MINIMAL BOUNDS AND MEMBERS OF EFFECTIVELY CLOSE SETS

AHMET ÇEVİK

Abstract. We show that there exists a non-empty special $\Pi^0_1$ class in which no member is a minimal cover for any set, hence prove that degrees of minimal covers cannot be a basis for $\Pi^0_1$ classes.

Study of effectively closed sets, or namely $\Pi^0_1$ classes, has been a long-standing theme in classical recursion theory. Particularly the problem of determining degree theoretic complexity of members of $\Pi^0_1$ classes, going back to Kleene [14], has resulted in a well developed theory. By compactness of the Cantor space, degree theoretic complexity of members of $\Pi^0_1$ classes also determines reals that can be defined by compactness rather than using replacement. Jockush and Soare [11] [12], in their leading papers, showed some very interesting degree theoretic properties of members of $\Pi^0_1$ classes. Many of these results have come to known as basis theorems for $\Pi^0_1$ classes. A typical basis theorem states that every $\Pi^0_1$ class has a member, or a member of degree, of a particular kind. It may be the case that not every $\Pi^0_1$ class has members with the desired property. Similarly, it may be that there is a $\Pi^0_1$ class in which all members satisfy a property. A non-zero Turing degree $a$ is minimal if there is no degree $b$ such that $0 < b < a$. Relativizing the minimal degree construction to any set, we know that every degree has a minimal cover, yet not every degree is a minimal cover. Our motivation arises from the work of Groszek and Slaman [9] and from the problem that which Turing degrees are minimal covers, particularly their relationship with members of $\Pi^0_1$ classes. We consider a question asked by Andy Lewis-Pye in a personal communication, whether or not there exists a non-empty $\Pi^0_1$ class in which no member is a minimal cover, and we give a positive answer, thus showing that degrees of minimal covers cannot be a basis for $\Pi^0_1$ classes.

1. Notation and Terminology

We shall first give our notation and then give some background knowledge on $\Pi^0_1$ classes. We assume some familiarity with basic properties of relative computability and Turing degrees. For a detailed account of computability, the reader may refer

2010 Mathematics Subject Classification. Primary .

The author was supported during this research by the Scientific and Technological Research Council of Turkey (TUBITAK) 1059B191500188. Special thanks to Theodore A. Slaman for many helpful discussions.
to [21, 22, or 7]. Readers who are familiar with computability theoretic notions and \( \Pi^0_1 \) classes may skip to Section 2.

Let \( \omega \) denote the set of natural numbers. We let \( 2^{<\omega} \) denote the set of all finite sequences of 0’s and 1’s. We denote sets of natural numbers by uppercase Latin letters \( A, B, C \).

The subset relation (not necessarily proper) is denoted by \( \subset \). We identify a set \( A \subset \omega \) with its characteristic function \( f : \omega \to \{0, 1\} \) such that, for any \( n \in \omega \), if \( n \in A \) then \( f(n) = 1 \); otherwise \( f(n) = 0 \). We let \( \{\Psi_i\}_{i \in \omega} \) be an effective enumeration of the Turing functionals. Turing functionals will also be denoted by uppercase Greek letters \( \Psi, \Phi, \Theta, \Xi \), etc. We say that \( \Psi_e \) is total if it is defined on every argument, otherwise it is called partial. The join of any given two sets \( A \) and \( B \) is denoted by \( A \oplus B = \{2i : i \in A\} \cup \{2i + 1 : i \in B\} \). \( \Psi_e(A; n) \downarrow \) means that the \( e \)-th Turing functional with oracle \( A \) on argument \( n \) is defined and equal to \( m \). \( \Psi_e(A; n) \uparrow \) denotes it is not the case that \( \Psi_e(A; n) \downarrow \). We will write \( \Psi_e(A; n)[s] \) to mean \( \Psi_e(A; n) \) as defined at stage \( s \). \( A \leq_T B \) means \( B \) computes \( A \) via some Turing functional \( \Psi_e \). So \( A \leq_T B \) iff there exists some \( e \in \omega \) such that \( A = \Psi_e(B) \). \( A \leq_T B \) means \( A \leq_T B \) and \( B \leq_T A \). If \( A \leq_T B \) and \( B \leq_T A \), then \( A \) and \( B \) are Turing equivalent, which is denoted by \( A \equiv_T B \). The Turing degree of a set \( A \subset \omega \) is the set \( \{X : X \equiv_T A\} \). We denote degrees with boldcase letters \( a, b, c \).

We denote finite strings in \( 2^{<\omega} \) by lowercase Greek letters \( \sigma, \tau, \eta, \rho, \pi, \upsilon \). They may also be denoted by the same letters with accents and subscripts such as \( \sigma', \sigma'', \sigma^-, \sigma_0, \sigma_1 \), etc. We let \( \sigma \ast \tau \) denote the concatenation of \( \sigma \) followed by \( \tau \). For two strings \( \sigma \) and \( \tau \), we let \( \sigma \subset \tau \) denote that \( \sigma \) is an initial segment of \( \tau \) as a substring. We say a string \( \sigma \) is compatible with \( \tau \) if either \( \sigma \subset \tau \) or \( \tau \subset \sigma \). Otherwise we say that \( \sigma \) and \( \tau \) are incompatible. Similarly, we say that \( \sigma \) extends \( \tau \) if \( \tau \subset \sigma \), i.e., \( \sigma \) is an extension of \( \tau \). Let \( |\sigma| \) denote the length of \( \sigma \). The unique string of length 0 is called the empty string and it is denoted by \( \lambda \). We let \( \sigma(i) \) denote the \( (i+1) \)-st bit of \( \sigma \).

For any \( \sigma \in 2^{<\omega} \) and \( n \in \omega \), conventionally we let \( \Psi_e(\sigma; n) \) be defined and equal to \( \Psi_e(A; n) \) if \( \sigma(i) = A(i) \) for all \( i < |\sigma| \) and if computing \( \Psi_e(A; n) \) requires only values \( A(i) \) for \( i < |\sigma| \). Let \( A \vdash z \) and \( \sigma \vdash z \) denote, respectively, the restriction of \( A(x) \) or \( \sigma(x) \) to those \( x \leq z \). For a set \( A \subset \omega \), we define the jump of \( A \), denoted by \( A' \), to be the set \( \{e : \Psi_e(A; e) \downarrow\} \). We write the \( n \)-th jump of a degree \( a \) as \( a^{(n)} \).

A set \( T \) of strings is downward closed if \( \sigma \in T \) and \( \tau \subset \sigma \) implies \( \tau \in T \). Occasionally we refer to downward closed sets of strings as trees. We say that a set \( A \) lies on a tree \( T \) if there exist infinitely many \( \sigma \) in \( T \) such that \( \sigma \subset A \). A set \( A \) is a path on \( T \) if \( A \) lies on \( T \). We denote the set of infinite paths of \( T \) by \( [T] \). We say that a string \( \sigma \in T \) is infinitely extendible if there exists some \( A \supseteq \sigma \) such that \( A \in [T] \). If \( \sigma, \tau \in T \) and \( \sigma \subset \tau \) and there does not exist \( \sigma' \) with \( \sigma \subset \sigma' \subset \tau \) then we say that \( \tau \) is an immediate successor of \( \sigma \) in \( T \) and \( \sigma \) is the immediate predecessor of \( \tau \) in \( T \).

We say that \( \mathcal{P} \subset 2^\omega \) is a \( \Pi^0_1 \) class if there exists a downward closed computable set of strings \( T \) such that \( \mathcal{P} = [T] \). We can then have an effective enumeration \( \{\Lambda_i\}_{i \in \omega} \) of downward closed computable sets of strings such that for any \( \Pi^0_1 \) class \( \mathcal{P} \) there exists some \( i \in \omega \) such that \( \mathcal{P} \) is the set of all infinite paths through \( \Lambda_i \).

