of disjoint c.e. sets
iv) X computes a complete consistent extension of PA.
Proof Idea: For iv) $\implies$ i) we have to use the fact that complete theories contain a sentence or their negation.

Theorem (Basis theorems) Let P be a non empty Π^0_1 class. Then :
i)P has a set of *c.e.* degree
ii)P has a low set.
iii)P has a hyperimmune free set.
Proof Idea: i) The set of strings to the left of the left most path is a *c.e.* set. This set also computes the left most path.
ii) Force with Π^0_1 classes. Try to make $\varphi^\sigma_e(e)$ diverge if possible. If the set of such strings is infinite ($0'$ question) then take this as the next tree in our sequence. Otherwise keep the same tree.
iii) Try to force $\varphi^A_e(x) \uparrow$ for a given e, x at the e^{th} stage of the construction. If there is an infinite set of strings where this happens, set that as the next

tree in the sequence, otherwise $\forall x$ the set of strings is finite. We can use this to find a computable function to majorize any total function which A computes.

By the recursion theorem, we can control a computable sequence of positions of the diagonal functions as follows: Given a p.r. $\psi(n)$, by the $s-m-n$ theorem there is a computable function α such that $\psi(n) = \varphi_{\alpha(n)}(\alpha(n))$. (For example $\alpha(n)$ is the index of a constant function which takes value $\psi(n)$). Now by the recursion theorem we know an index for ψ and can use it while defining ψ. Therefore we effectively control the computable sequence $\{\alpha(n)\}_n$ of positions of the diagonal function.
Given a *c.e.* set W_e its modulus function $m : \omega \to \omega$ is given by $m(x)$ is the least stage s so that W_e and $W_{e,s}$ agree on the first x bits. Any function that majorizes m can compute W_e.

Theorem Given a set X of DNC degree and a *c.e.* set C, either X and C together compute $0'$ or X is $C-DNC$.
Proof Idea: Let $g \leq_T X$ be DNC. Assume we control an array of positions of the diagonal function. We make the n^{th} column of this array take values $\{\varphi_m^{C_s}(m)\}_{m\in\omega}$ if $n \in 0'_s - 0'_{s-1}$.
This makes the DNC function g's image of the n^{th} column 'C_s DNC'. If one of these columns is truly $C-DNC$ we are done. Otherwise for every column there is an entry whose g value agrees with a $C-diagonal$ value. This lets us majorize the modulus function of 0' using $g \leq_T X$ and C.
Note DNC degrees are precisely the FPF degrees (degrees which compute a fixed point free function). Therefore any *c.e.* set which does not compute $0'$ will have a fixed point.

Theore If $0'$ is $A-DNC$ then A must be GL_1.
Proof Idea: Let g be a $A-DNC, 0'$ computable function. Let $\{k_n\}_n$ be a computable sequence of positions of the $A-diagonal$ that we $A-computably$ control. g can be approximated by a computable function h.
If A witness n enter A' at stage s then we set the k_n^{th} diagonal position to be $h(k_n, s)$. Since the limit of h is g, and g disagrees with a diagonal this ensure that the k_n^{th} column of h changes value after s. This lets us define a function which majorizes the modulus of A' and so $A \oplus 0'$ can compute A'.

Definition: A set is simple if it is *c.e.* and its complement is immune i.e. does not contain an infinite c.e set.
Theorem(Kucera): Every Δ_2^0 DNC function computes a simple set. (Answering Post's problem)
Proof Idea: Let h be a computable approximation to our Δ_2^0 DNC function g. We use the computable sequence of diagonal positions $\{k_e\}_e$ that we control by setting say for some $x = k_e$, $\varphi_x(x) = h(x, s)$ for some stage s where we want to ensure h hasn't settled down yet.
At stage s we look at all unsatisfied $W_{e,s}$ for $e \leq s$ which have an element x which is between $2e$ and s such that the k_e^{th} column of h is constant in the interval

$(x, s]$. Such an x is ennumerated into our *c.e.* set, and we set $\varphi_{k_e}(k_e) = h(k_e, s)$. This ensures that h hasn't settled down yet (as it will differ from g which is DNC).
This will let us compute the *c.e.* set from g as we can check for when h has settled down on a particular column and if the element corresponding to that column hasn't entered the *c.e.* set yet, it never will.

