ON THE C.E. DEGREES REALIZABLE IN Π^0_1 CLASSES

BARBARA F. CSIMA, ROD DOWNEY, AND KENG MENG NG

ABSTRACT. We study for each computably bounded Π^0_1 class P the set of degrees of c.e. paths in P. We show, amongst other results, that for every c.e. degree $\boldsymbol{a}$ there is a perfect Π^0_1 class where all c.e. members have degree $\boldsymbol{a}$. We also show that every Σ^0_3 set of c.e. indices is realized in some perfect Π^0_1 class, and classify the sets of c.e. degrees which can be realized in some Π^0_1 class as exactly those with a computable representation.

1. INTRODUCTION

This paper is concerned with computably bounded Π^0_1 classes. Of course we can consider these classes, up to Turing degree, as being a collection of infinite paths through a computable binary tree. They have deep connections with computability theory in general, as well as reverse mathematics, algorithmic randomness and many other areas. For example, see Cenzer and Jockusch [3].

The meta-question we want to address in this paper concerns realizing c.e. degrees as members of Π^0_1 classes. One of the fundamental theorems in this area is Kriesel's Basis Theorem, which says that each computably bounded Π^0_1 class has a member of computably enumerable degree. That is, if α is the left- or right-most path of a Π^0_1 class P, then there is a c.e. set W such that $W \equiv_T \alpha$.

Definition 1.1. We will say that a c.e. degree $\mathbf{w}$ is *realized* in a Π^0_1 class P iff there exists some $\beta \in P$ with $deg_T(\beta) = \mathbf{w}$. A set of c.e. degrees S is realised in P if for every c.e. degree $\mathbf{w}$, $\mathbf{w} \in S$ if and only if $\mathbf{w}$ is realised in P.

Our fundamental question is "What sets of c.e. degrees can be realized in a Π^0_1 class?" In this paper, we will give a characterization of the sets of c.e. degrees that can be realized. In turn, this also leads to the related question of representing index sets, and it turns out that these two notions coincide. In more detail, recall that a set $I \subseteq \mathbb{N}$ is an *index set* if it is closed under Turing equivalence, that is, for every $W_e \equiv_T W_j$, $e \in I \Leftrightarrow j \in I$. Index sets are another central area of classical computability theory, and many early results calculated the complexity of such sets. For example, $\mathrm{Cof} = \{e \mid W_e \text{ is cofinite}\}$ is a well known Σ^0_3 m-complete index set. But our concern here is the corresponding set of degrees in an index set, and how we might represent them most simply. For example, $\{e \mid W_e \equiv_T \emptyset'\}$ is a well-known Σ^0_4-complete index set, but we can represent it by a single index for $\emptyset'$, if we allow closure under Turing degrees. With this in mind, define for $S \subseteq \mathbb{N}$ the index set that S represents, $G(S) = \{e : (\exists j \in S)\ W_e \equiv_T W_j\}$. Note that if S is Σ^0_4 then $G(S)$ is also Σ^0_4, while the complexity of $G(S)$ generally cannot be reduced even if

Csima is partially supported by an NSERC Discovery Grant. Downey is partially supported by Marsden Fund of New Zealand. Ng is partially supported by the grant MOE2015-T2-2-055 and RG131/17.

S is of lower arithmetical complexity. For instance $G(\{\emptyset'\})$ is Σ^0_4-complete, as we have seen, so that sometimes the complexity of S can be much simpler than that of $G(S)$. Thus, our main task in this paper is also to investigate, for a given index set I, what the minimal complexity of S such that $G(S) = I$ can be.

In the same way that we did for degrees, we will say that a c.e. set W is *realizable* in a Π^0_1 class P, if $deg_T(W)$ is realizable in P. Henceforth in this paper, all Π^0_1 classes are effectively closed subsets of the Cantor space.

Definition 1.2. If P is a Π^0_1 class, let $W[P] = \{e : W_e \text{ is realizable in } P\}$.

For any Π^0_1 class P, $W[P]$ is clearly an index set. We let $\boldsymbol{W}[P] = \{\boldsymbol{a} : \boldsymbol{a} \text{ is c.e. and realizable in } P\}$. The obvious upperbound on the complexity of $W[P]$ can be easily calculated:

Proposition 1.3. *If P is a Π^0_1 class, then $W[P]$ is Σ^0_4.*

Proof.

$$e \in W[P] \Leftrightarrow (\exists j \exists k)(\forall n)(\exists m)(\exists s > n) \Big[\Phi_j^{W_e} \upharpoonright m[s]\downarrow \ \wedge\ W_e[s] \upharpoonright \varphi_j(m) \text{ is correct}$$
$$\wedge\ \Phi_j^{W_e} \upharpoonright m[s] \text{ is extendible in } P[s] \ \wedge\ \Phi_k^{\Phi_j^{W_e}\upharpoonright m[s]} \upharpoonright n = W_e \upharpoonright n\Big].$$

Note that the predicate within the square brackets is Δ^0_2. □

Note that this upperbound is sharp, in the sense that $W[P]$ is Σ^0_4-complete for some P. For instance if we let P be a class of Martin-Löf random reals, or P be the class of all PA degrees. Recall that every c.e. Martin-Löf random or PA degree must have degree $\mathbf{0}'$. (See, for example, Downey and Hirschfeldt §2.21 and §8.2.)

With these broad questions in mind, we now state some specific questions that shall be addressed in this paper:

(1) Does Proposition 1.3 reverse, that is, is every Σ^0_4 index set realizable in a Π^0_1 class?
(2) If not, can we characterize the index sets S which can be realized?
(3) Is every upper and every lower cone of c.e. degrees realizable?
(4) What about the c.e. degrees realizable in Π^0_1 classes of restricted rank?

We remark that our results go hand in hand with the analogous index set questions about representability. We say that a set S of indices *represents* an index set I if $I = G(S)$. We shall also address the analogous questions about index sets:

(1) Can every Σ^0_4 index set be represented by a Σ^0_3 set or lower?
(2) If not, can we characterize the index sets I which can be represented by Σ^0_3 sets, or even by computable sets?
(3) Is every upper and every lower cone of c.e. degrees representable by a set of lower complexity?

The only related results we are aware of are early results about representing index sets and particularly those corresponding to ideals in the Turing degrees. For example, Yates [11, 12] showed that if $D <_T C$ are c.e. and if S is Σ^C_3, then there is a computable collection of c.e. sets $\{W_{f(k)} \mid k \in \mathbb{N}\}$ such that $D \leq_T W_{f(k)} \leq_T C$ for all k, and where $e \in S$ is equivalent to $W_{f(e)} \equiv_T C$. This was used for an alternative proof of Sacks Density Theorem (see e.g. Soare [10], Ch XII). Yates also showed that a collection of c.e. sets $\mathcal{C}$ containing all finite sets is Σ^0_3 iff there is a computable collection $\{W_{f(d)} \mid d \in \mathbb{N}\}$ which equals $\mathcal{C}$. He used this to show

the classical result that the index of the lower cone $\{e \mid W_e \leq_T A\}$ is Σ^0_3 iff A is low_2. More recently, Barmpalias and Nies [1] proved that for an ideal I in the c.e. degrees:

(1) If I is uniformly superlow generated[1] then it has a superlow upper bound in the c.e. degrees.
(2) If I is uniformly low generated then it has a low c.e. upper bound.
(3) If I is a Σ^0_3 generated proper ideal, then it has a low_2 c.e. upper bound. (See also Downey and Hirschfeldt [5], Ch 11.11.)
(4) If I is a Σ^0_4 generated proper ideal, then it has an incomplete upper bound.

Our paper sharpens some of these results. For example in Theorem 4.6 we show that if I is generated by a Σ^0_4 subset S of a Turing independent set of c.e. sets $\{A_i \mid i \in \mathbb{N}\}$, (i.e. for all finite sets F if $i \notin F$, then $A_i \not\leq_T \oplus_{j\in F} A_j$), then there is a c.e. set B selecting these sets, in that $A_i \leq_T B$ iff $i \in S$.

The first result, Theorem 2.1, shows that singletons can be realized in Π^0_1 classes. Using this we show in Theorem 3.1 that any set of indices which is representable by a Σ^0_3 set, is realized in a Π^0_1 class. Moreover if the set contains an index for a computable set, we can realize this in a rank 2 Π^0_1 class. This allows us to show that certain collections of high c.e. sets can be realized, under a suitable assumption about the uniformity of highness. Moreover, in Corollary 4.3 we establish the rather surprising result, at least to us, below:

Corollary 4.3. An index set I is realizable in a Π^0_1 class iff I has a Σ^0_3 representation iff I has a computable representation.

Next we give an answer to the natural question of exactly which index sets have Σ^0_3 (or equivalently, computable) representations.

Theorem 4.1. Let $S \subseteq \omega$. The following are equivalent.

(i) S has a computable representation, that is, $G(S) = G(R)$ for some computable set R.
(ii) There is a Π^0_1 class P such that the set of c.e. degrees represented in P has index set $G(S)$, that is, $W[P] = G(S)$.
(iii) There is a perfect Π^0_1 class P such that $W[P] = G(S)$.
(iv) There is a computable function g such that for every n,

$$n \in G(S) \Leftrightarrow W^{W_n}_{g(n)} \text{is cofinite.}$$

(v) There is a truth-table functional R such that for every n,

$$n \in G(S) \Leftrightarrow \exists a \forall b \exists c \; R^{W_n}(n, a, b, c).$$

Clearly this result allows us to show certain index sets are not realizable, and others such as, for example, the K-trivial c.e. sets are, as are all upper cones. Finally we turn to the question of exactly which lower cones can be realized. Although we don't characterize this, as mentioned above, we show how to code Σ^0_4 sets into lower cones. In an early incarnation of the present paper, we conjectured that if A is a incomplete c.e. set and the lower cone below A can be realized, then A is low_2. Inspired by our work, Downey and Melnikov have resolved this conjecture affirmatively ([7]).

[1]That is there is a uniformly superlow collection of c.e. sets $\{W_e \mid e \in D\}$ such that $I = \{W_e \mid \exists F \subset D \wedge W_e \leq_T \oplus_{i\in F} W_i\}$.

Finally, we suggest investigating the question of which sets of c.e. degrees are realised in a restricted collection of Π^0_1 classes, for instance, in the collection of thin Π^0_1 classes, or in the collection of separating classes. For instance, the class of PA degrees is a separating Π^0_1 class, and hence $\{\mathbf{0'}\}$ is realised by a separating class. On the other hand, not every singleton can be realised in a separating class. To see this, suppose that the separating class for the pair A, B is realised by a single array computable c.e. degree $\mathbf{d}$. Since A and B are both c.e. members of the separating class, we have that $A \in \mathbf{d}$ and $B \in \mathbf{d}$. Downey, Jockusch and Stob [6] proved that if $A \oplus B$ has array computable degree, and if A, B are disjoint c.e. sets, then there is a Turing complete set separating A and B, a contradiction. In [4], Cholak, Downey, Greenberg and Turetsky have shown that some uppercones can be realised, and some pairs, but the classification of what is realizable remains complex and mysterious.

2. Every c.e. singleton is realizable in a perfect Π^0_1 class

The natural direction to begin our investigation is to look at singletons. Clearly $\{\mathbf{0'}\}$ can be realized by a perfect Π^0_1 class, for instance the class of Martin-Löf random reals, or the class of all PA degrees. Our first result is that $\{\boldsymbol{a}\}$ can be realized by a perfect Π^0_1 class for any c.e. degree $\boldsymbol{a}$.

Theorem 2.1. *For any c.e. degree $\boldsymbol{a}$, we can find (effectively in an index of a c.e. member of $\boldsymbol{a}$) a perfect Π^0_1 class P such that $\boldsymbol{W}[P] = \{\boldsymbol{a}\}$.*

Proof. Let $C \in \boldsymbol{a}$ be c.e., and fix a 1-1 enumeration $\{C_s\}_{s\in\mathbb{N}}$ of C. We build a perfect Π^0_1 class P such that $\boldsymbol{W}[P] = \{\boldsymbol{a}\}$ using a stage by stage construction and satisfying requirements.

Before stating the requirements, we clarify our notational conventions. Recall that a function f is a modulus for the computable approximation $Z(x, s)$ if for every x and every $t > f(x)$, $Z(x, t) = Z(x, f(x))$. If f is nondecreasing then this means that $Z \upharpoonright x + 1$ is stable at $f(x)$. We let $Z_e(x, s)$ be the e^{th} possible Δ^0_2 approximation in some effective listing. Each Z_e is total computable with range in $\{0, 1\}$, where $Z_e(x)$ may not exist. Every Δ^0_2 set is approximated by some Z_e. As usual Φ_e is the e^{th} Turing functional.