1.1. **Background on \( \Pi^0_1 \) classes.** In this section, we will overview some works in the literature to give an overall idea of how our claim is related to other works. However, for a detailed survey on \( \Pi^0_1 \) classes we refer the reader to [1] and [6]. A
\(\Pi_0^1\) class is an effectively closed subset of \(2^\omega\) Cantor space. One important property of \(\Pi_0^1\) classes is that for any recursively axiomatizable theory (the deductive closure of a recursively enumerable set of sentences in a language), the set of complete consistent extensions can be seen as a \(\Pi_0^1\) class [19]. The opposite direction is also proved in [8]. That is, any \(\Pi_0^1\) class can be seen as the set of complete consistent extensions of an axiomatizable theory. The compactness property of the Cantor space is provided by Weak König’s Lemma which tells us that if \(T\) is an infinite downward closed set of finite strings, then there exists an infinite path through \(T\).

Countable \(\Pi_0^1\) classes are another type of effectively closed sets. It is important to note that all countable \(\Pi_0^1\) classes contain an isolated point and that every isolated point is computable [15]. So if a \(\Pi_0^1\) class contains no computable member then it must be uncountable and so \(T\) must be perfect, i.e., every \(\sigma \in T\) has at least two incompatible extensions in \(T\).

Effectively closed sets in which no two members are of the same degree are called \(\Pi_0^1\) choice classes, and the properties of these classes were studied in [4]. Any countably infinite \(\Pi_0^1\) class has members of the same degree, hence \(\Pi_0^1\) choice classes cannot be countably infinite. They also cannot be finite unless it is a singleton since all members of a finite \(\Pi_0^1\) must be computable. Another property of \(\Pi_0^1\) choice classes is that they do not contain any 1-random set.

We are particularly interested in complexity of members of \(\Pi_0^1\) classes in the Turing degree universe. Some of the most important and frequently used results are basis theorems: a basis theorem tells us that every non-empty \(\Pi_0^1\) class has a member of a particular kind. Anything which is not a basis is called non-basis. The most celebrated Low Basis Theorem of Jockusch and Soare [11] states that every non-empty \(\Pi_0^1\) class contains a member of low degree, i.e., a degree \(a\) such that \(a' = 0'\). Same authors proved that any non-empty \(\Pi_0^1\) class contains a member of hyperimmune-free degree, i.e., a degree \(a\) such that for any \(A \in a\) and for any function \(f \leq_T A\), there exists a computable function \(g\) such that \(g(n) \geq f(n)\) for all \(n\). These results were proved by forcing with \(\Pi_0^1\) classes in which we successively move from a set to one of its subsets in order to force satisfaction of a given requirement. Another important basis theorem for \(\Pi_0^1\) classes is that every non-empty \(\Pi_0^1\) class has a member of recursively enumerable (r.e.) degree, in particular, the leftmost path of any downward closed computable set of strings is of r.e. degree. One interesting result by Jockusch and Soare [11] is that every \(\Pi_0^1\) class which does not contain a recursive member contains members of degrees \(a\) and \(b\) such that \(a \land b = 0\). However, this does not hold for the cupping case. In fact, it was shown in [5] that there exists a non-empty \(\Pi_0^1\) class \(T\) with no recursive member such that \(0' \nleq_T A \oplus B\) for any \(A, B \in \mathcal{P}\). Another non-basis result, given in [12], is that the class of r.e. degrees strictly below \(0'\) does not form a basis. Similarly, the class of recursive sets does not form a basis either since there exists a \(\Pi_0^1\) class such that all members are non-recursive. We call \(\Pi_0^1\) classes with no recursive member special \(\Pi_0^1\) classes. In [6], it was proven that every non-empty special \(\Pi_0^1\) class contains a member of properly low\(n\) degree, i.e., a degree \(a\) such that \(a^{(n)} = 0^{(n)}\) but \(a^{(n-1)} \neq 0^{(n-1)}\). We say that a degree \(a\) satisfies the join property if for all non-zero \(b < a\) there exists \(c < a\) such that \(b \lor c = a\). In [5], it was shown that there exists a non-empty special \(\Pi_0^1\) class in which no member satisfies the join property.
An antibasis theorem tells us that a \( \Pi^0_1 \) class cannot have all/any members of a particular kind without having a member of every Turing degree. Kent and Lewis \cite{13} proved the Low Antibasis Theorem which says that if a \( \Pi^0_1 \) class contains a member of every low degree, then it contains a member of every degree. In \cite{3}, the latter result was extended in relation to the jump hierarchy, that for a given degree \( a \geq 0' \), if a \( \Pi^0_1 \) class \( P \) contains members of every degree \( b \) such that \( b' = a \), then \( P \) contains members of every degree. A local version of this result is also given in the same work. That is, when \( a \) is also \( \Sigma^0_2 \), it suffices in the hypothesis to have a member of every \( \Delta^0_2 \) degree \( b \) such that \( b' = a \).

2. OVERALL IDEA OF THE PROOF

The question of whether there exists a non-empty special \( \Pi^0_1 \) class in which no member is a minimal cover was asked by Andy Lewis-Pye in a personal communication around 2013. Our aim is to show that degrees of minimal covers cannot be a basis for \( \Pi^0_1 \) classes. There are three closely related results in the literature. First was provided by Lewis \cite{16}, where he showed that there exists a non-empty special \( \Pi^0_1 \) class every member of which is of degree with strong minimal cover. Second related result was given by Sasso \cite{20}, where he modified Sacks’ minimal degree construction below \( 0' \) to obtain, recursively in \( 0' \), a tree the limit of every branch of which is recursive or of minimal degree. Of course Sasso didn’t really construct a \( \Pi^0_1 \) class, but it can be thought of as a limit computable version of an effectively closed set in which all members satisfy the minimal degree requirements. The third relevant work is a paper by Groszek and Slaman \cite{9}, where they show the existence of a non-empty special \( \Pi^0_1 \) class in which all members compute a set which is of minimal degree. In fact, their theorem uses some level of non-uniformity, for that they construct a \( \Pi^0_1 \) class in which every member is either of minimal degree or computes a set of minimal degree. The non-uniformity here is used to satisfy the second literal of the disjunction. They define a co-r.e. tree \( M \), with no terminal nodes, such that the \( n \)-th minimality requirement

\[
\Psi_n(A) \leq_T A \Rightarrow \Psi_n(A) \text{ is recursive or } A \leq_T \Psi_n(A)
\]

is satisfied for all \( A \in [M] \) except for countably many members \( B \in [M] \). They start with Sacks’ \cite{17} minimal degree construction method below \( 0' \), that is, by enumerating a \( \Psi_n \)-splitting tree \( T_n \) in \( M \) and guaranteeing that all branches through \( M \) are either branches through \( T_n \), in which case we have \( A \leq_T \Psi_n(A) \), or they extend a terminal node of \( T_n \). Now for branches \( B \) extending a terminal node of \( T_n \), they guarantee that \( B \) computes a non-recursive r.e. set via some reduction that the authors construct. Hence by Yates’ theorem \cite{22}, that every non-recursive r.e. set computes a set of minimal degree, \( B \) computes a set which is of minimal degree. Of course, by the hyperimmune-free basis theorem such a \( \Pi^0_1 \) class must contain a member of minimal degree since hyperimmune-free sets cannot compute any non-zero r.e. set. Thus, the \( \Pi^0_1 \) class constructed by Groszek and Slaman must necessarily contain a member of minimal degree.