5 Old School Constructions

Definition: A set is cohesive if it is infinite and there is no infinite *c.e.* set so that both it and its complement contain an infinite chunk of the cohesive set. A *c.e* set is maximal if its complement is cohesive.
Theorem There is a maximal *c.e.* set
Proof Idea: Moveable marker construction of a c.e set A to meet the cohesive requirements. For each e we try to make almost all the markers be in W_e. We move the e^{th} marker only when an i^{th} marker for $i > e$ occurs in (lexographically) more of the $W_{0,s}....W_{e,s}$ in which case we ennumerate $[a_e^s, a_i^s)$. All the markers eventually settle down and markers to the right occur in fewer of $W_0, W_1, ..., W_e$ (lexographically). Due to this monotonicity, all but finitely many of the a_n are in W_e or $\overline{W_e}$.

Theorem: High c.e. sets compute maximal sets.

6 Finite Injury Method

Definition A set is called immune if it is infinite and does not contain an infinite *c.e.* subset. A set is simple if it is c.e. and its complement is immune.
Theorem There is a low simple set.
Proof Idea; We have a positive requirement P_e saying that $W_e \cap A$ is non empty and a negative requirement N_e which ensures lowness saying that the limit of $g(e,s) = 1$ if $\varphi_{e,s}^{A_s}(e) \downarrow$ and 0 otherwise exists.
Positive elements try to ennumerative a large enough element of W_e into A while respecting higher priority negative requirements which put up a restraint to try to preserve a computation and make sure $lim_s g(e,s)$ exists for all e.

Theorem A non computable c.e. set, it computes a low simple set.
Proof Idea; To the above argument, when a P_e is trying to ennumerate x into A we have to wait for permission from our non computable *c.e.* set B i.e $B_{s+1}|_{x+1} \neq B_s|_{x+1}$.
If there is an unsatisfied P_e then we can compute $n \in B$ by waiting for a large enough stage so that a number $x > n$ enters W_e. Since x didn't get permission to enter A at that stage s, $B|_{x+1} = B_s|_{x+1}$, a contradiction to B being non computable.

Theorem (Friedberg Muchnik): There are turing incomparable *c.e.* sets.
Proof Idea; We ennumerate two c.e. sets while requiring that neither one computes the other. To make sure $\varphi_e^A \neq B$ we pick a witness x to diagonolize against and wait for $\varphi_e^A(x)$ to converge and equal 0 on our witness in which case we put it into B. We then try to preserve this computation with the priority available to us. Every requirement can only be injured finitely often (each injury makes us choose a new witness), and eventually gets satisfied.

Theorem There is a perfect Π_1^0 class such that any two elements in it are turing incomparable.
Proof Idea; We build a computable sequence of computable trees whose limit exists, call it T. Then the requirements are that for any two string at the same level, an extension of one does not compute an extension of the other.
An unsatisfied requirement corresponding to (e, σ, τ) requires attention if there is some extension of $T_s(\sigma)$ on the tree, which makes $\varphi_e^{T_s(\sigma')}$ look like a total function upto length $T_s(\tau) + 1$. Picking the highest priority requirement which needs attention we diagonalize against it.

Theorem: For every non computable c.e. set there is a simple set which does not compute it.
Proof Idea: If $\{C_s\}$ is an approximation of the non computable c.e. set and A is the simple set we are trying to build, the positive requirements try to ennumerate elements into A to make it intersect the infinite W_e. The negative requirements try to make sure that $C \neq \varphi_e^A$. For this we use the sacks preserva-

tion strategy, we define the length of agreement $l(e,s)$ between $\varphi_{e,s}^{A_s}$ and C_s and try to preserve the computation of $\varphi_{e,s}^{A_s}$ upto $l(e,s)+1$ by putting up a restriant. For a stage beyond which our negative requirement is not injured then a change in $l(e,s)$ can only be due to a C change. If an element $\leq l(e,s)$ is put into C then $l(e,s)$ decreases as φ_e^A is preserved. If no such element enters C then $l(e,s)$ keeps increasing.
If $l(e,s)$ stabilizes, our N_e requirement is satisfied. Otherwise we can compute C using our computable function $l(e,s)$ (which is monotonically increasing beyond a stage) and so this can't happen.
A is low since the limit of $l(e,s)$ can determine if $e \in A'$ (using $\varphi_{g(e)}^A(n) = C(0)$ when $n = 0, \varphi_e^A(e) \downarrow$, and diverges otherwise. Here $lim_s l(g(e),s)$ is either 0 or 1).

Theorem (Sacks Splitting): Given c.e. sets B, C with C non computable, there is a partition of B into *low c.e.* sets A_1, A_2 neither of which compute C.
Proof Idea: We want to ensure that $C \neq \varphi_e^{A_i}$. When an element appears in B we either put it in A_1 or A_2. To decide which A_i to put it in, we find the highest priority requirement whose restraint would be violated by x entering A_i (where the restraint is based on length of agreement of $\varphi_{e,s}^{A_{i,s}}$ and C_s as before) and put it into the other side A_{3-i}.