We will construct P to meet the requirements

$$\mathcal{R}_e : \text{If } Z_e(x) := \lim_s Z_e(x, s) \text{ exists for every } x, \text{ and } Z_e \in P \text{ and } \Phi_e^{Z_e} \text{ is total and is a modulus for the approximation } Z_e(x, s), \text{ then } Z_e \equiv_T C.$$

Since we are only interested in those Φ_e which are modulus functions, we may assume that for any string ν and $y < x$, if $\Phi^\nu_e(x) \downarrow$ then $\Phi^\nu_e(y) \downarrow$ and Φ^X_e is nondecreasing on its domain for every X. When we refer to $\mathcal{R}_e$ we actually mean Z_i and Φ_j where $e = \langle i, j\rangle$. To avoid introducing too many notations we will write Z_e and Φ_e instead.

If Z is in P and is of c.e. degree, then the premise of $\mathcal{R}_e$ will hold for some e. We will construct P to be the limit of total function trees (see Soare [10]). Namely, we let $T_s : 2^{<\omega} \mapsto 2^{<\omega}$ be a uniform sequence of total computable functions, such that

- for every s, T_s preserves incomparability and inclusion,
- for every s and σ, there is a τ such that $T_{s+1}(\sigma) = T_s(\tau)$,
- for every σ, $\lim_s T_s(\sigma)$ exists.

- for every σ and $i = 0, 1$, $T_s(\sigma * i) \supseteq T_s(\sigma) * i$.

The fourth condition is new and says that T_s has to split at the first place possible. This is not necessary but we adopt this to simplify notation. As before $P = [\lim_s T_s]$ is a perfect Π^0_1 class.

2.1. **Notations.** The construction is carried out on a 2-branching construction tree. We refer to the finite strings on the construction tree as *nodes*, which we denote by letters early in the Greek alphabet such as α, β, γ, etc. We refer to the finite strings in the domain and range of T as *strings*, and we use letters later in the Greek alphabet to denote them, such as σ, τ, ν, etc. Nodes of length e are assigned requirement $\mathcal{R}_e$, and this has two outcomes, ∞ to the left of f. We write $\alpha <_L \beta$ if α is strictly left of β. We say that α *is of higher priority than* β if $\alpha \subset \beta$ or $\alpha <_L \beta$ holds. We say that α *is stronger than* β if $\alpha <_L \beta$, $\alpha \supseteq \beta * \infty$ or $\alpha * f \subseteq \beta$ holds. β will not be allowed to injure a stronger node α, even though traditionally α may be of lower priority.

To show that $Z_\alpha \equiv_T C$ we will not build the reductions explicitly. Instead each α maintains two parameters σ_α and $root_\alpha$ to help guide this reduction. The intention is that σ_α is the string such that $T_s(\sigma_\alpha)$ is currently an initial segment of Z_α which α believes is a true initial segment. We always keep $root_\alpha \subseteq \sigma_\alpha$ if both are defined, and that $root_\alpha \downarrow$ iff $\sigma_\alpha \downarrow$. For each $k \leq |\sigma_\alpha| - |root_\alpha|$, we let $root^k_\alpha = \sigma_\alpha \upharpoonright (|root_\alpha| + k)$, the k-bit extension of $root_\alpha$ along σ_α. C will be coded into Z_e above $root_\alpha$. In particular we code $C(k)$ in $T_s(root^{k+2}_\alpha)$, and if k enters C we force Z_α to change below $T(root^{k+2}_\alpha)$.

For $\alpha, \sigma \in 2^{<\omega}$ we say that σ *is* α*-good* if whenever $\sigma_\alpha \downarrow$, we have $\sigma \not\subseteq \sigma_\alpha$. The three possibilities are:

- $root_\alpha$ and σ are incomparable,
- $root_\alpha \subseteq (\sigma \cap \sigma_\alpha) \subset \sigma_\alpha$, or
- $\sigma \supset \sigma_\alpha$.

Note that any extension of a good string is good. By σ^- we mean the string σ with the last bit removed, and is defined to be the empty string if $|\sigma| = 0$. We divide stage s into substages $u < s$, in which a different node gets to act at each substage. If a node α is visited at (sub)stage s, we let s^- denote the (sub)stage of the previous visit to α (it will be clear from context which α we are referring to). If α is visited at a stage s, and β is a node such that $\beta * f \subseteq \alpha$, we let $\ell_\alpha(\beta) = \max\{|\tau| : \tau \subseteq \sigma_\beta \ \wedge \ T(\tau) \subset Z_\alpha\}$. This measures how much of $T(\sigma_\beta)$ is currently an initial segment of Z_α. If $\sigma_\beta \uparrow$ then this is taken to be 0. We let $\ell_\alpha = \max\{\ell_\alpha(\beta) : \beta * f \subseteq \alpha\}$. A (sub)stage t is α*-expansionary*, if α is visited at t, and $\ell_\alpha[t] > \max\{\ell_\alpha[t'] : t' < t$ where $\alpha * \infty$ is visited at $t'\}$. For any α, x and numbers $s < t$, we say that $Z_\alpha[s] \upharpoonright x \equiv Z_\alpha[t] \upharpoonright x$ if $Z_\alpha[s'] \upharpoonright x$ is constant for all $s \leq s' \leq t$. If $\rho \supset \eta$ by *moving* η *to* ρ, we mean to obtain the modification T' of T by letting T' copy T at all inputs not extending η. Then let $T'(\eta * \sigma) = T(\rho * \sigma)$ for every $\sigma \in 2^{<\omega}$.

2.2. **Proof idea.** We assume that $C = \emptyset$, and consider the general case later. We consider $\mathcal{R}_0$ with the highest priority. The basic strategy for $\mathcal{R}_0$ is simple, in fact every $\mathcal{R}_e$ follows roughly the same basic strategy with minor modifications: We begin by letting $\sigma_0 = \emptyset$, and search for some $\rho \supset \sigma_0 * i$ such that $Z_0 \supset T(\rho)$ and $\Phi_0^{T(\rho)}$ looks like a modulus function for Z_0. That is, at the current stage

approximation, $\Phi_0^{T(\rho)}(|T(\sigma * i)|)\downarrow$ and there is no contradition observed for it to be a correct modulus for $Z_0 \upharpoonright (|T(\sigma * i)| + 1)$. While searching for ρ, $\mathcal{R}_0$ will prevent the rest of the construction from changing T on σ_0. If ρ is found we keep the split at σ_0 and kill off all other branches of T between $T(\sigma_0 * i)$ and $T(\rho)$ by moving $\sigma_0 * i$ to ρ. We then append i to σ_0, define $X_0 \upharpoonright |T(\sigma_0)| = Z_0 \upharpoonright |T(\sigma_0)|$. We then repeat. If ever during this process the approximation to $Z_0 \upharpoonright |T(\sigma_0)|$ changes after we have already defined $X_0 \upharpoonright |T(\sigma_0)|$, then we shift $\emptyset$ to σ_0 and consider $\mathcal{R}_0$ satisfied, with no further action required (since we have assured that the modulus is wrong). If this never happens but still σ_0 is extended finitely often then $\mathcal{R}_0$ only restraints a finite part of T, and $\mathcal{R}_0$ is satisfied vacuously. If Z_0 is a path on P with modulus Φ_0^Z, then we be able to extend σ_0 infinitely often and so X_0 is total and equal to Z_0. Thus Z_0 is computable since X_0 is.

From the above description we see that $\mathcal{R}_0$ either acts finitely often (and only restrains a finite part of T), or it acts infinitely often and it restrains other requirements from moving strings extended by σ_0. Note that it is important for $\mathcal{R}_0$ to restrain the rest of the construction in this way because if we allow other requirements to delete $\tau = T(\sigma_0)$ after it has been recorded by X_0, then Z_0 is free to change and become different from X_0. Unlike before we cannot force $\mathcal{R}_0$ to lose by making every path in P extend τ (since τ has already been deleted).

We now consider how $\mathcal{R}_1$ can act consistently with $\mathcal{R}_0$'s demands. If $\mathcal{R}_0$ acts finitely often then $\mathcal{R}_1$ begins a new version after $\mathcal{R}_0$'s final action, which will never be interrupted. Suppose $\mathcal{R}_0$ acts infinitely often. Then we know that $X_0 = Z_0 = T(\lim \sigma_0) \in P$, and $\mathcal{R}_0$ will restrain more of Z_0 as the construction proceeds. In this case $\mathcal{R}_1$ will only begin running its basic strategy if it is able to define σ_1 so that $Z_1 \supset T(\sigma_1)$ and is incomparable with $T(\sigma_0)$. Note that if $\mathcal{R}_1$ is unable to do this then $Z_1 = Z_0$ which we know is computable through the actions of $\mathcal{R}_0$. On the other hand suppose $\mathcal{R}_1$ starts its basic strategy, and begins building X_1 incomparable with X_0. Suppose Z_1 then moves at a stage t below a length for which we have defined X_1. As in the basic strategy we need to kill $\mathcal{R}_1$. This time due to priority we kill off every part of the tree except for $T(\sigma_0)[t]$ and $T(\sigma_1)[t]$, and all other nodes above which which $\mathcal{R}_1$ has been killed. Unlike the case for $\mathcal{R}_0$ this is not an immediate win for $\mathcal{R}_1$, even though $\mathcal{R}_1$ will no longer be active above $T(\sigma_1)[t]$ we may have $Z_1 \supset T(\sigma_0)[t]$. Hence we may need to restart $\mathcal{R}_1$ at a different part of the tree. If $\mathcal{R}_1$ is killed and restarted infinitely often then $Z_1 = Z_0$ which is computable through the actions of $\mathcal{R}_0$. Otherwise $\mathcal{R}_1$ begins building X_1 incomparable with X_0 without interruption, and hence $Z_1 = X_1$ is computable.

If we now consider the third requirement $\mathcal{R}_2$, then the strategy for $\mathcal{R}_2$ will have the obvious modifications. $\mathcal{R}_2$ will build X_2 incomparable with X_0 and X_1. Each time we kill off $\mathcal{R}_2$ we have to leave $T(\sigma_0)$ and $T(\sigma_1)$ on the tree. If $\mathcal{R}_2$ is killed and restarted infinitely often then Z_2 is equal to either Z_0 or Z_1 and hence computable. Considering three requirements add a certain complexity to the situation: It is now possible for $\mathcal{R}_1$ to be killed and restarted infinitely often (hence $R_1 = R_0$) and this can affect $\mathcal{R}_2$ in the following way. For instance $X_1 \cap X_0 = X_2 \cap X_0$ may hold, and when $\mathcal{R}_1$ is killed we need to initialize $\mathcal{R}_2$ since X_2 is taken off the tree and hence Z_2 can now change freely. If this happens infinitely often then we also have $Z_2 = Z_0$ and hence Z_2 is computable. Otherwise Z_2 will extend $T(\sigma)$ for some σ where $\mathcal{R}_1$ is killed. Then Z_2 will begin building X_2 in this part without interference.

The construction will be implemented on a priority tree. This is not necessary, but will make organization easier. Outcome f of a node α stands for the situation where α is killed finitely often, while outcome ∞ means that α is killed infinitely often. This proof has the unusual feature where a node α may initialize $\gamma \supseteq \alpha * f$ infinitely often, even when α has stopped playing outcome ∞. Hence a node β on the true path of the construction may be injured infinitely often by some $\alpha * f \subseteq \beta$. We ensure that if this is the case, then the requirement assigned to β will be met through some node which is extended by β, as in the discussion above. Each node α will respect the nodes β extending $\alpha * \infty$, even though β is traditionally of lower priority. However β will only get to act at those stages where α's current attempt has just been destroyed, and β will only be allowed to act infinitely often if α is injured infinitely often. Hence the portion of the tree restrained by β does not grow unless α is injured again (hence allowing β to be visited).

Now we consider the case where C is arbitrary. We have to code C into Z_e. We modify the strategy outlined above in the following way. A node α will now look for an appropriate string $root_\alpha$ so that $T(root_\alpha)$ is incomparable with all strings which are restrained by stronger requirements. Once found α will begin a new attempt at demonstrating that $C \equiv_T Z_\alpha$ where the coding locations are above $T(root_\alpha)$. Initially we start off with $root_\alpha = \sigma_\alpha$. Each time α extends σ_α we set a new coding location for C. If n enters C for a small k, then we have to force $Z_\alpha \upharpoonright T(root_\alpha^{k+2})$ to move. We do this by removing the branch $T(root_\alpha^{k+2})$ from the tree. Hence Z_α has to move or else it stays out of P. To see that Z_α is computable in C, we note that as above, Z_α cannot change unless C changes and we kill off $T(root_\alpha^k)$ for some k. If Z_α changes without a corresponding C change, then we have not yet removed $T(\sigma_\alpha)$ from the tree, so we can now force a win for α above the string $root_\alpha$ by killing off every string extending $T(root_\alpha)$ except for $T(\sigma_\alpha)$. If $\alpha * f \subseteq \beta$ and if $Z_\alpha \neq Z_\beta$ then α will only initialize β finitely often. In fact β will be able to carry out its basic strategy with $T(root_\beta) \supset Z_\alpha \cap Z_\beta$. On the other hand if α initializes β infinitely often then $Z_\alpha = Z_\beta$ and α will be responsible for building the reduction $Z_\alpha \equiv_T C$.