For our problem, the most naive approach in constructing a non-empty special \( \Pi^0_1 \) class with the property that no member is a minimal cover would be to make sure that all members are non-recursive and of r.e. degree. But this cannot work since every non-empty \( \Pi^0_1 \) class contains a member of hyperimmune-free degree. Of course if we drop the non-recursiveness condition, then obviously a \( \Pi^0_1 \) class with all
members recursive gives an effectively closed set in which no member is a minimal cover. But this does not give solution to our problem. The approach we will consider to solve our problem will be similar to the Groszek-Slaman construction, but will contain some different features. So first we shall discuss to what extent they are related and how they differ from each other so as to give a general idea of our strategy. Like Groszek and Slaman, we will find it more convenient to build a $\Pi^0_1$ class from its complement, i.e., we define a co-r.e. tree $T$ with no terminal nodes such that no member of $[T]$ is a minimal cover. Groszek and Slaman use the level $n$ construction for satisfying the $n$-th minimality requirement as the main procedure of their overall construction with a subroutine what they call the $\psi_n$-subconstruction procedure, where they take care of the paths $B$ for which they fail to satisfy the condition $B \leq_T \psi_n(B)$ for countably many sets $B$ in the $\Pi^0_1$ class, but instead they guarantee that $B$ computes a non-recursive r.e. set via a Turing reduction that they also construct (unless $\psi_n(B)$ is partial or recursive, thus $B$ is subject to a successful level $n + 1$ construction). In other words, when enumerating a $\psi_n$-splitting tree, as long as the outcome is $\Pi_2$, i.e., in the $\psi_n$-splitting region of their tree at stage $s$, the $n$-th minimality requirement is satisfied by ensuring $B \leq_T \psi_n(B)$. In the $\Sigma_2$ outcome, where no usable $\psi_n$-splitting are found until stage $s$, they ensure that either $B$ computes a non-recursive r.e. set, or $\psi_n(B)$ is partial or recursive and $B$ is subject to a successful level $n + 1$ minimality construction.

We will define $\psi_n$-splittings inside our tree at each stage, just like in Groszek and Slaman’s construction, and our approach for the $\Pi_2$ outcome of the minimality requirement (in the $\psi_n$-splitting region of the tree) is the same. If $A$ is a branch on a $\psi_n$-splitting tree $T$, then $A \leq_T \psi_n(A)$. The proof of this is quite standard. Given $\psi_n(A)$ we can generate increasingly long segments of $A$ recursively in $\psi_n(A)$. Given $\sigma \subset A$, assuming that $A$ lies on the tree $T$, either $\sigma \ast 0$ or $\sigma \ast 1$ must be an initial segment of $A$, and we have to decide which one of them is. Since $T$ is $\psi_n$-splitting, there exists some $x \in \omega$ such that $\psi_n(\sigma \ast 0; x) \downarrow \neq \psi_n(\sigma \ast 1; x) \downarrow$. But then only one of them can be compatible with $\psi_n(A; x)$. We just take the one which is compatible, and this determines which of the two strings is an initial segment of $A$. So if $A \leq_T \psi_n(A)$, the theorem is satisfied rather trivially and we will win by that particular $\psi_n$. As for the $\Sigma_2$ outcome, though, our strategy will be different since our problem is not to ensure that the branches compute a set of minimal degree, but to ensure that if $A \in [T]$, where $T$, say, is the co-r.e. tree to be constructed, then there exists a reduction $\Phi$ satisfying that $\psi(A) <_T \Phi(A) <_T A$. In other words, we carry out a density argument inside the region of $T$ where and as long as we fail to find any $\psi_n$-splittings. This is where most of work in the paper will be given. The $n$-th minimality requirement, taken together with the non-recursiveness requirement that all branches through the tree are non-recursive, will roughly correspond to Groszek and Slaman’s level $n$ construction. We will not work with the $n$-th minimality requirement aiming to satisfy it on all branches of $T$, as that would be an attempt to produce a $\Pi^0_1$ class all of whose members are of minimal degree, which is impossible anyway since every non-empty $\Pi^0_1$ class contains a member of r.e. degree (and r.e. degrees cannot be minimal due to Sacks’ density theorem [18]). Instead, if we can’t seem to get $A \leq_T \psi_n(A)$ for satisfying the theorem trivially, we will succeed in other ways, such as making relevant functionals in the requirements (listed below) partial or we will make $A$ of r.e. degree (see Figure[1] for a sketch of possible outcomes).
Along the construction, we define $\Psi_n$-splittings inside the tree $T$ at each stage, and the $\Psi_n$-splitting region will grow over the dummy region where we have not yet found any $\Psi_n$-splittings for computations up to that stage. The density strategies (will be denoted by $S_1$ and $S_2$) will be divided into $\Pi_2$ and $\Sigma_2$ cases, depending if the node equipped with the strategy is in the $\Psi_n$-splitting region or otherwise of $T$ of that particular stage. In the $\Psi_n$-splitting region, we will have no problem in satisfying the theorem in a trivial way. The density requirements will hold trivially in the $\Pi_2$ case since we get $A \leq_T \Psi_n(A)$ for every $A$ that lies on the $\Psi_n$-splitting subtree of $T$. If, however, we observe that there are no $\Psi_n$-splittings by stage $s$, that is where we need the density requirements. In that case we start to implement a density construction for this $\Sigma_2$ region until we get $\Psi_n$-splittings, and we have to ensure that if $A$ is a branch that leaves a terminal node of the $\Psi_n$-splitting tree, then we guarantee that $\Phi(A)$ is total if $\Psi_n(A)$ is total. This is to ensure that $\Psi_n(A) \leq_T \Phi(A)$. We will observe that if it were the case that $\Theta(\Psi(A)) = \Phi(A)$, then either $\Theta$ is partial (and so we win by that particular $\Theta$) or we can get an anti-chain of $\Phi$-splits on a unique infinite path $A$ where $\Phi$ is partial everywhere but on $A$. In that case we will show $A$ is of r.e. degree. Similarly, as for the other density requirement, if it were the case that $\Xi(\Phi(A)) = A$ for some Turing functional $\Xi$, then we make sure that either $\Xi$ is partial (and so we win by that particular $\Xi$) or $A$ is of r.e. degree.

**Definition 2.1.** Given any stage $s$, let $\sigma \in T_s$, where $T_s$ denotes the set of strings that are not enumerated out of $T$ before stage $s$. For a given $n \in \omega$, we say that $\sigma$ has an active $\Psi_n$-split at stage $s$ if there exist two incompatible strings $\sigma_1, \sigma_2 \supset \sigma$ in $T_s$ such that $|\sigma_1| = |\sigma_2| = s$ and $\Psi_n(\sigma_1) \neq \Psi_n(\sigma_2)$.

The requirements will be as follows. For any $A \in [T]$,

- $S_0(n)$: Either $\Psi_n(A) \geq_T A$ or $\exists \sigma \subset A \exists s \in \omega$ such that $\sigma$ has no active $\Psi_n$-split extension after stage $s$.
- $S_1(n)$: $\Theta(\Psi_n(A)) = \Phi(A) \Rightarrow A$ is of r.e. degree.
- $S_2(n)$: $\Xi_n(\Phi(A)) = A \Rightarrow A$ is of r.e. degree.
- $D(n)$: $A \neq \Psi_n(\emptyset)$.

The non-recursiveness requirements $D$ will be handled inside the $S_1$ and $S_2$ strategies separately. The strategy to satisfy the requirement denoted by $S_0$, when taken together with the strategy to satisfy the $D$ requirements, will act as our “level $n$ construction” akin to the one in Groszek-Slaman construction. The density requirements, where we act so long as we don’t find any active $\Psi_n$-splittings, are denoted by $S_1$ and $S_2$, and will have wins by $\Pi_2$ or $\Sigma_2$ outcomes. The density requirements make sure there is a set inbetween $\Psi_n(A)$ and $A$ for $n \in \omega$. We sketch the possible outcomes in Figure 4 and we will go through it in the next section. The outcomes for $S_1$ (not shown in Figure 4) will be similar, except that instead of $\Xi$, we will have $\Theta$ partial. How we satisfy the $S_1$ and $S_2$ requirements will become clear once we define the strategies.

In the next section we give the proof of our main result based on the idea we discussed so far. It should be noted that the construction mechanics is complicated enough that it may not be possible to work out with a single level of procedure.
3. Proof of the main theorem

We shall now give the main result of this paper.