7 The Infinite Injury Method

Theorem (Friedberg Muchnik) There are two turing incomparable *c.e.* sets.
Proof Idea: We give an infinite injury style proof using the tree method. The requirements are $A_i \neq \varphi_e^{A_{1-i}}$. Each level of the tree handles one requirement. There are two outcomes for a requirement $\{w, d\}$ (for wait and diagonalize) with d being to the left of w. At stage s there are s substages- we start at the root, and move upto the s^{th} level following the current outcome at each node to decide which path to choose.
The path through the tree which picks the 'true' outcome of a strategy, is called the true path, it is the $liminf$ of the current path. We then argue that every node on the true path meets its requirement.

Theorem There is a high *c.e.* set not computing $0'$.
Proof Idea: We built c.e. sets C, H and a computable function Γ so that ToT is the column limit of Γ^H (for highness) and C is not H computable (so that $H \not\geq_T 0'$). The positive requirements P_e say that $ToT(e) = lim_s \Gamma^H(< e, s >)$, while the negative requirements ask for $\varphi_e^H \neq C$.
We construct our total Γ by ennumerating a suitable *c.e.* set of axioms (x, i, σ) interpreting σ as the oracle, x is the input to Γ and i is the output. At stage v we set $\Gamma^H(e, v)$ to be 0 by ennumerating the axiom saying Γ has use $\gamma(e, v) + 1$ where $\gamma(e, v)$ is a number larger than any number listed so far. If we later want to make $\Gamma^H(e, v) = 1$, ennumerate $\gamma(e, v)$ into H and ennumerate the corresponding axiom for it.
A node for the P_e strategy initialized at s_0 sets a variable $n = s_0$, waits for the first n elements to enter W_e and then makes $\Gamma^H(e, v) = 1$ for all column entries between s_0 and the current stage v. It then increments n by one and waits for the next element to enter W_e and repeats this process.
A N_e node, picks a large witness, keeping it out of C while waiting for $\varphi_{e,s}^{H_s}(x) \downarrow = 0$ for a computation it believes in, and then ennumerates x into C. We say an N_e node believes a computation if for every P_i strategy which is an ancestor of it on the tree of strategies, which the N_e node believes will have a ∞ outcome, the $\gamma(i, v)'s$ which will be ennumerated into H have already been put into H- this ensures that ancestors cannot hurt our N_e node.
The outcomes for P_e are $\infty < 0 < 1 < 2...$ and for N_e are $d < w$.
I can be argued considering all the possible cases that every node on the true path satisfies its requirement. Note that once a node is on the true path, no node to its left at the same level is every visited again (for both P_e and N_e nodes). So we have to consider predecessor nodes, successor nodes and nodes to the right.
Theorem(Sacks Jump Inversion): If S is Σ_2^0 and computes $0'$, then there is a *c.e.* set A whose jump is S, i.e $A' = S$.
Proof Idea: This is an extension of the proof idea of the previous theorem. The P_e requirements are now that the column limit $lim_s \Gamma^A(e, s)$ computes $\overline{S}(e)$. Since $\overline{S} \leq_m ToT$, we can now use the same strategy as before to meet P_e requirements.

The N_e requirement is to force the jump of A. If $\varphi_{e,s}^{A_s}(e)\downarrow$ for a computation which the node fullfilling the strategy believes in, then it tries to preserve the computation by putting a restraint upto the use of the computation. Whenever the current path moves to the left of a particular node, it gets reinitialized i.e. P_e's pick new $\gamma(e,v)$'s and N_e's remove their restraint.
The P_e's nodes on the true path meet their requirement, the N_e nodes on the true path place finite restraint, and S computes the true path and so can compute A'. By construction A' computes S, and so we are done.

Theorem: There is a minimal pair of c.e. degrees.
Proof Idea: We build two *c.e.* sets A_1, A_2. The positive requirements are to diagonalize against computable functions $P_e^i : A_i \neq \varphi_e$. The negative requirements N_e says that if A_1 and A_2 both compute a total function g, then g must be computable. They manage this by setting $n = 0$, and waiting for $\varphi_{e,s}^{A_{1,s}}$ and $\varphi_{e,s}^{A_{2,s}}$ to agree on the first $n+1$ bits and if this happens, preserving one of them on the first $n+1$ bits and increasing n by 1 and going back to waiting.
As usual all already visited nodes to the right of the current path get reinitialized. We further ensure that at each stage at most one element is ennumerated, either into A_1 or A_2 (once a P_e substage ennumerates an element, we terminate that stage). We can now argue that all nodes on the true path fullfil their requirements (Lots of cases!). To see that the N_e requirement is satisfied, if the ∞ outcome occurs for N_e, that is, length of agreement is infinite, then to compute $g(x)$ we just have to wait for a stage until $\varphi_{e,s}^{A_{1,s}}$ and $\varphi_{e,s}^{A_{2,s}}$ agree upto $x+1$, and output whatever they compute for x. Since we preserve one of them and only allow the other to change, this ensures that we get the correct answer.