2.3. **Formal construction.** At stage $s = 0$ we let T_0 be the identity function, and *initialize* α (i.e. set σ_α and $root_\alpha$ undefined) for every α. In the construction α may be initialized when another node of higher priority acts, or when α is visited and we see the approximation for Z_α move below σ_α. The former we call an *external initialization*.

All searches performed in stage s are limited to numbers and strings of length $< s$. At stage $s > 0$ suppose $T = T_{s-1}$ is given. We define δ_s, the approximation to the true path of length s. At substage $u < s$ we define $\delta_s(u)$. Whenever a parameter is referenced we mean the value of the parameter at the instance (or substage) they are mentioned. Since the construction takes place on a priority tree, to make housekeeping easier we will assume that when each node α is visited for the k^{th} time in the construction, it only looks at $Z_\alpha(x, k)$. That is, during the construction and verification, when we refer to $Z_\alpha[t]$ and $\Phi^{Z_\alpha}[t]$, we mean the values when both are evaluated at k.

Suppose we have defined $\delta_s \upharpoonright u$ for some $u < s$. Let $\alpha = \delta_s \upharpoonright u$. If α was initialized externally since the last visit to α, or if $\sigma_\alpha \uparrow$ and s is α-expansionary, we let $\delta_s(u) = \infty$, initialize every node to the right of $\alpha * \infty$ and end the substage. Otherwise there are two cases:

(1) $\sigma_\alpha \uparrow$: Find a string η where $|\eta| > \max\{|\alpha|, s_0\}$ such that η is β-good for every β stronger than α. Here $s_0 < s$ is the largest stage such that α was initialized externally. We also assume that s_0 is at least as large as all historical $|root_\alpha|$. We require that there is some $\rho \supset \eta$ such that $\Phi_\alpha^{T(\rho)}(|T(\eta)|) \leq s$, and $Z_\alpha[s] \supset T(\rho)$. Furthermore we need η to be *correct with respect to* ρ: i.e. for every $x \leq |T(\eta)|$, we have $Z_\alpha[\Phi_\alpha^{T(\rho)}(x)] \upharpoonright x \equiv Z_\alpha[s] \upharpoonright x$. If η is found (we always pick η of minimal length), we move η to ρ. Let $\sigma_\alpha = root_\alpha = \eta$.
(2) $\sigma_\alpha \downarrow$: There are three subcases. Pick the first subcase that applies, and perform the actions described there.
   (a) *Killing*: Suppose that $Z_\alpha[s] \not\supseteq T(\sigma_\alpha) \upharpoonright (|T(\sigma_\alpha^-)| + 1)$. Then Z_α has moved below a length which $T(\sigma_\alpha)$ had promised would never change. Now note that Z_α can never extend $T(\sigma_\alpha)$ correctly again, and we can now thin the tree accordingly: Move $root_\alpha$ to σ_α, and initialize α.
   (b) *Coding* C: If there is some $k \leq |\sigma_\alpha| - |root_\alpha| - 2$ such that $k \in C_s - C_{s^-}$, we move $root_\alpha^{k+1}$ to $(root_\alpha^{k+1}) * d$ where $d = 1 - \sigma_\alpha(|root_\alpha^{k+1}|)$. That is, we move $root_\alpha^{k+1}$ to its immediate extension which is not along σ_α. This ensures that the old value of $T(root_\alpha^{k+2})$ is removed from the tree. Now trim σ_α to have length $|root_\alpha^k|$.
   (c) *Extending* σ_α: There is some $i \in \{0, 1\}$ and some $\rho \supset \sigma_\alpha * i$ such that $\Phi_\alpha^{T(\rho)}(|T(\sigma_\alpha * i)|) \leq s$, $Z_\alpha[s] \supset T(\rho)$ and $\sigma_\alpha * i$ is correct with respect to ρ. If i and ρ are found, move $\sigma_\alpha * i$ to ρ and extend σ_α by one digit (i.e. append i to σ_α).

If none of the cases above apply let $\delta_s(u) = f$ and go to the next substage. Otherwise one of the above applies. Then we must have moved some node η to some $\rho \supset \eta$. For every node $\beta \supseteq \alpha * f$ such that $root_\beta$ is comparable with η we initialize β. We let $\delta_s(u) = f$. Go to the next substage.

2.4. **Verification.** For each α, we write "step α.2(a)" to refer to step 2(a) of the construction during substage $|\alpha|$ of some stage, in which α is visited and given attention. Correspondingly we use α.1, α.2(b) and α.2(c) in the obvious way. We use s to refer to stages in the construction, and t, u to refer to substages. If $t_1 < t_2$ then t_1 is a substage before t_2, though they are not necessarily both substages of the same stage. Unless otherwise specified, if P is a parameter then $P[t]$ refers to the value of P at the end of substage t. We begin by verifying several facts about the construction.

Lemma 2.2. (a) *At the end of every substage, for every α stronger than β, if $\sigma_\beta \downarrow$ then $root_\beta$ is α-good.*
(b) *At the end of a substage t, suppose $\sigma_\alpha \downarrow$ and $\tau \subseteq \sigma_\alpha$. Let t' be the largest such that $t' \leq t$ where $|\sigma_\alpha[t'-1]| < |\tau|$ (hence at t' we most recently assigned the value of σ_α to extend τ). Then no prefix of τ is moved strictly between t' and t.*
(c) *For any α once $T(root_\alpha) = \tau$ is killed under step α.2(a), it is impossible for $T(root_\alpha) \supseteq \tau$ again.*
(d) *T_s satisfies all the conditions we required of a sequence of a sequence of total function trees. In particular each σ is moved finitely often.*

Proof. (a): Note that if β plays outcome ∞ at t or if we move left of β, then $\sigma_\beta[t]\uparrow$. Let $t' < t$ be the substage where $root_\beta$ is defined. At t' $root_\beta$ is α-good. Between t' and t if α tries to make $\sigma_\alpha \supseteq root_\beta$ then that same action must initialize β.

(b): Suppose the node β moved some prefix of τ. Then α cannot be stronger than β because of (a). Since $root_\alpha$ is comparable with the node being moved, hence if β is stronger than α it is easy to see that this would cause α to be initialized. Finally suppose that $\alpha = \beta$. Steps $\alpha.1$, $\alpha.2$(b) and $\alpha.2$(c) are not possible since $|\sigma_\alpha| \geq |\tau|$ holds between t' and t. Step $\alpha.2$(a) is not possible because this will cause α to be initialized.

(c): Suppose for a contradiction that $\alpha.2$(a) is taken at t_1, and $T(root_\alpha) \supseteq \tau$ at some $t_2 > t_1$. Hence at t_1 we have $Z_\alpha[t_1] \not\supseteq T(\sigma_\alpha)\restriction(|T(\sigma^-_\alpha)|+1)$. Let $t_3 < t_1$ be the largest substage where $\alpha.1$ or $\alpha.2$(c) was taken. This means that at the end of t_3, we have $\Phi_\alpha^{T(\sigma_\alpha)}(|T(\sigma^-_\alpha)|+1) \leq t_3$, and $Z_\alpha[t_3] \supset T(\sigma_\alpha) = \tau$. By part (b) no prefix of $\sigma_\alpha[t_3] = \sigma_\alpha[t_1 - 1]$ is moved between t_3 and t_1. This means that Z_α has changed below the length $|T(\sigma^-)[t_3]|+1$ between substages t_3 and t_1. There must be some substage between t_1 and t_2, call it $\hat{t}$, for which $root_\alpha$ is defined such that (by applying part (b) again) $T(root_\alpha)[\hat{t}] \supseteq \tau$ under step $\alpha.1$. This is however impossible given that $root_\alpha[\hat{t}]$ cannot be correctly defined at $\hat{t} > t_1$.

(d): At every substage we modify the tree by moving an η to an extension $\rho \supset \eta$, so it is clear that $T_{s+1}(\sigma)$ is on T_s. Suppose for a contradiction that σ is a minimal node moved infinitely often. Since $|root_\alpha| > |\alpha|$ for every α, it follows that if σ is moved infinitely often, there is a highest priority α which moves σ infinitely often. Suppose α moves σ under $\alpha.2$(a) at t. Then $T(\sigma)[t] = \tau$, and we may assume that no proper prefix of σ is moved after t. Hence $T(\sigma) \supseteq \tau$ holds, and by part (c) $root_\alpha \not\supseteq \sigma$ after t.

If α moves σ under $\alpha.2$(b) then it must be because k has entered C for some $k < |\sigma|$, so $\alpha.2$(b) acts finitely often to move σ. Suppose σ is moved under $\alpha.2$(c) at t_1. Then the only way to move σ under $\alpha.2$(c) again at some $t_2 > t_1$ is for $|\sigma_\alpha[t_2 - 1]| < |\sigma| = |\sigma_\alpha[t_1 - 1]|$ to hold. If α gets initialized externally between t_1 and t_2 then $root_\alpha$ will be later picked to have length larger than $t_1 > |\sigma|$ and so $\alpha.2$(c) cannot move σ at t_2. Therefore σ_α must get shorter due to some action of α. If $\alpha.2$(a) applies between t_1 and t_2 then a prefix of σ has to be moved, contradicting the minimality of σ. If $\alpha.2$(b) applies then it must be due to k entering $\emptyset'$ for some $k \leq |\sigma|$ which can only happen finitely often. Hence $|\sigma_\alpha[t_2]| < |\sigma| = |\sigma_\alpha[t_1]|$ cannot hold and so case $\alpha.2$(c) can only move σ finitely often.

Finally assume that σ is moved under $\alpha.1$ infinitely often. This means that α has to be initialized between each such movement of σ. Again this initialization cannot be external, otherwise $root_\alpha$ will be picked to have long length. It is possible for $\alpha.2$(a) to initialize α, but by above this only happens finitely often. □

Hence $P = [\lim_s T_s]$ is a perfect Π^0_1 class. We let TP denote the true path of the construction, defined in the usual way to be the leftmost path visited infinitely often during the construction.

Lemma 2.3. *Suppose that $\alpha * f \subset TP$. The following hold.*

(a) *$|root_\alpha|$ is bounded.*
(b) *Steps $\alpha.1$ and $\alpha.2(a)$ are taken finitely often.*
(c) *$\max\{\ell_\alpha[t] \mid \alpha$ is visited at $t\} < \infty$.*
(d) *If the $\mathcal{R}_\alpha$ hypotheses fail then α acts finitely often.*

Proof. (a): Assume the contrary. Since α is only initialized externally finitely often, it is easy to see that at infinitely many substages t when we pick $root_\alpha$, we have $root_\alpha{}^-$ is not β-good for some stronger β. This must be some $\beta * f \subseteq \alpha$, since nodes to the left of α and extending $\alpha * \infty$ are never visited again. Hence we have $\ell_\alpha \geq |root_\alpha{}^-|$ at these t, which means that some large t has to be α-expansionary, and $\sigma_\alpha \uparrow$, which means that $\alpha * \infty$ has to be played at t, a contradiction.

(b): By (a) and Lemma 2.2(d) it follows that $\alpha.1$ is taken finitely often. Consequently $\alpha.2$(a) is taken finitely often.

(c): Suppose the contrary that $\limsup \ell_\alpha = \infty$ and α plays outcome ∞ finitely often. Let s_0 be large. Note that after s_0, α cannot be initialized externally. By (b) we kill α under $\alpha.2$(a) finitely often. Hence either $root_\alpha$ remains undefined forever, or it receives a final definition. In the former case we will eventually play outcome ∞. In the latter case we have $\ell_\alpha > |root_\alpha|$ holds eventually at some visit to α. Hence $\sigma_\beta \supseteq root_\alpha$ for some $\beta * f \subseteq \alpha$, otherwise we would kill α at that visit. Hence we have a contradiction to Lemma 2.2(a).

(d): Suppose the $\mathcal{R}_\alpha$ hypotheses fail. By (b) either $root_\alpha \uparrow$ at almost every stage or it receives a final value. If the former holds then we are done, so suppose $root_\alpha$ is stable with value σ, and let $T(\sigma)$ be the final position of $T_s(\sigma)$. Suppose for a contradiction that α acts infinitely often. Hence $\alpha.2$(c) has to be taken infinitely often, otherwise $|\sigma_\alpha|$ is bounded and so $\alpha.2$(b) is taken finitely often. Hence for each k, $root^k_\alpha$ must receive a final value, call it σ_k.