**Definition 3.1.** A set $A$ is a *minimal cover* for a set $B <_T A$ if there is no $C$ such that $B <_T C <_T A$. A degree $a$ is a *minimal cover* for a degree $b < a$ if there is no $c$ such that $b < c < a$.

**Theorem 3.2.** There exists a non-empty special $\Pi^0_1$ class in which no member is a minimal cover for any set.

The proof is a type of density argument inside a tree. We construct a non-empty special $\Pi^0_1$ class $\mathcal{P}$ such that for any $\Psi(A)$ of any given $A \in \mathcal{P}$, if $\Psi(A) <_T A$, we define $\Phi(A)$ with $\Phi(A) \leq_T A$ and $\Psi(A) \leq_T \Phi(A)$, and we satisfy the requirements given above in Section 2.

Our approach in satisfying $S_0$ (together with $D$ requirements) is similar to the method discussed in Groszek and Slaman. One difference however is the application of the density requirements used in the splitting trees and that the $D$ requirements are handled separately by $S_1$ and $S_2$ strategies.

We shall represent the $\Pi^0_1$ class as the collection of branches through a co-r.e. tree $T$ with no terminal strings. For this we describe a recursive construction that will determine $T$ by specifying at each stage $s$, which strings of the full binary tree will be enumerated out of $T$ before stage $s$. We denote by $T_s$ the set of strings that are not enumerated out of $T$ before stage $s$ and we let $T = \bigcap_s T_s$. We ensure that each $T_s$ is an infinite perfect tree, i.e., every string in $T_s$ has at least two incompatible extensions. In the end, $T$ will be an infinite tree with no terminal strings such that $[T] \neq \emptyset$. 

![Figure 1. Outcomes and wins.](image-url)
Notation. For two strings $\sigma$ and $\tau$ in $T$ such that $\sigma \subset \tau$, let us write $\sigma \sim \tau$ to denote that there is no string $\eta \in T$ such that $\sigma \subset \eta$ and that $\eta$ and $\tau$ are incompatible. For convenience, when we enumerate a string $\tau$ out of $T$, we also enumerate out any $\sigma \in T$ such that either $\tau \subset \sigma$ or $\sigma \subset \tau$ and $\sigma \sim \tau$. This preserves that $T_s$ has no terminal strings. Enumerating out of $T$ finitely many strings (and their extensions) and guaranteeing to leave at least one string will ensure all together that each $T_s$ is an infinite perfect tree.

3.1. Control mechanics. We give some more conventions regarding the construction. When working with trees, the terms ‘string’ and ‘node’ will often be used interchangably to denote the elements of the tree. The overall recursive construction will contain the instructions of how to define the $\Pi^0_1$ class $\mathcal{P} = [T]$ for a co-r.e. tree $T$. But the main procedure used in the overall construction is the level $n$ construction, i.e., the $S_0(n)$-strategy. The level $n$ construction inside $\Psi_n$-splitting tree above the node $\tau$ describes a part of the overall recursive construction which produces above $\tau$ a sequence of trees $\{T_s\}_{s \in \omega}$. The $\Psi_n$-splitting tree will be a splitting tree on $T$. The level $n$ construction is what corresponds to the strategy for $S_0$ (when taken together with the strategy for non-recursiveness $D$ requirements).

The density strategies will act as a subprocedure in the level $n$ construction so long as we don’t find active $\Psi_n$-splits. For the level $n$ construction, our notation for handling the construction mechanics will be slightly different from that of the Groszek-Slaman level $n$ construction in the sense we will explain below.

During the construction we will place on nodes various strategies for the requirements that we listed earlier. A node may be equipped with more than one strategy. A fixed priority ordering can be defined on the equipped strategies to execute them in some particular order. At any stage $s$, each node which is equipped with a strategy acts when the hypothesis of the associated requirement holds up to stage $s$.

We also use what may be called flagging along the construction in the following sense. At each stage $s$, every node will have a binary flag value, namely $\Pi$ and $\Sigma$, associated with the $n$-th Turing functional $\Psi_n$ for each $n$ such that $n \leq s$. Since the outcomes of the $S_0$ requirement can be divided into $\Sigma_2$ and $\Pi_2$ cases, the flag value of the node denotes whether the node is in the splitting region or otherwise of $T_s$ with respect to $\Psi_n$ at stage $s$. We say that a node $\sigma$ is in the $\Pi$-region in $T_s$ with respect to $\Psi_n$ if for any $\sigma_1, \sigma_2 \supset \sigma$ of length $\leq |s|$ in $T_s$ we have that $\Psi_n(\sigma_1; x)[s] \neq \Psi_n(\sigma_2; x)[s]$ for some $x \in \omega$. If there is no active $\Psi_n$-splittings above $\sigma$, then $\sigma$ is in the $\Sigma$-region of $T_s$. Of course a node may be in different regions with respect to different indices of Turing functionals. For example if every incompatible extensions of $\sigma$ are $\Psi_n$-splitting but there are no active $\Psi_m$-splittings above $\sigma$, then $\sigma$ is in the $\Pi$-region with respect to $\Psi_n$, but in the $\Sigma$-region with respect to $\Psi_m$.

Depending on which region $\sigma$ is in, with respect to an index $i$, the relevant versions of the $S_1$ and $S_2$ strategies will be executed accordingly. The $\Pi$ and $\Sigma$ versions of the $S_i$-strategy, for $i \in \{1, 2\}$, are denoted respectively as $S_{i, \Pi}(n)$ and $S_{i, \Sigma}(n)$. In the given scenario, then, $S_{1, \Pi}(n)$ and $S_{2, \Pi}(n)$ will be executed above $\sigma$ in $T_s$ along with $S_{1, \Sigma}(m)$ and $S_{2, \Sigma}(m)$.

The control mechanics of the strategy for $S_2$ may need a bit more care. We will use a subroutine in the $S_2$ strategy which enumerates and adjusts possible $\Phi$ values in such a way to satisfy some condition related to $\Psi$ values that we will explain shortly.
In constructions alike, it is customary to describe which requirements, in particular \( S_2 \) requirements in this case, are ‘active at a node \( \sigma' \) at any given stage of the construction and which ones deserve to be called ‘complete’ in order to move on to the next requirement.

Suppose that \( \sigma \) is a node and \( i \in \omega \) is a given index. Let \( \sigma_0, \sigma_1 \) be two incompatible successors of \( \sigma \) on which some strategies are placed. We describe the control mechanics taking \( S_2 \) as an example, but it can be similarly applied to the others. We say that an \( S_2(i) \)-strategy is active at \( \sigma \) if \( \sigma \) is not yet enumerated out of \( T \) by stage \( s \) and \( \Xi(\Phi(A))[s] = \tau \subset A \) for some functional \( \Xi \). When the \( S_2(i) \)-strategy is active at \( \sigma \), given \( \Xi_i \), the \( S_2(i) \)-strategy placed on \( \sigma \) will search for each \( k \in \{0, 1\} \), an extension \( \sigma_k \supset \sigma_k \) of length \( \leq |s| \) in \( T_s \) such that \( \Xi_i(\Phi(\sigma_k)) \supset \sigma_k \).

When such \( \sigma_k^* \) is found, we say that \( \sigma \) is complete (for the associated strategy) for all pairs \((i, \sigma')\) such that \( \sigma' \supset \sigma_k^* \). A node with \( S_2(i) \)-strategy being complete means that the hypothesis of the \( S_2(i) \) requirement is now satisfied for \( \Psi_i \) and extensions of \( \sigma_k \) at stage \( s \). Once a node is complete for a strategy, we can pass the control to the one with next lower priority.