Theorem: Given a $0''$ computable linear order $\mathcal{L}$ with a least element, $\omega.\mathcal{L}$ is computable.
Proof Idea: Let $<_g$ be the $0''$ computable order on ω which makes it isomorphic to $\mathcal{L}$ and let 0 be the $<_g$ least element. The set of outcomes is $\infty < 0 < 1 < 2...$ and for the tree of strategies, the string σ on the tree is the guess that the largest initial segment of ω in W_e is $\sigma(e)$. So the true path has ∞ outcome at a level $e \iff W_e$ is total.
At every stage we construct a finite linear order $A, <_A$ and each node β has a map $\gamma_\beta : \omega \to A$ which it thinks gives the cannonical embedding of $\mathcal{L}$ into $\omega.\mathcal{L}$. The strategy for a node is as follows: If the node with its guess at ToT determines $<_g$ on a larger initial segment of ω than its predecessor node does,

8 Computable Well-orders and the Hyperarithmetical Hierarchy

9 Important concepts for E section

Theoremi) (Konig)$\kappa \geq 2, \lambda$ infinite $\implies cf(\kappa^\lambda) > \lambda$.
ii) (Cantor normal form) Any $\alpha \in ON$ can be written uniquely as $\alpha = \sum_{i=1}^{n} \omega^{a_i}.n_i$ where a_i's are a strictly decreasing sequence of ordinals and n_i are natural numbers.
iii) (Hartog) Given any set X there is a cardinal which does not embed into it (without using AoC).

Definitions:i) A set of sentences Σ is said to be complete if it models every ψ or its negation.
ii) Two structures are elementarily equivalent if they satisfy the same set of sentences. Isomorphism $\implies$ elementary equivalence. iii) A set of sentences Σ is called κ categorical if all models of Σ of cardinality κ are isomorphic.
iv) A substructure $B \subset A$ is an elementary substructure if for any formula with interpretation in B, A models the formula $\iff$ B does. (This is stronger than elementary equivalence since we have parameters in formulas (not just sentences)).
v) A set of sentences Σ is model complete if for any two of its models such that one is the substrcture of the other, it is an elementary substructure.

Theoremi) (Compactness) If Σ is a set of sentences such that every finite subset has a model, then Σ has a model. This can be restated as- if $\Sigma \models \psi$ then there is a finite subset of Σ which models ψ.
ii)(LST) If Σ is a set of sentences which has models of size $\geq n$ for every $n \in \omega$, then $\forall \kappa \geq |L| + \aleph_0$ there is a model of Σ of cardinality κ.
iii) (Los Vaught's test): Given a consistent set of sentences Σ of a countable language such that Σ has no finite models, if Σ is κ categorical for some cardinal $\kappa \geq \aleph_0$, then Σ is complete.
iv) (Tarski Vaugh Criteria); If A is a substructure of B, then it is an elementary substructure $\iff$ for an existential formula $\varphi(x) = \exists y \psi(x, y)$ such that for parameters $a \in A, B \models \varphi(a)$, there is a $b \in A$ such that $B \models \psi(a, b)$.
v) (Downward LST): Given a model B and a cardinal κ such that $|L| + \aleph_0 \leq \kappa \leq |B|$ and a subset $X \subset B$ with $|X| \leq \kappa$, there is an elementary substructure of A containing X of size κ.
vI) (Upward LST): Given a infinite model B and a $\kappa \geq |L| + |B|$, there is an elementary extension of B of size κ.

Definitions: A theory T is axiomatizable if there is a computable set of sentences Σ such that $consequences(\Sigma) = T$. It is decidable if T itself is computable. If T is c.e. it is semidecidable.
Theorem i) Decidable $\implies$ axiomatizable $\implies$ semi decidable. Also axiomatizable and complete $\implies$ decidable.
ii) (Godels first incompleteness) The theory of the natural numbers is undecidable (also unaxiomatizable since its complete). iii) (Tarski - undefinability of truth) The set $\#Th(\mathcal{N})$ is not definable in $\mathcal{N}$.

iv) (Fixed point lemma for $\mathcal{N}$ - Godel) For every formula $\varphi(.)$ there is a sentence τ such that $\mathcal{N} \models \tau \iff \varphi(\#\tau)$.
v) (Godels second incompleteness): Given a computable set of sentences Σ such that its consequences contain PA, then $\Sigma \vdash Con(\Sigma) \iff \Sigma$ is inconsistent.