Suppose first of all that Z_α does not exist. Then there is some least i such that for each n, $Z_\alpha(i)[t] = n$ for infinitely many α-stages t. Since $|T(\sigma_i)| > i$, then after $T(\sigma_{i+1})$ is stable Z_α has to change at position i, and we will kill α under $\alpha.2$(a), contradicting the fact that $root_\alpha$ is stable. Next suppose that Z_α exists and is not in P. Then at almost every visit to α, $Z_\alpha[t]$ is not on $[T_t]$. Consequently $\alpha.2$(c) is taken finitely often. Now suppose that Z_α exists and is in P, but $\Phi^{Z_\alpha}_\alpha(i) \uparrow$. At substage t we will set $root^i_\alpha$ to have its final value. At the end of t we must have $\Phi^{T(root^i_\alpha)}_\alpha(i)[t] \downarrow$ and $Z_\alpha[t] \supset T(root^i_\alpha)$. By assumption Z has to change below $T(root^i_\alpha)$, and $T(root^i_\alpha)$ will retain its value until then. This means that when α is next visited it will be killed giving a contradiction. Finally suppose $\Phi^{Z_\alpha}_\alpha$ is total but not a modulus of convergence. Hence for some i, $Z_\alpha \upharpoonright i$ has to change after $\Phi^{Z_\alpha}_\alpha(i)$. As above at the end of substage t we note that $root^i_\alpha$ is correct. Hence Z has to change below $T(root^i_\alpha)$ after t, because otherwise $\Phi^{Z_\alpha}_\alpha(i) = \Phi^{T(root^i_\alpha)[t]}_\alpha(i)$, contradicting the fact that $Z_\alpha \upharpoonright i$ has to change after $\Phi^{Z_\alpha}_\alpha(i)$. Hence at the next visit to α we will kill α, another contradiction. □

Lemma 2.4. *Let α be on TP.*

(a) *If the $\mathcal{R}_\alpha$ hypotheses hold and $\alpha * f \subset TP$ then for each k, $root^k_\alpha$ will be defined permanently and $T(root^k_\alpha) \subset Z_\alpha$.*
(b) *$\mathcal{R}_\alpha$ is satisfied.*

Proof. Fix α on TP, and assume that (a) and (b) holds for every $\beta \subset \alpha$. Suppose the $\mathcal{R}_\alpha$ hypotheses hold. Consider the case when $\alpha * \infty \subset TP$. Since (a) holds trivially we show (b) holds for α. We have that α is either initialized externally infinitely often, or there are infinitely many α-expansionary stages. Suppose the former holds. Let β be a node such that $\beta * f \subseteq \alpha$ and β initializes α infinitely often. By Lemma 2.3(d) the $\mathcal{R}_\beta$ hypotheses must hold. If $Z_\alpha = Z_\beta$ then $Z_\alpha = Z_\beta \equiv_T C$, and we are done. Otherwise suppose $Z_\alpha(i) \neq Z_\beta(i)$ for some i. Without loss of

generality let $Z_\alpha \supset \rho * 0$ and $Z_\beta \supset \rho * 1$. Hence at almost every visit to α we have $Z_\alpha[t] \supset \rho * 0$ and $Z_\beta[t] \supset \rho * 1$. Take a sufficiently large t where β initializes α. The actions of β at t must move some string η which is comparable with $root_\alpha$. We may assume that $|\eta| > i+1$ and $|root_\alpha| > i+1$. By Lemma 2.3(b) we may assume that this action is either β.2(b) or β.2(c). Since $|root_\alpha| > i+1$ we may assume that $T(root_\alpha)[t-1] \supset \rho * 0$. Since η is comparable with $root_\alpha$ we also have $T((\eta^-)^-)[t-1] \supset \rho * 0$. Since $Z_\beta[t] \supset \rho * 1$ this means that at t, β must have chosen to kill instead, a contradiction.

Suppose instead there are infinitely many α-expansionary stages. Let $\beta * f \subseteq \alpha$ be such that $\ell_\alpha(\beta)[t] > \max\{\ell_\alpha(\beta)[t'] : \alpha * \infty \text{ is visited at } t' < t\}$ for infinitely many t in which $\alpha * \infty$ is visited. Hence $\limsup \ell_\alpha(\beta) = \infty$, and by Lemma 2.3(d) the $\mathcal{R}_\beta$ hypotheses must hold and $Z_\alpha = Z_\beta$. Since $\mathcal{R}_\beta$ is satisfied we have $Z_\alpha = Z_\beta \equiv_T C$.

Now we consider the case when $\alpha * f \subset TP$. We first show (a) for α. Clearly α is only initialized finitely often. We first claim that $root_\alpha$ is defined permanently at some stage. Pick a sufficiently large substage t, and we claim that $root_\alpha$ will be defined at t (and hence retain this value permanently). The only thing preventing us from finding a suitable η under α.1 is the requirement of being β-good. If β is to the left of α or extends $\alpha * \infty$, then β acts finitely often and so σ_β varies finitely and will not be a problem. If $\beta * f \subseteq \alpha$ and the $\mathcal{R}_\beta$ hypotheses fail then β also acts finitely often. If the $\mathcal{R}_\beta$ hypotheses hold and $Z_\beta = Z_\alpha$ then by induction hypothesis (a) on β, we have $\ell_\alpha \to \infty$ contradicting Lemma 2.3(c). On the other hand if $Z_\beta \neq Z_\alpha$ then $\sigma_\beta \not\subset Z_\alpha[t']$ holds at almost every t', and will not be a problem to defining η. Hence we will be able to define $root_\alpha$ permanently. It is clear that $Z_\alpha \supset T(root_\alpha)$, because otherwise $Z_\alpha \cap T(root_\alpha) \subseteq T(root_\alpha{}^-)$ since $Z_\alpha \in P$. Consequently we will kill α contradicting the fact that $root_\alpha$ is stable. It is then easy to verify (inductively on k) that for each $k > 0$, $root^k_\alpha$ is defined permanently and $T(root^k_\alpha) \subset Z_\alpha$.

We now verify (b) for α, i.e. $C \equiv_T Z_\alpha$. To compute $C(k)$, find t large enough such that $T(root^{k+2}_\alpha)[t] \subset Z_\alpha[t]$, and Z_α is correct up to $|T(root^{k+2}_\alpha)[t]|$. If k enters C after t, then at the next visit to α we will remove the old value $T(root^{k+2}_\alpha)[t]$ from P, which contradicts $Z_\alpha \in P$. Hence $k \in C$ iff $k \in C[t]$. Now to compute $Z_\alpha \upharpoonright i$ from C, we look for t large enough such that α is visited, $T(root^{i+1}_\alpha)[t] \subset Z_\alpha[t]$ and $C[t] \upharpoonright i+1$ is stable. Since no coding of $C \upharpoonright i+1$ takes place after t, it follows that $root^{i+1}_\alpha[t]$ is stable. By Lemma 2.2(b) $root^{i+1}_\alpha[t]$ is never moved after t. Consequently $T(root^{i+1}_\alpha)[t]$ is stable, and since $|T(root^i_\alpha)| > i$ we conclude that $Z_\alpha[t] \upharpoonright i = Z_\alpha \upharpoonright i$, because otherwise we would kill α after t. □

This ends the proof of the theorem. We note that the construction of P from $\mathbf{a}$ is effective. □

3. Realizable index sets

In this section we investigate which index sets can be realized. The proof of Theorem 2.1 shows that any computable sequence of c.e. degrees containing $\mathbf{0}$ can be realized in a perfect Π^0_1 class; the uniformity of the proof of Theorem 2.1 allows us to produce a class P_i realising each degree $\mathbf{a_i}$ in a computable set S, and we can realise $S \cup \{\mathbf{0}\}$ by taking the class $\{0^i * P_i\}_{i\in\omega}$. We wish to improve on this. In Theorem 3.1, we show that generally any index set generated by a Σ^0_3 set S can be realized, even if S does not contain an index for $\emptyset$. If S contains an index for $\emptyset$ then (Corollary 3.4) we can also realize $G(S)$ in a rank 2 class. Note that this is

the best possible rank, since a rank 1 Π^0_1 class only realizes finitely many degrees. If S is not required to contain $\emptyset$ and is non-empty, then we can always realize $G(S)$ by a perfect Π^0_1 class. We also give an example of a Π^0_3 set S where $G(S)$ cannot be realized in any Π^0_1 class (Theorem 3.5).

Theorem 3.1. *For any non-empty Σ^0_3 set S, there is a perfect Π^0_1 class P such that $W[P] = G(S)$.*

Proof. Let $V_{e,i}$ be a uniform sequence of c.e. sets such that $S = \{e : \exists i |V_{e,i}| = \infty\}$. Fix an $e_0 \in S$, and let T_{j_0} be the computable tree that gives $W[T_{j_0}] = G(\{e_0\})$ in Theorem 2.1. If $\sigma_0 \subset \sigma_1 \subset \cdots \subset \sigma_n$ is a sequence of finite strings, and f is a total computable function, we define the tree $Tree(f, \sigma_0, \cdots, \sigma_n)$ to be the tree which is the result of copying $T_{f(n)}$ above $\sigma_n * 0, \sigma_n * 1$ and copying $T_{f(m)}$ above $\sigma_m * 1$ for each $\sigma_m * 0 \subseteq \sigma_n$. Formally take the downwards closure of the set of strings $\{\sigma_n * 0 * T_{f(n)}\} \cup \{\sigma_n * 1 * T_{f(n)}\} \cup \{\sigma_m * 1 * T_{f(m)} : m < n \text{ and } \sigma_m * 0 \subseteq \sigma_n\}$. We can extend this to an infinite σ-sequence $Tree(f, \sigma_0, \sigma_1, \cdots)$ in a natural way, by copying $T_{f(m)}$ above $\sigma_m * 1$ for each m such that $\sigma_m * 0 \subseteq \sigma_{m+1}$. The tree $Tree(f, \sigma_1, \cdots)$ might not be computable unless the sequence of σ_m is computable.

In the following we fix $\langle f, e, i, \rangle$, where $e, i \in \mathbb{N}$ and f is a total computable function. We describe the construction of a Π^0_1 class $P_{f,e,i}$ (the relationship $\langle e, i, f\rangle \mapsto P_{f,e,i}$ is an effective one). For ease of notation we temporarily drop f, e, i. We build P by defining a computable tree U, in stages. At stage s we define U up till level s. To make U computable we only enumerate strings of length s at stage s. We also define a sequence of finite strings $\sigma_{0,s} \subset \sigma_{1,s} \subset \cdots$. The intention is that if $|V| = \infty$ then $\cup_n \lim_s \sigma_{n,s}$ will be a path in $P = [U]$ which is of the same Turing degree as W_e, with all other paths in P having degrees of paths occuring in the $T_{f(n)}$

If $|V| < \infty$, we still make sure that P is perfect, now containing only paths whose degrees are realized in the $T_{f(n)}$. We do this by aiming to make $P = [Tree(f, \lim \sigma_0, \lim \sigma_1, \cdots)]$, by making $U = Tree(f, \lim \sigma_0, \lim \sigma_1, \cdots)$ except on the dead ends introduced by the coding of W_e.

The plan: The idea is to break down the membership question $e \in S$ into infinitely many Π^0_2 questions (about the cardinality of $V_{e,i}$). We approximate the truth of $e \in S$ locally by examining if $|V_{e,i}| = \infty$, and build a class $P_{e,i}$. We use $P_{e,i}$ to denote $P_{f,e,i}$ when f is simply the function $f(x) = j_0$. At the end we combine the different Π^0_1 classes $P_{e,i}$ together. Locally if $|V_{e,i}| = \infty$, then we must make some $\alpha \in P_{e,i}$ such that $\alpha \equiv_T W_e$. The standard way of doing this is to make α the complement of the retraceable Π^0_1 set associated with W_e; we refer the reader to Cenzer, Downey, Jockusch and Shore [2], where they showed that $\{\boldsymbol{w}, \mathbf{0}\}$ is realizable for any $\boldsymbol{w}$. The problem is that this gives a rank 1 class in which some $\alpha \equiv_T W_e$ sits, and contains computable (in fact, isolated) paths. Inside $P_{e,i}$ we have to be careful not to introduce c.e. degrees which are not in $G(S)$. The only reason why the isolated paths show up is because we have to allow a different path to be extendible in the tree; this is to allow for the later coding of some $n \in W_e$. Observe that for the purpose of making $\alpha \equiv_T W_e$, it would not matter if we keep many paths (associated with $n \in W_e$) extendible on the tree instead of only two. Hence if α_s is the current approximation of an initial segment of α, we would copy the tree T_{j_0} above every node distinct from α_s which we currently want to keep extendible. If later on some m enters W_e and we need to switch α_{s+1} to a different string σ, then we simply take the leftmost extension of σ of length s. On the other

hand if σ is such a string which is never used, then we will copy the entire tree T_{j_0} above σ. Doing so puts $deg(W_{e_0})$ into $W[P_{e,i}]$ – no other c.e. degree is introduced. Of course we only extend α_s if new numbers enter $V_{e,i}$. If $|V_{e,i}| < \infty$ then we will copy T_{j_0} above every terminal node.

Following the plan above gives effectively the class $P_{e,i}$ for each pair e, i, which introduces no new c.e. degree and contains $deg(W_e)$ iff $|V_{e,i}| = \infty$. Finally we want to glue the separate $P_{e,i}$'s into a single class P. A slight technical issue arises if we take the simple amalgamation $P = \{0^n 1 * P_n : n \in \mathbb{N}\}$; even though none of the P_n introduces new c.e. degrees, however the support $X = \emptyset$ will be in P. To get around this problem we nest the construction of each P_x within the outer construction of P_{e_0}. This will create X in P with Turing degree $X \equiv_T W_{e_0}$, which supports the trees associated with $P_0, P_1, \cdots$.