The level of a string \( \sigma \) in \( T_s \) will refer to the number of proper initial segments of \( \sigma \) in \( T_s \). We decide whether or not \( i \in \omega \) requires the attention of \( S_2 \) at node \( \sigma \) as follows. Let us say that \( i \) requires \( S_2 \)-attention at \( \sigma \) unless there exists some proper initial segment \( \rho \subset \sigma \) with an \( S_2(i) \)-strategy such that \( S_2(i) \) is active at \( \rho \) but \( \rho \) is not complete for all \((i, \tau)\) such that \( \tau \supset \rho \). Let us also denote the set of \( S_2 \) requirements active at a node \( \sigma \) by \( \beta_\sigma \). This will be determined by their indices. For example if \( \beta_\sigma = \{m\} \), then only \( S_2(m) \) is active at \( \sigma \).

If \( \sigma = \lambda \), then \( \beta_\sigma = \{0\} \).

Suppose that \( \sigma \neq \lambda \) and \( \sigma \) is a string of level \( n > 0 \). Let \( \sigma' \) be the initial segment of \( \sigma \) equipped with an \( S_2 \)-strategy placed and which is of level \( n - 1 \). If \( \sigma' \) is \( S_2 \)-complete for all \((i, \sigma)\) such that \( i \in \beta_{\sigma'} \), then

\[
\beta_\sigma = \beta_{\sigma'} \cup \{i'\},
\]

where \( i' \) is the least such number not in \( \beta_{\sigma'} \) which requires \( S_2 \)-attention at \( \sigma \). Otherwise, \( \beta_\sigma \) is the set of all \( i \in \beta_{\sigma'} \) which require \( S_2 \)-attention at \( \sigma \).

Strategy for \( S_1 \) is executed over a pair of indices \((i, j)\), for arbitrary functionals \( \Theta \) and \( \Psi \). However, without loss of generality, we may assume for simplicity working with a single index \( n = \langle i, j \rangle \) for either of the arbitrary functionals appearing in \( S_1 \) requirements. Whenever an \( S_1(n) \)-strategy is placed on a string \( \sigma \) such that \( \sigma_0 \) and \( \sigma_1 \) are two incompatible extensions of \( \sigma \), it will search for strings \( \sigma_0^* \supset \sigma_0 \) and \( \sigma_1^* \supset \sigma_1 \) satisfying either

\[
\sigma_0 = \Phi(\sigma^*_0) \subset \Theta(\Psi_n(\sigma^*_0)) \quad \text{or} \quad \sigma_1 = \Phi(\sigma^*_1) \subset \Theta(\Psi_n(\sigma^*_1)).
\]

For \( D \) requirements, given \( n \in \omega \), the \( D(n) \)-strategy will be executed to satisfy the non-recursiveness requirement above \( \sigma \). That is, it will ensure that for any \( A \supset \sigma \) in \([T], A \neq \Psi_n(\emptyset) \).

There will be finitely many requirements active on a given node bounded by stage \( s \). At each stage, the strategy placed on a node performs the instructions for
all of these in order of some fixed priority. We consider the $S_0(n)$-strategy acting as the main procedure of the overall construction of our $\Pi_0^0$ class since $S_1$ and $S_2$ strategies will have two versions, each depending on the outcome of $S_0$ at each stage. Similarly, the $D(n)$-strategy will be performed in accordance with $S_1$ and $S_2$.

We also define a set $\Phi^+$ of strings for maintaining possible $\Phi$ axioms during the construction. We define axioms of the form $\Phi(\sigma) \supset \tau$. Set of possible axioms of $\Phi$ that are enumerated in $\Phi^+$ may not be necessarily consistent with each other when they are considered as axioms, but we will be able enumerate in the construction proper axioms of $\Phi$ from $\Phi^+$. The important point when enumerating axioms for $\Phi$ is that we will aim to make sure that $\Phi(A)$ is total if $\Psi(A)$ is total for any $A \in [T]$.

Now we give the strategies. First we describe how the strategy for $S_0$ works above a given string $\sigma$.

3.2. $S_0(n)$-strategy above $\sigma$. Given stage $s$ and index $n \in \omega$, we say that a string $\sigma \in T$ is a $\Pi_{s,n}$-boundary point if $\sigma$ is in the $\Pi$-region of $T_s$ but every $\tau \supset \sigma$ is in the $\Sigma$-region of $T_s$ with respect to $\Psi_n$. Intuitively, $\Pi_{s,n}$-boundary points determine the longest strings in $T$, as defined at stage $s$, above which there are no active $\Psi_n$-splittings (see Figure 2 for depiction).

The role of $S_0$ will be to update the $\Pi$-region of $T$ at each stage. The strategy will define the $\Pi_{s,n}$-boundary points at each stage $s$ which will determine the layers containing $\Psi_n$-splitting strings and strings above which there is no active $\Psi_n$-split. When we say $\sigma$ is a $\Pi_{s,n}$-boundary point, we mean this so relative to the given index $n$ in the strategy and the stage $s$ we are in. So occasionally we will drop the indices and just write $\Pi$-boundary point as it should be understood relativized to that stage and functional index. Same thing when we say a string is in the "$\Pi$-region", where it should be understood that, in fact, we mean this so relative to a functional index $n$.

Let $\sigma$ be a $\Pi_{s,n}$-boundary point on which $S_0(n)$-strategy is placed. Intuitively, at stage $s$, in $S_0(n)$-strategy we define $T_s \subseteq 2^{<\omega}$ such that either $\Psi_n(\sigma_1)[s] \neq \Psi_n(\sigma_2)[s]$ for every incompatible extensions $\sigma_1 \supset \sigma$ and $\sigma_2 \supset \sigma$ of length $s$ in $T_s$ or that there exists some $\tau \supset \sigma$ above which there are no active $\Psi_n$-splittings of length $s$.

Instructions of step $s$ inside $T_s$ above $\sigma$ are as follows:

At stage 0: Define $\tau \supset \sigma$ as the least string in $T$ such that $\tau \neq 0$ and $\tau \neq 1$ are not yet enumerated out.

At stage $s > 0$, suppose that $\sigma$ is a $\Pi_{s,n}$ boundary point on which an $S_0(n)$-strategy is placed, where $n \leq s$. We see if there exist strings $\sigma_1, \sigma_2 \supset \sigma$ of length $s$ which are extendible in $T_s$ such that $\Psi_n(\sigma_1)[s] \neq \Psi_n(\sigma_2)[s]$. If not, we do nothing with the $\Pi$-boundary points and just place $S_1(n)$ and $S_2(n)$-strategies on $\sigma$ if not placed yet. Otherwise, when we find $\sigma_1$ and $\sigma_2$, we enumerate out all $\tau \in T$, for $\tau \supset \sigma$, such that $\tau$ is incompatible with $\sigma_1$ and $\sigma_2$. Take two least extensions $\tau_{1,i}$ and $\tau_{2,i}$ of $\sigma_i$, for each $i \in \{1, 2\}$, and declare them as the new $\Pi$-boundary points. Then place $S_1(n)$ and $S_2(n)$-strategies on two incompatible extensions of $\tau_{1,i}$ and $\tau_{2,i}$.

Since the $\Pi$-boundary points of the previous stage are extended by the ones of the current stage, this ensures that the $\Pi$-region expands over the $\Sigma$-region. We also
ensure that \( S_0(n) \) does not terminate any nodes in the \( \Pi \)-region or the permanent \( \Sigma \)-region. This can be shown by induction on \( s \) that for every \( n \) and \( \sigma \), step \( s \) of level \( n \) construction inside \( T_s \) above \( \sigma \) does not enumerate \( \sigma \) out of \( T \). At step \( s + 1 \) of the construction, we execute step \( t \) for some \( t \leq s \) of level \( n + 1 \) construction inside \( T_t \) above \( \tau \supset \sigma \), and we apply the inductive hypothesis.

By the end of stage \( s \), \( S_0(n) \) will have two outcomes in \( T_s \). Either we find \( \Psi_n \)-splittings up to length \( s \) above the given string \( \sigma \) in the hypothesis, or there is no active \( \Psi_n \)-splittings. Let us denote these outcomes, respectively, as \( \Pi_2 \) and \( \Sigma_2 \).