Construction of U: at stage 1 declare $\sigma_0 = \langle\rangle$ and $\sigma_1, \sigma_2, \cdots$ undefined, and enumerate $\langle\rangle, 0, 1$ into U. For a $\sigma \in U$, when we say we *copy the next level of T_j above σ* at stage s we mean the following. For each $\sigma * \tau \in U$ such that $|\sigma * \tau| = s-1$, we enumerate $\sigma * \tau * 0$ into U if $\tau * 0$ is in T_j; similarly for $\sigma * \tau * 1$. Clearly if $\sigma \in U_{|\sigma|}$ and we issue this command for σ at every stage after $|\sigma|$, then T_j will be copied successfully above σ; i.e. $U \cap \{\tau : \tau \supseteq \sigma\} = \sigma * T_j$. A string is said to *require copying* at stage s, if it is either of the form $\sigma_m * 1$ for some convergent $\sigma_m * 0 \subseteq \sigma_{m+1}$, or it is $\sigma_m * 0$ or $\sigma_m * 1$ for the maximal convergent σ_m. When we say we copy T_f above such a σ we mean that we copy the next level of $T_{f(m)}$ for the corresponding m. The actions at stage $s+1$ involve the following. Look for the largest n such that $\sigma_n \downarrow$. See if there is some least $2m < n$ such that $m \in W_{e,s+1} - W_{e,s}$:

(i) m exists. Let τ be the leftmost string of length s extending $\sigma_{2m} * 1$, such that $\tau \in U$. Move $\sigma_{2m+1} = \tau$ and put both $\tau * 0, \tau * 1$ in U. Declare $\sigma_{2m+2}, \sigma_{m+3}, \cdots$ undefined.

(ii) m does not exists. If $s+1 \in V$ we let τ be the leftmost string of length s extending $\sigma_n * r$ which is in U. Here $r = 0$ if n is odd, and $r = W_{e,s+1}(\frac{1}{2}n)$ if n is even. Define $\sigma_{n+1} = \tau$ and enumerate both $\tau * 0, \tau * 1$ into U, otherwise if $s+1 \notin V$ do nothing.

Copy T_f above every string which needs copying, and go to the next stage. This ends the construction. It is easy to see that U is a computable tree.

Lemma 3.2. *If $|V| < \infty$ then $[U] = [Tree(f, \tilde{\sigma}_0, \cdots, \tilde{\sigma}_n)]$ for a finite collection of strings.*

Proof. Let n be the largest such that $\tilde{\sigma}_n = \lim_s \sigma_{n,s}$ exists. It is easy to check that $[U] \subseteq [Tree(f, \tilde{\sigma}_0, \cdots, \tilde{\sigma}_n)]$, since after a large enough stage the only thing we do in the construction is to copy T_f above a string which require copying. To see the reverse inclusion, fix an $m < n$ and consider the stage s where $\tilde{\sigma}_m$ is picked as $\sigma_{m,s}$. Then $|\tilde{\sigma}_m| = s - 1$ and we may assume that $(\tilde{\sigma}_m) * 0 \subseteq \tilde{\sigma}_n$. Since $(\tilde{\sigma}_m) * 1 \in U_s$ it follows that we will successfully copy $T_{f(m)}$ above $(\tilde{\sigma}_m) * 1$. A similar argument follows for $m = n$. □

Lemma 3.3. *If $|V| = \infty$ and $[T_{f(m)}] \neq \emptyset$ for every m, then $\tilde{\sigma}_m = \lim_s \sigma_{m,s}$ exists for every m, and $\cup_m \tilde{\sigma}_m \equiv_T W_e$. Furthermore $[U] = [Tree(f, \tilde{\sigma}_0, \tilde{\sigma}_1, \cdots)]$, and $(\tilde{\sigma}_{2m+1}) * 0 \subseteq \tilde{\sigma}_{2m+2}$ for all m.*

Proof. It is straightforward to verify that the search for τ in steps (i) and (ii) of the construction are always successful, and consequently $\lim_s \sigma_{m,s}$ exists for

every m. Observe that $X = \cup_m \tilde{\sigma}_m$ can compute the sequence $\tilde{\sigma}_0, \tilde{\sigma}_1, \cdots$, and we have $m \in W_e \Leftrightarrow X(|\tilde{\sigma}_{2m}|) = 1$. To see that $X \leq_T W_e$, fix n and compute a stage s large enough so that $W_{e,s} \upharpoonright n+1 = W_e \upharpoonright n+1$ and all of $\sigma_{0,s}, \cdots, \sigma_{n,s}$ are defined. These must already be at their final values, and since $n < \sigma_{n,s}$ it follows that $n \in X$ iff $\sigma_{n,s}(n) = 1$. To prove the last statement, observe that $[U] \supseteq [Tree(f, \tilde{\sigma}_0, \tilde{\sigma}_1, \cdots)]$ follows from a similar argument as in Lemma 3.2. To see that $[U] \subseteq [Tree(f, \tilde{\sigma}_0, \tilde{\sigma}_1, \cdots)]$, observe that any $Z \in [U]$ satisfies either $Z = X$ or else there is some m such that $Z \supset (\tilde{\sigma}_m) * 1$ and $X \supset (\tilde{\sigma}_m) * 0$. Finally note that $(\tilde{\sigma}_{2m+1}) * 0 \subseteq \tilde{\sigma}_{2m+2}$ clearly holds for all m. □

We are ready to show that there is a perfect Π^0_1 class P such that $W[P] = G(S)$. Let $f(x) = j_0$ for every x; we want to combine all the classes $P_{f,e,i}$. Let g be a computable function defined by the following: $g(2x) = j_0$ and for the odd inputs we select an index such that $[T_{g(2\langle e,i\rangle+1)}] = P_{f,e,i}$. From Lemmas 3.2 and 3.3, and the fact that T_{j_0} is perfect, we may conclude that each $T_{g(x)}$ is perfect. Since $e_0 \in S$ pick i_0 such that $|V_{e_0,i_0}| = \infty$, and let $P = P_{g,e_0,i_0}$. It is not hard to verify that P itself is also perfect, and $W[P] = G(S)$. □

Corollary 3.4. *Given a Σ^0_3 set S that contains an index for $\emptyset$, there is a Π^0_1 class P of rank 2 such that $W[P] = G(S)$.*

Proof. Suppose S contains an index for $\emptyset$. We want to produce P of rank 2. We run the argument in the proof of Theorem 3.1, except now we let $T_{j_0} = \{0^m : m \in \mathbb{N}\}$, and set $f(x) = j_0$ for all x. Each of the $P_{f,e,i}$ has rank at most 1 - if $V_{e,i}$ is finite then so is $P_{f,e,i}$ and if $V_{e,i}$ is infinite then $P_{f,e,i}$ has rank 1. We let $P = \{0^{\langle e,i\rangle} 1 * P_{f,e,i} : \langle e,i\rangle \in \mathbb{N}\}$, which is of rank at most 2, and $W[P] = G(S)$. □

We now turn to the question of whether every Σ^0_4 set can be realized. An obvious counter-example is the Π^0_3 index set $\{e : W_e \text{ is not computable}\}$. In fact no Π^0_3 set which is downwards dense in the c.e. degrees (in the sense that for any c.e. $W >_T \emptyset$ there is $e \in S$ such that $W_e \leq_T W$), and containing no computable set can be realized, due to an old result of Jockusch and Soare [8]. In the following theorem we give another example of a Π^0_3 set which cannot be realized, but which *contains* an index for $\emptyset$.

Theorem 3.5. *There is a Π^0_3 set S containing an index for $\emptyset$, such that there is no Π^0_1 class P with $W[P] = G(S)$.*

Proof. For any c.e. set $L >_T \emptyset$, one can effectively obtain a c.e. set D and reduction Ψ such that $D = \Psi^L$ and $D > \emptyset$ and $D \not\geq_T L$. If we iterate this process starting with some noncomputable low_2 c.e. set L, we get computable increasing functions g and ℓ such that $L = W_{g(0)} >_T W_{g(1)} >_T W_{g(2)} >_T \cdots$, and $W_{g(n)} = \Phi^L_{\ell(n)}$ for all n. Let T_e be the e^{th} primitive recursive tree. Since the $W_{g(e)}$ are uniformly computable from L via ℓ, it follows that there are L-computable relations R_1 and R_2 such that

$$R_1(e,i,x,s) \Leftrightarrow \Phi_i^{W_{g(e)}} \upharpoonright x[s]\downarrow \text{ and is on } T_e,$$

$$R_2(e,i,j,x,m,s,\sigma) \Leftrightarrow \Phi_i^{W_{g(e)}} \upharpoonright m[s]\downarrow = \sigma \text{ and } \Phi_j^{\sigma} \upharpoonright x[s]\downarrow = W_{g(e)} \upharpoonright x.$$

We define $n \notin S$ iff whenever $n = g(e)$, then there are some i, j for which

(i) $\Phi_i^{W_{g(e)}}$ is total and is in $[T_e]$, and
(ii) $\Phi_j(\Phi_i^{W_{g(e)}})$ is total and equals $W_{g(e)}$.

The quantifier for e is bounded since g is increasing. Clause (i) can be expressed as $\forall x \exists s R_1(e, i, x, s)$. Since this is $\Pi^0_2(L)$, and L is low_2 it is also Σ^0_3. Similarly clause (ii) can be expressed as $\forall x \exists m \exists s \exists \sigma R_2(e, i, j, x, m, s, \sigma)$ which is also Σ^0_3. Hence S is Π^0_3, and it is easy to verify that there can be no Π^0_1 class P such that $W[P] = G(S)$. To see that S can be made to contain $\emptyset$, note that $S \cup \tilde{S}$ is also good for any Π^0_3 set $\tilde{S}$ where $G(\tilde{S}) \cap G(W_{g(n)}) = \emptyset$ for every n. □

Note that S is clearly non-empty, since empty index sets are realized by the empty Π^0_1 class. The above counterexample relies on the fact that for a low_2 set L, $\Pi^0_2(L) = \Sigma^0_3$.

So far all the examples we have of a set S that can be realised are those with a Σ^0_3 definition (such as singletons). We wish to show that some S can be realized which has no Σ^0_3 definition. The next result does this.

If A is a high set then by Martin [9], there is an A-computable function Γ^A, such that which dominates every computable function, meaning that if f is computable, then for almost all x, $\Gamma^A(x) > f(x)$. Thus, if A is high then some $\Gamma = \Phi_j$ satisfying this will exist. We will call such a j a *domination index* for A. As usual, the use of $\Gamma^A(x)[s]$ is denoted by $\gamma(x, s)$.

Theorem 3.6. *Consider a computable collection $\mathcal{C}$ of pairs of the form $\langle e, j(e)\rangle$, where j is a computable function such that each W_e has high degree, and $j(e)$ is a domination index for W_e. Let S be any Σ^0_4 subset of $\mathcal{C}$, and fix an index e_0 for $\emptyset$. Then there is a rank 2 Π^0_1 class P with such that $W[P] = G(S \cup \{e_0\})$.*

In particular, if S is Σ^0_4-complete, then $S \cup \{e_0\}$ is an example of a realizable set which is not Σ^0_3.

Proof. For simplicity of notation, we will pretend that $\mathcal{C} = \omega$ and also write $\Phi_{j(e)} = \Gamma_e$ for the domination function for e.

In Theorem 3.1, we realized each Σ^0_3 set of e's. Here we will be dealing with a Σ^0_4 set S. Thus, $e \in S$ iff $\exists p \forall m \exists s \forall t R(e, p, m, s, t)$. For a fixed p, should the Π^0_3 condition $\forall m \exists s \forall t R(e, p, m, s, t)$ hold, we need to code W_e.

Again, the idea will be to devote a unique part of the Π^0_1 class, to the pair $\langle e, p\rangle$. These indices indicate that we are concerned with W_e, (with domination function $g_e^{W_e}$), and p is the Σ^0_4 witness.

We will be making the class to be of rank 2. Concentrating on this fixed pair, $\langle e, p\rangle$ we are thinking of the part of the class being a rank 1 subclass extending some fixed $\sigma * 1 = \nu$, say. The idea is again to more-or-less follow the coding of Cenzer, Downey, Jockusch and Shore [2], where they showed that $\{\mathbf{w}, \mathbf{0}\}$ is realizable for any $\mathbf{w}$. In this construction, it is convenient to allow $\langle m, 0\rangle$ to correspond to the (initial) coding location for "$m \in W_{e,s}$". That is, initially we will think of $\nu * 0^{\langle m,j\rangle}$ as the coding location $c(\langle m, j\rangle, s)$; and in the construction, $c(\langle m, 0\rangle, s)$ for "$m \in W_e$". Should m enter $W_{e,s}$, and conditions are opportune, we would code by changing this to $\nu * 0^{\langle m,0\rangle - 1} * 1^{s - \langle m,0\rangle} * 0^i$ and re-assign the coding location for $\langle m', j'\rangle$ to be $c(\langle m', j'\rangle, s)$ accordingly for $\langle m, 0\rangle < \langle m', j'\rangle$. (Of course all other paths in this subclass are isolated ones of the form $\rho * 1^\omega$, as in Theorem 3.1.) Again this makes a rank 1 subclass extending ν.

As we see below, there are other reasons we might move a coding location. In the construction, we would only allow $c(n, s)$ to move if $W_{e,s} \upharpoonright \gamma(n, s)$ changes. This is to keep the degree *below* that of W_e. Without loss of generality we will assume that for $i \le s$, $\gamma(i, s), \Gamma^{W_e,s}(i) \downarrow [s] < s$.

We only want to code W_e if Σ^0_4 is correct. We do *not* want to code $W - e$ should the Π^0_4 outcome be correct. Thus, each time Σ^0_3 of the triple corresponding to e, p, m in $\neg R(e, p, m, s, t)$ looks correct, what we would like to do is to *move* the coding location for $c(\langle m + i, 0\rangle s)$ for $i \geq 0$ spontaneously to $\nu * 0^{\langle m,0\rangle - 1} * 1^{s - \langle m,0\rangle} * 0^i$. We call this action *kicking*. Then, if $e \notin S$, so that the Π^0_4 outcome is correct and has witness m for this p, eventually all the paths in in the class extending ν will be computable, as they will be driven to infinity. If $e \in S$, and p is the correct Σ^0_4 witness, then since each triple e, p, m has a Σ^0_2 witness, for each m the above process will fire only finitely often, and hence only move the coding location for $m \in W_e$ a finite number of times.

Now, of course there is a small problem with all of this, as it upsets the computation of the coding locations *from* W_e. In the Σ^0_3 case in Theorem 3.1, locations $c(m, s)$ only moved when $m' \leq m$ entered $W_{e,s'}$ for some $s' \leq s$.

The solution is to use the highness of W_e. At the same time as the above, we will be monitoring $\Gamma_e^{W_e,s}(k)$ for $k \leq s$, and building a sequence of potential computable functions f_k. The reader should note that if the Π^0_4 outcome is true, it is enough the force almost all coding markers to infinity. That is, even if $\langle e, p, m\rangle$ is the witness of the failure of the Σ^0_4 condition, it is enough that some $c(n', s) \to \infty$ for some $' \geq \langle m, 0\rangle$.

So suppose that the we have a fixed e, p, m, and m seems to be a witness $\forall s \exists t \neg R(e, p, m, s, t)$ because this fires. At this stage we would define $f_{\langle e,p,m\rangle}(\langle m, 0\rangle)$[2] to be large and fresh, and continue to do this for $f_{\langle e,p,m\rangle}(\langle m, k\rangle)$ (i.e. for more and more k each time it continues to fire) until we see some W_e change on $\gamma(\langle m, k\rangle)$ to make $g_e^{W_e}(\langle m, k\rangle) > f_{\langle e,p,m\rangle}(\langle m, k\rangle)$, which it must as Γ^{W_e} is dominant, and $f_{\langle e,p,m\rangle}$ will be total if it fires infinitely often. Note that in the construction, for each k where we get a W_e change on $\gamma(\langle m, k\rangle)$ at some $t \geq s$, we would kick the coding markers $c(z, t)$ for $z \geq \langle m, k\rangle$. *Additionally, we would kick all coding markers* $c(\langle m', q\rangle, s)$ *for* $m' > m, k$ *to also extend* $c(\langle m, s\rangle t + 1)$. Here s is the stage where we first saw a Γ^{W_e} change for some $\langle m, k\rangle$, and hence this is last defined value of $f_{\langle e,p,m\rangle}$. Notice that now the original ordering of the coding markers is no longer valid. If we need to kick for m because the Π^0_4 outcome seems correct, then we will move *all* coding markers for $m' > m, k$, and they will now be above $c(\langle m, s + 1\rangle, t + 1)$. If the process begins anew, and k was the place where Γ^{W_e} is dominant above $\langle m, k\rangle$, then if it fires again, all coding markers corresponding to $m' > m, k$ will again be moved as we'd get a change of W_e on $\gamma(\langle m, s + 1\rangle, t + 1)$.

The key point is that *if* $\langle e, p, m\rangle$ fires infinitely often, than for almost all m' coding markers corresponding to m' will be driven to infinity, as required. The remaining details work as in the Σ^0_3 case. □

4. Characterizing the sets which can be realized in a Π^0_1 class and computable representations of index sets

Recall that for $S \subseteq \omega$, we defined $G(S) = \{e \mid W_e \equiv_T W_i \text{ for some } i \in S\}$. We say that a set $S \subseteq \omega$ *represents* a set of c.e. degrees D if $G(S) = \{e \mid deg(W_e) \in D\}$. Theorem 3.1 says that every Σ^0_3 representable set of c.e. degrees is represented in some Π^0_1 class. We show that in fact the converse holds; that every set of c.e. degrees realized by a Π^0_1 class has a computable representation. Hence the index sets which

[2]Strictly speaking, we would be defining $f_{\langle e,p,m\rangle}(z)$ for $z \leq \langle m, 0\rangle$ to make $f_{\langle e,p,m\rangle}$ total in the limit.

are realized by some Π^0_1 class are exactly those with a computable representation. In fact, we shall be able to classify the sets S with a computable representation. Notice that all $G(S)$ which are represented in some Π^0_1 class has complexity between Σ^0_3 and Σ^0_4. Condition (v) below describes the exact syntactic criterion for an index set to have a computable representation; the condition "$\exists a \forall b \exists c\ R^{W_n}(n, a, b, c)$" is clearly between Σ^0_3 and Σ^0_4 (in the sense that every Σ^0_3 predicate can be written in this form, and its natural complexity upperbound is Σ^0_4). Thus, we characterise the index sets with a computable representation as exactly those which are "Σ^0_3 relative to the index being tested".

Theorem 4.1. *Let $S \subseteq \omega$. The following are equivalent.*

(i) *S has a computable representation, that is, $G(S) = G(R)$ for some computable set R.*
(ii) *There is a Π^0_1 class P such that the set of c.e. degrees represented in P has index set $G(S)$, that is, $W[P] = G(S)$.*
(iii) *There is a perfect Π^0_1 class P such that $W[P] = G(S)$.*
(iv) *There is a computable function g such that for every n,*

$$n \in G(S) \Leftrightarrow W^{W_n}_{g(n)} \text{ is cofinite.}$$

(v) *There is a truth-table functional R such that for every n,*

$$n \in G(S) \Leftrightarrow \exists a \forall b \exists c\ R^{W_n}(n, a, b, c).$$

Proof. (iv) $\Rightarrow$ (v): It is easy to see that (iv) is equivalent to the condition "there is a computable function $\hat{g}$ such that for every n, $n \in G(S) \Leftrightarrow \Phi^{W_n}_{\hat{g}(n,m)}$ is total for some m". For any inputs X, n, a, b and c, let $R^X(n, a, b, c)$ hold iff $\Phi^{W_n}_{\hat{g}(n,a)}(b)[c] \downarrow$ with use u and $W_n[s] \upharpoonright u \subset X$. Then R^X is clearly total and X-computable for all X, and has the required property.

(v) $\Rightarrow$ (iv): Let $\Phi^X_{\hat{g}(n,a)}(b) \downarrow$ iff $R^X(n, a, b, c)$ holds for some c.

(ii) $\Rightarrow$ (v): The predicate within square brackets in the proof of Proposition 1.3 is computable in W_e. As above, this is obviously witnessed by a truth-table functional by comparing $W_e[s]$ with the oracle.

(i) $\Rightarrow$ (iii): This is Theorem 3.1.

(iv) $\Rightarrow$ (i): Fix $\hat{g}$ such that for every e, $e \in G(S) \Leftrightarrow \Phi^{W_e}_{\hat{g}(e,q)}$ is total for some q. If $S = \emptyset$ then there is nothing to do, so fix a c.e. set D with an index in S.

For each pair e, q such that $e, q \in \omega$, we build (effectively) a c.e. set $V_{e,q}$. At the end we will take R to be the set of all indices for the sets $V_{e,q}$ (ranging over all pairs e, q) and show that R represents $G(S)$. R is of course computable.

Fix a pair e, q. We describe how to build $V = V_{e,q}$. We also define markers $m_0[s] < m_1[s] < \cdots$ for V. At each stage s define $n_i[s] = \varphi^{W_e}_{\hat{g}(e,q)}(i)[s]$, if this converges. We assume that $n_{i+1}[s] \downarrow \Rightarrow n_i[s] \downarrow$ and is less than $n_{i+1}[s]$. We shall ensure that $m_k[s] \downarrow$ if and only if $n_k[s] \downarrow$ for all $k < s$. At stage 0 set all our parameters undefined and set $V = \emptyset$. Now assume we are at stage $s + 1$. Let $z \in W_e[s+1] - W_e[s]$ be the least such, if it exists. If $z < n_k[s]$, pick the least such k with $n_k[s] \downarrow$ and enumerate $m_k[s]$ into V. Note that $n_0[s+1], \cdots, n_{k-1}[s+1]$ are all still defined since there has been no W_e change below them, and therefore $m_0, \cdots, m_{k-1}$ are all still defined. For every number x larger than m_{k-1} that has been assigned to code $D(i)$ for some i, we set $D(i) = V(x)$. See if $n_k[s+1] \downarrow$, and

if so set m_k to be a large fresh number, and assign a new number less than m_k to code the next $D(i)$. Otherwise set $m_k \uparrow$.

If z does not exist, or if z is larger than any $n_k[s]$ seen at the previous stage s, we do the following. For the least k such that $n_k[s] \uparrow$, and for every number x larger than m_{k-1} that has been assigned to code $D(i)$ for some i, we set $D(i) = V(x)$. Assign a new number to code the next $D(i)$. Check and see if $n_k[s+1] \downarrow$. If so, set m_k to be a large fresh number, otherwise do nothing else. This ends the definition of V.

Now we state some properties about V. There are two possibilities:

(I) $\Phi_{\hat{g}(e,q)}^{W_e}$ *is total.* Then $\lim_s n_k[s]$ and $\lim_s m_k[s]$ exist for every k. In this case it is easy to check that $W_e \equiv_T V$. To compute $W_e \leq_T V$, fix x and wait for a stage large enough such that $m_x[s] \downarrow$ and V is correct up to $m_x[s]$. Then $x \in W_e$ iff $x \in W_e[s]$. To compute $V \leq_T W_e$ we wait for a stage large enough such that $n_k[s] \downarrow$ and W_e is correct up to $n_k[s]$. This means that $m_k[s] \downarrow$ and the construction will never again change V below $m_k[s]$.

(II) $\Phi_{\hat{g}(e,q)}^{W_e}$ *is not total.* Then there must be some least k such that either $\lim_s n_k[s] = \lim_s m_k[s] = \infty$, or $n_k[s], m_k[s] \uparrow$ for almost every s. Let r be the final value of m_{k-1}. (If $k = 0$ set $r = 0$). In the first case there are infinitely many stages where m_k is enumerated into V, and at every such stage we will assign for every i, some number larger than r to code $D(i)$. We will also be able to ensure that $V(x) = D(i)$ for every $x > r$ assigned to code some $D(i)$. In the second case $m_k[s] \uparrow$ for almost all s, and the construction will attend to the coding of D above r at almost every stage. In either case, we see that every bit of V larger than r is either assigned to code some $D(i)$, or is never again modified after m_k is lifted past it. Thus, $V \equiv_T D$.

Now we verify that $G(R) = G(S)$. Suppose that $V_{e,q}$ is included in R. Then either (I) or (II) holds for the construction of $V_{e,q}$. In case (II) $V_{e,q} \equiv_T D$ and so $V_{e,q}$ is represented in $G(S)$. In case (I) $\Phi_{\hat{g}(e,q)}^{W_e}$ is total and $V_{e,q} \equiv_T W_e$. This also means that $V_{e,q}$ is represented in $G(S)$. Hence $R \subseteq G(S)$.