\( S_1 \) and \( S_2 \) will act accordingly for each outcome of \( S_0 \). Then, the strategy \( S_1 \) has two versions, namely \( S_{1,\Pi} \) and \( S_{1,\Sigma} \). Similarly for \( S_2 \). Suppose that \( \sigma \) is a node with an \( S_0 \)-strategy and consider the \( \Pi_2 \) outcome of \( S_0 \). In that case, we carry out the construction as if \( \sigma \) has an active \( \Psi_i \)-split. So all low priority \( S_1 \) and \( S_2 \) requirements will perform their \( S_j,\Pi \) and \( S_j,\Sigma \) strategies, for \( j \in \{1,2\} \), meaning that \( S_1 \) and \( S_2 \) will act as if there are active \( \Psi_i \)-splits but no active \( \Psi_{i+1} \)-splits above the string on which the strategy is placed.

\( S_1 \) and \( S_2 \) strategies will therefore either end up with a \( \Pi_2 \) or \( \Sigma_2 \) outcome depending on the flag value of the equipped node. In each case we have a different win. In Figure 2 we sketch these possible outcomes and wins.

Note that the \( S_0 \)-strategy we gave above is for a fixed Turing functional \( \Psi_i \). For other functionals, without loss of generality we may use, just as in the Sacks construction, \( \Psi_j \)-splittings inside a \( \Psi_i \)-splitting tree for \( j > i \). In the overall construction, as we will give in the end, the \( S_0(i+1) \)-strategy will enumerate \( \Psi_{i+1} \)-splittings inside the \( \Psi_i \)-splitting subtree of \( T \).

Let us now analyze Figure 1. Outcome 1 is achieved when we suppose that we are in the \( \Psi_n \)-splitting region. In this case we have \( \Psi_n(A) \geq_T A \) for \( A \) on the
Let \( \sigma \) be a node on which the next highest priority \( D(n) \)-strategy is placed. Let \( \sigma_1 \) and \( \sigma_2 \) be two successors of \( \sigma_0 \). We wait until stage \( s' \) such that \( \Psi_n \) is defined on all strings \( \tau \) in \( T_s \), up to length \( s' \), where \( \tau \supseteq \sigma_i \), for \( i \in \{1, 2\} \). Until we find so, the idea is to place \( \Phi \)-splits on every string which is a \( \Pi \)-boundary point of stage \( s \) (This will allow us to make \( \Psi \) partial on branches where we have \( \Phi \)-splittings). We can do this so as follows. Suppose \( \eta \supseteq \sigma \) is a \( \Pi \)-boundary point at stage \( s \) with respect to functional \( \Psi_n \). See if there exist two incompatible strings \( \eta_0, \eta_1 \) in \( \Phi^+ \) extending \( \Phi(\eta) \) whenever \( \Phi(\eta) \) is defined (if \( \Phi(\eta) \) is undefined at this stage,

Before we give the strategy for \( S_1 \), let us first explain the strategy for \( S_2 \). The strategies for \( S_1 \) and \( S_2 \) are invoked inside the (main) \( S_0 \)-strategy. The \( S_1 \) and \( S_2 \) strategies will execute their instructions depending if the node equipped with that strategy is in the \( \Pi \) or \( \Sigma \)-region of \( T_s \). For the \( S_2 \) strategy, for instance, if the flag value of \( \sigma \) is \( \Pi \), we execute \( S_{2, \Pi}(n) \); otherwise we execute \( S_{2, \Sigma}(n) \).

3.3. \( S_{2}(n) \)-strategy above \( \sigma \). We always try to diagonalize if possible. If we find out that \( \Xi(\Phi(\sigma)) \neq \sigma \) at any stage of any version of the \( S_2 \) strategy given below, we enumerate out from \( T \) every node incompatible with \( \sigma \) and cancel all strategies placed on them.

3.3.1. \( S_{2, \Pi}(n) \)-strategy instructions above \( \sigma \). If the flag value of \( \sigma \) is \( \Pi \), that is, if we are working in a \( \Psi_n \)-splitting region above \( \sigma \), then \( \Psi_n(\sigma) \) computes \( \sigma \) so things work out easier and the requirements are satisfied trivially. We just need to proceed with the non-recursiveness requirement to make sure no infinite path above \( \sigma \) is recursive. We place the \( D(n) \) strategy on two incompatible strings of \( \sigma \) and execute \( D(n) \) for both. At most one of them will get diagonalized and enumerated out of \( T \). Then we place \( S_2(n+1) \) strategy on the other string which remains in \( T \). This ensures that the \( S_{2, \Pi}(n) \)-strategy does not enumerate out \( \sigma \) from \( T \).

3.3.2. \( S_{2, \Sigma}(n) \)-strategy instructions above \( \sigma \). Suppose that we are at stage \( s \) and the flag value of \( \sigma \) with respect to \( \Psi_n \) is \( \Sigma \), and so there are no active \( \Psi_n \)-splits above \( \sigma \). Let \( \tau_0 \) and \( \tau_1 \) be two incompatible extensions of \( \sigma \) that are not yet enumerated out of \( T \). We ensure the following property:

For every \( \tau_0 \supseteq \tau_0 \) and \( \tau_1 \supseteq \tau_1 \), if \( \Psi_n(\tau_0) \) and \( \Psi_n(\tau_1) \) are defined and compatible with each other, then \( \Phi(\tau_0) \) and \( \Phi(\tau_1) \) are compatible. (*)&

Let \( \sigma_0 \supseteq \sigma \) be a node on which the next highest priority \( D(n) \)-strategy is placed. Let \( \sigma_1 \) and \( \sigma_2 \) be two successors of \( \sigma_0 \). We wait until stage \( s' \) such that \( \Psi_n \) is defined on all strings \( \tau \) in \( T_{s'} \), up to length \( s' \), where \( \tau \supseteq \sigma_i \), for \( i \in \{1, 2\} \). Until we find so, the idea is to place \( \Phi \)-splits on every string which is a \( \Pi \)-boundary point of stage \( s \) (This will allow us to make \( \Psi \) partial on branches where we have \( \Phi \)-splittings). We can do this so as follows. Suppose \( \eta \supseteq \sigma \) is a \( \Pi \)-boundary point at stage \( s \) with respect to functional \( \Psi_n \). See if there exist two incompatible strings \( \eta_0, \eta_1 \) in \( \Phi^+ \) extending \( \Phi(\eta) \) whenever \( \Phi(\eta) \) is defined (if \( \Phi(\eta) \) is undefined at this stage,
we do nothing). If they exist, choose $\eta'_0$ and $\eta'_1$ extending $\eta$ such that $\eta'_0$ and $\eta'_1$ are incompatible. We then place $S_2(n+1)$-strategies on $\eta'_0$ and $\eta'_1$ and enumerate axioms $\Phi(\eta'_0) \supset \eta_0$ and $\Phi(\eta'_1) \supset \eta_1$ for a choice of shortest possible length of strings $\eta_0$ and $\eta_1$. If they do not exist, let $\tau' \supset \Phi(\eta)$ be the longest extension of $\Phi(\eta)$ in $\Phi^+$. Choose two incompatible strings $\eta_0$ and $\eta_1$ extending $\eta$. Place $S_2(n+1)$-strategies on $\eta_0$ and $\eta_1$, and enumerate the axioms $\Phi(\eta_0) \supset \tau \ast 0$ and $\Phi(\eta_1) \supset \tau \ast 1$. Also enumerate $\tau' \ast 0$ and $\tau' \ast 1$ into $\Phi^+$.