Now let $e \in G(S)$. Suppose q is such that $\Phi_{\hat{g}(e,q)}^{W_e}$ is total. But this means that case (I) holds for $V_{e,q}$ and the index of $V_{e,q}$ is in R. Hence $G(S) \subseteq G(R)$. □

Corollary 4.2. *If S is Σ_3^0 then there is some computable R such that $G(S) = G(R)$. However not every Π_3^0 set has a computable representation.*

Corollary 4.3. *An index set I is realisable in a Π_1^0 class iff I has a Σ_3^0 representation iff I has a computable representation.*

Corollary 4.4. *If S and T have computable representations then so do $G(S) \cap G(T)$ and $G(S) \cup G(T)$.*

Corollary 4.5. *The following classes of c.e. degrees have a computable representation and can be realized in some (perfect) Π_1^0 class:*

- *The set of all superlow c.e. degrees.*
- *The set of all K-trivial c.e. degrees.*
- *Any Σ_4^0 subset of a computable collection of high indices with an index for $\mathbf{0}$.*
- *Any uppercone of c.e. degrees, that is, the set $\{\boldsymbol{b} \mid \boldsymbol{b}$ is c.e. and $\boldsymbol{b} \geq \boldsymbol{a}\}$ for any c.e. degree $\boldsymbol{a}$.*

- *Any lowercone of c.e. degrees below a low$_2$ c.e. degree, that is, the set $\{\boldsymbol{b} \mid \boldsymbol{b}$ is c.e. and $\boldsymbol{b} \leq \boldsymbol{a}\}$ for any low$_2$ c.e. degree $\boldsymbol{a}$.*

Proof. The first two have obvious Σ^0_3 definitions. The third is just Theorem 3.6. For the fourth, let $A \in \boldsymbol{a}$ be a c.e. set. Then $W_n \geq_T A$ if and only if $\exists e \forall x \forall s \exists t > s\ \Phi_e^{W_n}(x)[t] \downarrow = A(x)[t]$ and $W_n[t] \upharpoonright \varphi_e(x)[t]$ is correct. For the fifth, note that $W_n \leq_T A$ if and only if $\exists e \left(\Phi_e^A \text{ is total and } \forall x \forall s \exists t > s\ \Phi_e^A(x)[t] \downarrow = W_n(x)\right)$. Since A is low$_2$, the statement "Φ_e^A is total" is Σ^0_3. □

Finally, we wish to consider which lowercone of c.e. degrees has a computable representation. From Corollary 4.5 we know that every low$_2$ lowercone has a computable representation. We will next show that not every lowercone of c.e. degrees has a computable representation, and will follow from the next theorem. The next result says that given any effective sequence of independent c.e. sets $A_0, A_1, \cdots$, and any Σ^0_4 set S, there is a c.e. set B that bounds A_i for all $i \in S$ and does not bound any A_i for $i \notin S$; roughly speaking, it says that the S-infinite join of any independent sequence exists, and we believe that this result is of independent interest.

Theorem 4.6. *Let C be a uniformly computable collection of c.e. sets $\{A_i \mid i \in \omega\}$ which are independent in that for any finite set F, if $i \notin F$, then $A_i \not\leq_T \oplus_{j \in F} A_j$. Let S be a Σ^0_4 set. Then there is a c.e. B such that for all $i \in \omega$, $i \in S$ iff $A_i \leq_T B$.*

Proof. The proof uses the $\mathbf{0''}$ methodology, even though each requirement shall be divided into infinitely many subrequirements, but there are no injuries along the true path. We need to meet for all e the following:

$$M_e : e \in S \to A_e \leq_T B,$$

$$N_e : e \notin S \to A_e \not\leq_T B.$$

To meet M_e, let R be a computable relation representing S in that

$$e \in S \text{ iff } \exists x \forall m \exists s \forall t R(e, x, m, s, t).$$

$M_{e,x}$ denotes the attempt to meet M_e at witness x. We set aside infinitely many potential columns $\{B^{\langle e,x,n\rangle} \mid n \in \omega\}$, for the sake of $M_{e,x}$. At each stage s, $n(e, x, s)$ denotes the current choice. As we will later see, we will in fact be using multiple positions to try to meet the same $\langle e, x\rangle$. We will ensure that $n^* = \liminf_s n(e, x, s)$ exists for at least one choice and this column $B^{\langle e,x,n^*\rangle}$ correctly codes A_e iff $e \in S$.

Description of the strategy. For a fixed choice $n(e, x, s)$, we do the following. For each number a, we will have a *coding location* $c(2a, s)$ which is coding whether $a \in A_i$. (The odd ones will be used for the Σ^0_4-testing.) If a enters $A_{i,s}$ put $\langle e, x, n(e, x, s), c(2a, s)\rangle$ into $B_{s+1} - B_s$. Unless restrained, each time $R(e, x, m, -, -)$ fires (for each m), we will kick all coding locations $c(a, e, x, s) = c(a, s)$ for $a \geq 2m+1$ to fresh places, dumping $\langle e, x, n(e, x, s), c(2m+1, s)\rangle \ldots \langle e, x, n(e, x, s), c(s, s)\rangle$ into $B_{s+1} - B_s$, and reassigning new coding locations for $a \geq 2m+1$ to be $c(2m+i, s+1) = c(s, s)+i$, for $i \geq 1$. This will obviously be modified below, to take into account of other requirements, but the actual action is more or less similar. Note that if x is not the Π^0_3 witness that $e \in S$, then for some m, the Π^0_2 fact $R(e, x, m, -, -)$ fires infinitely often, and hence, if this is a stable $n(e, x, s)$, almost all of the column $\omega^{\langle e,x,n(e,x,s)\rangle}$ will enter B.

To implement this strategy on a tree of strategies we will introduce $M_{e,x}$ at some mother node $\tau(e,x) = \tau$, and at various places in the cone below $\tau(e,x)$, we will have sub-requirements $\tau_m(e,x)$ testing whether $R(e,x,m,-,-)$ fires infinitely often, i.e. whether the predicate $\forall s \exists t R(e,x,m,s,t)$ holds. This node $\tau_m(e,x)$ will have outcomes $\infty <_L f$, corresponding to the truth value of the predicate $\forall s \exists t R(e,x,m,s,t)$. The way the mother node $\tau(e,x)$ and its children nodes $\tau_m(e,x)$ work is the following. The mother node $\tau(e,x)$ is assigned a column, say p of B. It wants to code A_e into column p of B if and only if the Π^0_3 predicate $\forall m \exists s \forall t \neg R(e,x,m,s,t)$. It measures this Π^0_3 predicate by distributing the task amongst its children nodes; as usual, each child $\tau_m(e,x)$ measures the Σ^0_2 fact $\exists s \forall t \neg R(e,x,m,s,t)$. The mother node $\tau(e,x)$ will assign coding location for each z, coding $A_e(z)$ into some position of $B^{\langle p \rangle}$. Its task whenever visited is to monitor if some $A_e(z)$ has changed since it last checked, and if so, put the appropriate coding marker into $B^{\langle p \rangle}$. It also assigns a new coding location for z if the previous one has been put into B. The task of each child node $\tau_m(e,x)$ is to monitor $\exists s \forall t \neg R(e,x,m,s,t)$. Every time the Π^0_2 instance holds it will work towards making $B^{\langle p \rangle}$ cofinite by picking some least coding location z (above some restraint) and enumerating everything in the column $B^{\langle p \rangle}$ between $c(2z+1)$ and s. Thus in the Π^0_3 case, each coding location is lifted finitely often by the children nodes and the mother node will successfully ensure $A_e \leq_T B^{\langle p \rangle}$. In the Σ^0_3 case, some child node will force $B^{\langle p \rangle}$ to be cofinite, and obviously in this case $A_e \not\leq_T B^{\langle p \rangle}$. Note that in the Σ^0_3 case, making $A_e \not\leq_T B^{\langle p \rangle}$ is not enough, as we in fact need $A_e \not\leq_T B$, but it is necessary for the N-requirements to work correctly.

Below outcome $\tau_m(e,x) * \infty$ we have confirmation that x is not the witness for $e \in S$. So we must make progress towards towards meeting the requirement N_e (i.e. making $A_e \not\leq_T B$) as described below. Note that we may not actually meet N_e at the end, since perhaps $e \in S$, but witnessed by some other $x' > x$. We will describe how this works in due course, but for the time being, we shall suppose that no such x' exist and that e is in fact not in S. Now, N_e is divided into infinitely many subrequirements $N_{e,k} : e \notin S \to \Phi^B_k \neq A_e$.

We first discuss this for $e = 0$ and $k = 0$. We will be assuming that $n(0,0,s) = 0$ for all s, so that A_0 is being coded into column $B^{\langle 0,0,0 \rangle}$ for all stages. Now, below some $\tau_m(0,0) * \infty$ (where recall that $\tau(0,0)$ is the mother node devoted to $M_{0,0}$) we will work as follows. We wish to make $\Phi^B_k \neq A_0$ for $k = 0$. The first case would be that $\tau_m(0,0)$ is immediately below the position where we introduced the mother node $\tau(0,0)$ and therefore $N_{0,0} = \tau_m(0,0) * \infty$ are the first two requirements below $\tau(0,0)$. The strategy for $N_{0,0}$ is relatively simple. We would allow $N_{0,0}$ to assert control of B by freezing $B \upharpoonright \varphi_k(z)$ at the first stage that we see a $B^{\langle 0,0,0 \rangle}$-correct computation such that $\Phi^B_k[s] \upharpoonright z = A_0[s] \upharpoonright z$. Then, since A_0 is noncomputable, this would only have finite restraint on the overall construction, since at some stage some small a will enter A_0 and create a disagreement, which will be preserved forever.

More generally, let us suppose that we are dealing with $\Phi^B_k \neq A_0$ and that the requirement in charge of this, $N_{0,k}$, is placed below, say, a coding column which is coding A_1. More specifically, we are now considering the situation where $N_{0,k}$ is below the node $\tau(1,0)$ (assigned to requirement $M_{1,0}$), which is in turn below $\tau_m(0,0) * \infty$.

So in this simplified set-up, $\Phi^B_k \neq A_0$ being met at σ will need to cope with, say, the coding of A_1 into some column of B (say column p), and it is not reasonable that this should be restrained by σ as the mother node $\tau(1,0)$ in charge of the coding of A_1 is at a higher priority place on the tree.

Now there are two possibilities. Either column p actually codes A_1 at the end (Π^0_3 outcome), or there is some m_1 which fires infinitely often for column p and A_1 (Σ^0_3-outcome). In the Π^0_3 outcome there are no difficulties with arranging for the coding of A_1 and $\tau(1,0)$ to be of higher priority than σ. Following the above strategy, σ will impose restraint on all requirements of lower priority as indicated, and the only injury would to σ will be caused by the A_1-coding done by $\tau(1,0)$. Since $A_0 \not\leq_T A_1$, the lim inf of the restraint would exist. Thus, on the tree of strategies, we will represent the possible outcomes as the sequence $w <_L 0 <_L 1 <_L 2 <_L 3 <_L \ldots$ with w representing the waiting outcome to see a valid length of agreement above 0, and the rest the lim inf of the length of agreement. Note that we are using the hat convention so that if a computation with length above n occurs and the use is injured, then the length of agreement must drop *below* n, for at least the next σ-stage. Naturally, we will need to prove one of these outcomes is on the true path assuming that σ is.

Our biggest problem come from the fact that the numbers entering B can either be caused by coding of some A_i set, *or* some m_1 trying to prove that some τ is incorrect. If this node σ trying to make $\Phi^B_k \neq A_0$ is below some $\tau_{m_1}(1,0) * \infty$, then σ knows that $\tau_{m_1}(1,0)$ is going to make column p of B cofinite. As usual, σ would wait for computations which are both $\tau_{m_1}(1,0) * \infty$ and $\tau_m(0,0) * \infty$ correct, and there is no real problem. However it could be that m_1 fires infinitely often, and σ is above all such $\tau_{m_1}(1,0) * \infty$-nodes. The decision of coding locations of A_1 inside column p of B is made at the mother node $\tau(1,0)$. However, as $\tau(1,0)$ is committed to coding A_1 into its assigned column of B if and only if some Π^0_3 predicate holds, the mother $\tau(1,0)$ will distribute this guess amongst its children nodes $\tau_k(1,0)$ for all k. Hence, in the Σ^0_3 case we will have to let some child node $\tau_{m_1}(1,0)$ make column p of B cofinite and move its corresponding coding location to ∞. If this child node $\tau_{m_1}(1,0)$ is below σ, then there is no way for σ to wait for a $\tau_{m_1}(1,0) * \infty$-correct computation.

In this case, the child node $\tau_{m_1}(1,0)$ has lower local priority than σ, and thus should defer to the wishes of σ. We will arrange for controlled injury of the column p in this case. When any k fires (i.e. $\tau_k(1,0)$ sees the next instance of Π^0_2 hold), if it is associated with some $\tau_k(1,0) * \infty \preceq \sigma$, then we let it act as usual, since σ must be looking for $\tau_k(1,0)$-correct computations anyway. However, if k is sufficiently large that it is handled below σ, the tree machinery will handle the injury. More specifically, $\tau_k(1,0)$ will be making column p of B coinfinite above the restraint r such that $\tau_k(1,0) \succeq \sigma * r$. If $\tau_k(1,0)$ is along the true path, then every time where it is visited, the restraint currently imposed by σ is less than r. Of course, if we move to some outcome $n < r$ of σ later on, then the restraint imposed by σ will drop to $n < r$, and the corresponding sibling node $\tau_k(1,0) \succeq \sigma * n$ will make enumerate all numbers in column p of B between n and r and lift all corresponding coding markers. Notice that the σ-restraint will only drop because something enters A_1 forcing a coding into column p. The upshot is that in *both* cases it cannot be that $\Phi^B_k = A_0$, as either $A_0 \leq_T R$, a computable set, or $A_0 \leq_T A_1$, in the latter case.

The analysis above shows how we can code a Π^0_3 set S into B, but since S is Σ^0_4, we have multiple attempts to code, e.g. A_0 into B, and the above dealt with attempt $(0,0)$. To wit, we will have infinitely many x such that $(0,x)$ potentially witnesses $0 \in S$, provided that it has the Π^0_3 outcome.

The remaining details are to put this together and argue that the combinatorics works. We turn to these details.

The Priority Tree. We will define PT by induction on length of ν on PT according to the following rules.

The tree will consist of τ-nodes, τ_m-nodes, and σ-nodes. τ and τ_m-nodes have associated with them a pair (e,x), namely $e(\tau), x(\tau))$ and in the case of τ_m, this refers to its *mother*, which will be a τ node introduced in the tree and above it. A τ-node has a single outcome o is simply indicates a place that we are introducing an attack on $(e(\tau), x(\tau))$ and hence trying to code $A_{e(\tau)}$ into B. This is done via some *column* $n(\tau)$ chosen from ones set aside for e as indicated at the beginning of the proof. We will not explicitly mention the relevant $\langle e,x,p,q\rangle$ as this seems unnecessary, but will associated coding markers $c(\tau,m,s)$ and m-markers $c(\tau,2m+1,s)$ the latter concerned with whether $e \in S$ is not witnessed by (e,x). τ_m has outcomes $\infty <_L f$. σ-nodes will have outcomes $w <_L 0 <_L 1 <_L 2 <_L 3 <_L \ldots$. Associated with σ is a triple $(e(\sigma), x(\sigma), k(\sigma))$ also indicating which A_e it is attempting to diagonalize and for which x. This information would be summarized by some pair (τ_m, k), with $e(\tau_m) = e(\sigma)$, and $x(\tau_m) = x(\sigma)$.

We then work by length and generate PT. Along some path we will gradually introduce the the nodes above. We will let λ, the empty string, be devoted to $(0,0)$ and thus $\tau(0,0) = \lambda$. This has a single outcome o. We would let this be $\tau_0(0,0)$ and have mother λ. This has two outcomes ∞, f. Below the ∞ outcome we would have a $\sigma(\tau_0,0)$-node and below the f-outcome, we would introduce a new $\tau(1,0)$-node so that it equals of. The σ-type node below the ∞-outcome will have outcomes $w <_L 0 <_L 1 <_L 2 <_L 3 <_L \ldots$. Below each of these we would introduce a new $\tau(1,0)$-node, and promise according to some priority ordering to introduce a new $\tau(0,1)$-node. (This latter case differs from the outcome f of o since we are recognising that we think we have killed $(0,0)$ and need to restart A_0 with a new $x = 1$; whereas below f we are thinking that $x = 0$ is good.)

In general as we work down the tree, gradually introducing nodes accoding to some priority ordering, but remaining sonsistent with the information. If we hit a node ν, we will see 3 cases.

Case 1. ν should be assigned to a new $\tau(e,x)$. Outcome o.

Case 2. ν should be assigned to some $\tau_m(e,x)$. This will have outcomes ∞, f. Below the ∞ outcome we will need to

- Start introducing $\sigma(e,x',k)$ nodes for $x' \leq x$
- Introduce some other $\tau(i,j)$-nodes and $\tau_m(i,j)$-nodes (but not for $i = e$ and $j \leq x$ as these have been dealt with) equitably according to the list, and
- And then introduce a $\tau(e,x+1)$-node.

Below the f-outcome, we would assign the next type of node not yet deal with at ν according to some priority list, and consistent with the information on the path. (So, for example, not a $\tau(e,y)$-node for $y > x$ for instance.)

Case 3. ν should be assigned to some $\sigma(\tau, k)$-node. It will have outcomes $w <_L 0 <_L 1 <_L 2 <_L 3 <_L \ldots$. Below each, we would assign the next type of node not yet deal with at ν according to some priority list, and consistent with the information on the path. Note that nothing will be re-started.

The Construction. We begin at λ. For any n which has entered A_0 at stage $s+1$, put the coding marker for n from column $n(\lambda)$ (Column $n(\lambda)$ equals $\langle 0,0,0\rangle$), into B.

Play outcome o. See if $(0,0)$ has fired. If so play outcome ∞, else play outcome f.

More generally, we have hit ν.

If ν is a $\tau_m(e,x)$-node, see if (e,x) has fired since the last ν-stage. Look at those σ-nodes $\rho \prec \nu$. Each such ρ will have an outcome $\rho * n \preceq \nu$, and an associated use $u(\rho, s)$ of the computation $\Phi^B_{e(\rho)} \upharpoonright n-1 = A_e \upharpoonright n-1$, and possibly $\Phi^B_{e(\rho)} \upharpoonright n \neq A_e \upharpoonright n$, since we preserve to the first disagreement using standard Sacks' strategy. Let u be the maximum of such uses. Accordingly, there will be some $m' \geq \max\{u, m\}$ which we will use to move all coding markers $c(z,s)$ for $z \geq 2m'+1$ which have not moved since the last ν-stage.

If ν is a τ-node, then if this is the first stage we have visited τ since initialization, assign a fresh acceptable column $n(\tau)$ to τ to code $(e(\tau), x(\tau))$. If ν already has a column, then see if any numbers have entered A_e since the last ν-stage and then enumerate the relevant coding markers into column $n(\tau)$.

If ν is a σ-node, then play outcome n for the current σ-correct length of agreement (i.e. consistent with all $\tau_m * \infty \preceq \nu$) up to the first diagreement, with w being played if there are no computations at all.

Generate the TP_s of length s in this way and initialize all nodes right of TP_s at the end of stage s.

This ends the description of the construction.

Verification. We verify the construction. We argue by induction on the length of ν that TP exists and the outcomes are true.

We begin with ν a σ-node. We need to prove that the overall restraint caused by ν is finite along the true path, and that $\nu * n \prec TP$ for some n. Now go to a stage s_0 where we are never left of σ. Let e, x, k be the relevant σ-parameters. Each time after stage s we see a ν-correct computation $\Phi^B_e \upharpoonright n = A_e \upharpoonright n$ (up to the first disagreement) we will play outcome n. The first time we do this, mothers of lower priority than ν below this outcome will have their coding locations set to be above the use of the computation. Furthermore any τ_{m_1}-node below outcome n has to use a proxy m_1' above the ν, n-use for its firings. Therefore this computation will be preserved unless some coding action i entering $A_{e'}$ for some mother $\tau' \preceq \nu$ occurs after stage s. This will cause the length of agreement to drop below n at the next ν-stage, and hence if we ever play n again, mothers below this outcome will again have their coding locations reset. Since no $\tau_m * \infty$-injury can occur for $\tau_m \preceq \nu$ as this is incorporated into ν-correctness, and τ_{m_1} extending any τ_{m_1}-node below outcome n has to use a proxy above the n-use for its firings. We can conclude that injuries only occur because of i entering $A_{e'}$ for some mother $\tau' \preceq \nu$, after s_0. So suppose that $\Phi^B_k = A_e$ at ν. For each n there will be a computation $\Phi^B_k \upharpoonright n = A_e \upharpoonright n$ at a ν-stage s which is A_i-correct for all A_i with mothers above ν. This computation therfore must actually be correct. But then we can conclude

$A_e \leq_T \oplus_{i\in F} A_i$ with F the collection of with mothers τ above ν, and $\tau_m * \infty \prec \nu$. But this means that A_e cannot be in F by the positioning of the σ-type nodes such as ν. This contradicts the independence of $\{A_j \mid j \in \mathbb{N}\}$. Therefore $\Phi_k^B = A_e$ at ν. Hence the $\liminf n$ must exist.

If ν is a τ node, then at some stage ν is never re-initialized, and its coding locations are set. Each time we visit ν we code as required.

If ν is a τ_m node, if it stops firing at some stage after s_0, then $\nu * f$ will be on TP. If m fires infinitely often, we know by induction that the uses associated with σ-type nodes $\sigma * n \preceq \tau_m$ will have finite use, and hence from some point onwards, the proxy m' for m will be fixed. Each time m fires it will move $c(2m'+1, s)$ in column $n(\tau)$ from the mother $\tau \prec \tau_m$, and if τ is below the outcome n for such σ-nodes, then τ will have its chosen so that $m = m'$.

Finally, if $e \in S$, then the way we construct the priority tree means that we will put some $\tau \prec TP$ with $(e(\tau), x(\tau))$ being correct in that x is a witness for $e \in S$. Below τ, every τ_m will have outcome f. Thus we will correctly code A_e into B since each coding marker for $n \in A_e$ can be moved at most a finite number of times by m below n.

If $e \notin S$, then for each pair (e, x) there will be some $\tau_m * \infty \prec TP$ with $(e(\tau), x(\tau)) = (e, x)$. Below each such node we will have $\sigma(e, x, k)$-nodes σ and these will be met as above. This means that $A_e \not\leq_T B$.

This concludes the proof of Theorem 4.6. □

Corollary 4.7. *Not every lowercone of c.e. degrees has a computable representation.*

Proof. Fix any effective independent sequence of low c.e. degrees $A_0, A_1, \cdots$ with uniform lowness indices, that is, $(A_k)' = \Phi^{\emptyset'}_{F(k)}$ for all k and some computable function F. This can be easily done by constructing the sequence directly, satisfying both lowness and diagonalization requirements. Both kinds of requirements are finitary, and the construction produces a uniform sequence of lowness indices F. Now let S be any Σ^0_4-complete set S. By Theorem 4.6 let B be a c.e. set such that $i \in S$ iff $A_i \leq_T B$. If the lowecone below B has a computable representation, say witnessed by the effective sequence $V_0, V_1, \cdots$ of c.e. sets, then we see that $i \in S$ iff $A_i \leq_T B$ iff $A_i \equiv V_k$ for some k. Since A_i is low, the predicate "$A_i \leq_T V_k$" is Σ^0_3 (in the variable k). Since the sequence $\{A_i\}$ is uniformly low, the statement "$A_i \equiv V_k$ for some k" is Σ^0_3. This is a contradiction. □

In an early draft of this paper, from several years ago, we concluded with the following question.

Question 4.8. *Is there some Turing incomplete non-low$_2$ c.e. set A such that the c.e. lowercone below A has a computable representation?*

However, in [7], Downey and Melnikov showed that the answer is *no*.

Theorem 4.9 (Downey and Melnikov [7])**.** *Suppose that A is c.e. and Turing incomplete, then the lowercone below A has a computable representation iff A is low$_2$.*

References

[1] G. Barmpalias and A. Nies, Upper bounds on ideals in the computably enumerable Turing degrees, *Annals of Pure and Applied Logic,* 162 (2011), 465–473.

[2] D. Cenzer, R. Downey, C. Joskusch, and R. Shore, Countable thin Π^0_1 classes, *Annals Pure and Applied Logic*, **59** (1993) 79-139.
[3] D. Cenzer and C. Jockusch, Π^0_1-classes - Structure and applications, in *Contemporary Mathematics* 257 (2000), 39-59.
[4] P. Cholak, R. Downey, N. Greenberg, and D. Turetsky, Realizing computably enumerable degrees in separating classes, accepted in "Higher Recursion Theory and Set Theory", (James Cummings, Andrew Marks, Yue Yang and Liang Yu, eds) volume in celebration of Ted Slamand and Hugh Woodin, Lecture Notes Series Institute for Mathematical Sciences. National University of Singapore, Singapore.
[5] R. Downey and D. Hirschfeldt, *Algorithmic Randomness and Complexity*, Springer-Verlag, 2010.
[6] R. Downey, C. Jockusch and M. Stob, Array nonrecursive sets and multiple permitting arguments, in *Recursion Theory Week, Lecture Notes in Mathematics*, **1432** (1990), 141-174.
[7] R. Downey and A. Melnikov, On realisation of index sets in Π^0_1-classes, *Algebra i Logika* Vol. 58 (2019), 659-663 (Russian).
[8] C. Jockusch and R. Soare, Degrees of members of Π^0_1 classes, *Pacific J. Math.*, **40** (1972), 605–616.
[9] D. Martin, Classes of recursively enumerable sets and degrees of unsolvability. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, **12** (1966), 295–310.
[10] R. I. Soare, *Recursively Enumerable Sets and Degrees*, Springer-Verlag, 1987.
[11] C.E.M. Yates, On the degrees of index sets, *Trans. Amer. Math. Soc.*, 121 (1966), 308-328.
[12] C.E.M. Yates, On the degrees of index sets. II, *Trans. Amer. Math. Soc.*, 135 (1969) 249–266.

Department of Pure Mathematics, University of Waterloo, Waterloo, ON, Canada N2L 3G1
URL: `www.math.uwaterloo.ca/∼csima`
Email address: `csima@uwaterloo.ca`

School of Mathematics, Statistics and Computer Science, Victoria University of Wellington, PO Box 600, Wellington, New Zealand
Email address: `Rod.Downey@ecs.vuw.ac.nz`

School of Physical and Mathematical Sciences, Division of Mathematical Sciences, Nanyang Technological University, Singapore
Email address: `kmng@ntu.edu.sg`