Along the construction we may run into the problem of having either one of the extensions of $\sigma$ removed from $T$, which may interfere with the preservation of ($\ast$). To avoid this problem and to preserve ($\ast$), we use ‘reflections’ in the sense that we arrange $\Phi$ values in such a way that any $\Phi$ value above $\sigma_1$ will be available in $\Phi^+$ above $\sigma_2$. For this, we modify a subroutine which appeared in an earlier work [8]. The $\Phi^+$-regulation subprocedure given below will ensure that if the branch extending one of the incompatible extensions of $\sigma$, say $\eta$, is enumerated out from $T$ at a later stage, we will still be able to use the other $\Psi_n$-splitting pair via $\Phi^+$ to define the value of $\Phi(\eta)$. So after enumerating $\tau' \ast 0$ and $\tau' \ast 1$ into $\Phi^+$, we run the $\Phi^+$-regulation procedure on $\sigma$.

**Subroutine for $\Phi^+$-regulation procedure on argument $\sigma$:**

The idea is to preserve compatible $\Phi$ pairs above incompatible extensions of $\sigma$. Let $\sigma_1, \sigma_2 \supset \sigma$ be the immediate incompatible successors of $\sigma$. We ensure that if $\tau \supset \sigma_1$, then there exists some $\pi \supset \sigma_2$ such that each $\Phi(\tau) \subset \Phi(\pi)$ or $\Phi(\pi) \subset \Phi(\tau)$ if $\Phi(\tau)$ and $\Phi(\pi)$ are defined in $\Phi^+$.

Pick the longest string $\eta \in \Phi^+$, where $\sigma \subset \eta$, such that $\Phi(\eta)$ is defined and all $\eta_0 \subset \eta$ are complete for all $i \in \beta_{\eta_0}$. For every $\rho$ of length $\leq |\eta|$ and incompatible with $\eta$, we define an anti-chain of strings in $\Phi^+$ whose $\Phi$ values are compatible with $\Phi(\eta)$. For this we take $\Phi(\eta)$. Then for every $\rho$ of length $l \leq |\eta|$ such that $\rho \not\subset \eta$ and $\eta \not\subset \rho$, we enumerate $\rho$ into $\Phi^+$. Then we define the axiom $\Phi(\rho) \supset \Phi(\eta) \ast 0^l$, where $0^l$ denotes $l$ consecutive $0$'s whenever $\Psi(\rho)$ and $\Psi(\eta)$ are defined and compatible. This ensures that all incompatible extensions $\rho, \eta \supset \sigma$ have compatible values in the $\Phi$ domain whenever $\Psi(\rho)$ and $\Psi(\eta)$ are compatible (More details as to why this preserves the ($\ast$) property are given in Lemma 1.2). We then enumerate $\rho \ast 0$ and $\rho \ast 1$, as well as $\eta \ast 0$ and $\eta \ast 1$ into $\Phi^+$. This allows us to extend the possible domain of $\Phi$ for enumerating further axioms.

**End of subroutine**

The $\Phi^+$-regulation subprocedure is to ensure that every value of $\Phi$ above $\sigma_1$ is also available in $\Phi^+$ as a value for $\Phi$ above $\sigma_2$ in case one of them gets diagonalized. We call these $\Phi$ values reflections of each other.

Now $S_2(n)$ will search for a string $\sigma_i' \supset \sigma_i$ such that $\Xi(\Phi(\sigma_i')) \supset \sigma_i \upharpoonright k$ for some fixed $k \geq |\sigma|$. If there does not exist such string, we are fine as the requirement is satisfied trivially in this case. Otherwise, we declare every string above $\sigma_i \upharpoonright k$ and incompatible with $\sigma_i'$ to be terminal and enumerate them out from $T$. We also enumerate those strings from $T$ whose $\Phi$ values are defined and are incompatible with $\Phi(\sigma_i')$. We let $T_{s+1}$ to be the resulting tree. Note that if $\Xi(\Phi(\sigma_i')) \supset \sigma_i \upharpoonright k$ then $\Psi$ must be partial anywhere but above $\sigma_i'$ since otherwise $\Xi(\Phi(\sigma_i'))$ would compute $\sigma_{1-i}$ (see Figure 3). It is also worth noting that we ensure to keep at
least one node alive above $\sigma$. The reason is that if a string $\sigma$ is removed for being inconsistent with the computation of $\Xi(\Phi)$ on a fixed argument, then we must be keeping strings incompatible with $\sigma$ since one of the two incompatible strings must be consistent with the latter computation.

Regardless of which version we follow, after these instructions we place $S_2(n+1)$-strategies on strings of the least level on which no lower priority $S_2$-strategy is placed yet.

3.4. $S_1(n)$-strategy above $\sigma$. We try to diagonalize if possible as usual. That is, whenever we find that $\Theta_n(\Psi_n(\sigma)) \neq \Phi(\sigma)$, we declare every string incompatible with $\sigma$ to be terminal and enumerate them out from $T$, and so $S_1(n)$ is satisfied above $\sigma$. Until we falsify the hypothesis of the $S_1(n)$ requirement, the strategy is instructed to perform the following in accordance with the flag value of $\sigma$, i.e., depending if $\sigma$ is in the $\Sigma$ or the $\Pi$-region of $T_\psi$ with respect to $\Psi_n$.

3.4.1. $S_{1,\Pi}(n)$-strategy instructions above $\sigma$. Suppose that $\sigma$ has an active $\Psi_n$-split. Then we will have no problem in satisfying the requirement trivially since we automatically get $\Psi_n(A) \geq_T A$ and we have a win over that particular $\Psi$.

We nevertheless need to apply the non-recursiveness strategy to make sure no infinite path above $\sigma$ is recursive. We place the $D(m)$-strategy on two incompatible strings $\tau_0, \tau_1$ such that $\tau_0, \tau_1 \supset \sigma$ and we execute $D(m)$ on both for least such $m$ that has not been picked yet for the non-recursiveness requirement. At most one
of them will get diagonalized, say \( \tau_i \), and enumerated out of \( T \). We then place \( S_2(n+1) \)-strategy on \( \tau_{1-i} \). Also define \( \tau_{1-i} \) to be a new \( \Pi \)-boundary point.

3.4.2. \( S_1, \Sigma(n) \)-strategy instructions above \( \sigma \). Suppose that an \( S_1(n) \)-strategy is placed on \( \sigma \) and that there are no active \( \Psi_n \)-splits above \( \sigma \). Given \( T_s \) at stage \( s \) we perform the following instructions:

1. Fix some new witness \( l \in \omega \).
2. See if there are two incompatible strings \( \sigma_1 \) and \( \sigma_2 \) in \( T_s \), extending \( \sigma \), such that \( \Psi_n(\sigma_1) \) and \( \Psi_n(\sigma_2) \) are defined up to \( l \). If so, then
   
   (i) Place a \( D(m) \)-strategy on \( \sigma_1 \) and \( \sigma_2 \) for least \( m \) that has not been picked yet, and define axioms for \( \Phi \) such that \( \Phi(\sigma_1) \) and \( \Phi(\sigma_2) \) are incompatible up to \( l \).
   
   (ii) Keep both extensions \( \sigma_1 \) and \( \sigma_2 \) in \( T_s \) until we see either one of two things happen:
      
      (a) There exists an extension of \( \sigma_i \), for some \( i \in \{1, 2\} \), say \( \sigma'_i \supset \sigma_i \), such that \( \Theta(\Psi_n(\sigma'_i)) = \Phi(\sigma_i) \).
      
      (b) The \( D(m) \)-strategy decides to remove one of \( \sigma_i \).

Now one of the two cases may happen for \( \Theta \) as the construction goes:

(i) \( \Theta \) may appear finitely often.
(ii) \( \Theta \) may appear infinitely often.

Case (i): Consider first the finite outcome. It may be that \( \Theta \) stops appearing above \( \sigma \). If we have a finite outcome, then we have a win over \( \Theta \) and so there is nothing to prove as we can satisfy the \( S_1 \) requirement above \( \sigma \).

Case (ii): We follow a similar argument as in the \( \Sigma_2 \) outcome for \( S_2 \). We keep both \( \sigma_1 \) and \( \sigma_2 \) extendible until the hypothesis in the \( S_1 \) requirement is satisfied. If we find that \( \Theta(\Psi_n(\sigma'_i)) = \Phi(\sigma_i) \), we define all \( \tau \supset \sigma_i \) to be terminal such that \( \tau \) is incompatible with \( \sigma'_i \). Note that when \( \Theta \) of \( \Psi_n \) is defined above \( \sigma \) which is compatible with \( \sigma_1 \), then \( \Psi \) cannot be defined above for \( \sigma_2 \) because the range of \( \Psi \) is below \( \sigma_1 \). If we later decide to kill the branch on which \( \Theta \) is defined, then \( \Theta \) may now get defined above \( \sigma_2 \). But let us suppose this does not happen, or at least it happens finitely many times. This means that \( \Theta \) will be defined along a unique infinite path \( A \) and \( \Psi_n \) will be partial on every other path (see Figure 4).

So for this unique infinite path on which \( \Theta \) is defined, we have \( \Psi_n(A) \geq_T A \) and \( \Phi(A) \geq_T A \). This infinite path, we will prove in Lemma 4.3, is of r.e. degree.

3.5. \( D(n) \)-strategy above \( \sigma \). Finally we explain the strategy \( D(n) \) for the non-recursiveness requirements. \( D(n) \) works in accordance with \( S_1 \) and \( S_2 \). Hence, it will have two versions, one for \( S_1 \) and the other for \( S_2 \).

Instructions of \( D(n) \) for \( S_1 \) at stage \( s \):

Let \( \tau \) be a string in \( T_s \) equipped with a \( D(n) \)-strategy and suppose that \( D(n) \) has not yet acted on \( n \), in the sense that \( n \) is a newly picked witness. We find the least \( \tau' \supset \tau \) such that \( \tau' \) is the least string on which no \( S_1(n) \)-strategy is placed. Let \( \tau'_0 \) and \( \tau'_1 \) be two strings extending \( \tau' \) such that \( \tau'_0 \) and \( \tau'_1 \) are incompatible. Define \( \Phi(\tau'_0) \) and \( \Phi(\tau'_1) \) to be two incompatible strings. If we find at some later stage that
\[ \Theta(\Psi(A)) = \Phi(A) \]

Figure 4. If \( \Psi_n \) is total on a unique infinite path \( A \) and there are infinitely many \( \Phi \)-splits, \( \Psi_n(A) \geq_T A \) and \( \Phi(A) \geq_T \Psi_n(A) \).

the \( n \)-th Turing functional \( \Psi_n \) extends one of \( \tau'_i \), for \( i \in \{0, 1\} \), then we declare all extensions of \( \tau'_i \) which are incompatible with \( \tau'_{1-i} \) as terminal, enumerate them out of \( T \) and remove all strategies from these strings. Then place \( D(n+1) \)-strategy on two incompatible extensions of \( \tau'_{1-i} \).

Instructions of \( D(n) \) for \( S_2 \) at stage \( s \):

Suppose that a \( D(n) \)-strategy is placed on \( \sigma \). We want the \( D(n) \)-strategy to decide which one of the two incompatible extensions of \( \sigma \) to keep in \( T \) at stage \( s \). When working with \( S_2 \) requirements we have to be careful that the domain of \( \Phi \) should not exceed the domain of \( \Psi_n \) before we decide which path to choose for diagonalization. Let \( \sigma_0 \) and \( \sigma_1 \) be two incompatible extensions of \( \sigma \) in \( T_s \). In order to keep at least one of \( \sigma_0 \) and \( \sigma_1 \) extendible, we leave reflections of \( \Phi \) values above both \( \sigma_0 \) and \( \sigma_1 \) in case the \( D(n) \)-strategy at a later stage decides to remove one of them and we make sure that \( \Phi \) is not defined until \( \Psi_n \) gets defined. For this it suffices to ensure that it is not the case that every extendible \( \sigma'_0 \supseteq \sigma_0 \) has a \( \Phi \) value incompatible with its reflection. More precisely, we make sure that above \( \sigma_0 \), there exist some \( \sigma'_0 \) for which there exists some \( \sigma'_1 \supseteq \sigma_1 \) such that \( \Phi(\sigma'_0) \) and \( \Phi(\sigma'_1) \) are compatible if \( \Psi_n(\sigma'_0) \) and \( \Psi_n(\sigma'_1) \) are compatible. This is ensured by the \( (\ast) \) property, which is preserved by the \( \Phi^+ \)-regulation procedure. So we only enumerate \( \Phi \) axioms while leaving a reflection of either of \( \Phi(\sigma'_0) \) or \( \Phi(\sigma'_1) \) in \( \Phi^+ \). Again, we do this because we want to preserve \( (\ast) \) for making \( \Phi \) values of extensions compatible whenever the \( \Psi \) values are compatible. If it were the case that all \( \Phi(\sigma'_0) \) were incompatible with \( \Phi(\sigma'_1) \), then assuming that \( \Phi(\sigma'_0) \downarrow \) and \( \Phi(\sigma'_1) \downarrow \), and that
there are no active $\Psi_n$-splits above $\sigma$, we would not be able to make the reflections compatible with each other which is necessary for the preservation of ($\ast$). The rest of the instructions is similar to the instructions of $\mathcal{D}(n)$ for $\mathcal{S}_1$.

### 3.6. Main construction

We are now ready to define the main recursive construction, which is just initiated by enumerating a $\Psi_0$-splitting subtree of $2^{<\omega}$.

**Stage 0.** Define $T_0 = 2^{<\omega}$. We define $\lambda$ to be the II-boundary point initially. We let $\Phi^+ = \{\lambda\}$ and enumerate $\Phi(\lambda) \supset \lambda$.

**Stage $s + 1.$** For any $\sigma$ of length $s + 1$ in $T_{s+1}$, carry out step $s$ of the level $0$ construction inside $T_0$ above $\sigma$.

The construction generates, at each stage, a $\Psi_{n+1}$-splitting subtree over the $\Psi_n$-splitting path, on the regions where $\Psi_n$ is partial.

We let $T = \bigcap_s T_s$ be the co-r.e. tree produced by the construction. Now, the construction will never enumerate $\lambda$ out of $T$. Furthermore, since no terminal nodes are left in $T_s$, $T$ must be infinite and so $[T] \neq \emptyset$ (see Lemma 4.1).

### 4. Verification

We first argue that the constructed $\Pi^0_1$ class is non-empty and contains no recursive members.

**Lemma 4.1.** $[T]$ is non-empty and it does not contain a recursive member.

**Proof.** It should be clear from both versions of the $\mathcal{D}(n)$-strategy that if $A \in [T]$ and $A = \Psi_n$ for some $n$, then there is some stage $s$ where $\sigma \subset A$ gets enumerated out of $T_s$ when we witness $\sigma \subset \Psi_n(\emptyset)[s]$. Hence, if $A$ is recursive then $A \notin [T]$. If $\sigma$ is a node with a $\mathcal{D}(n)$-strategy, the fact that we run the $\mathcal{D}(n)$-strategy on each incompatible extension $\sigma_0, \sigma_1 \supset$ ensures that $\mathcal{D}(n)$ does not enumerate out $\sigma$ from $T$.

To prove that $[T]$ is non-empty, it is clear from the instructions that if $\sigma$ is a node equipped with an $\mathcal{S}_i$-strategy, for $i \in \{0, 1, 2\}$, then $\sigma$ is not enumerated out of $T$. Also when we enumerate out some $\tau$ from $T$, we also enumerate out of $M$ all $\sigma \in T$ such that either $\tau \subset \sigma$ or $\sigma \subset \tau$ and $\sigma \sim \tau$. The latter property preserves the property that $T_s$ has no terminal nodes. Then these two facts together ensure that $T_s$ is an infinite perfect tree. Hence, $T = \bigcap_s T_s$ is an infinite perfect tree, and so $[T] \neq \emptyset$ by compactness of Cantor space.

Next we show that the domain of $\Psi$ does not exceed that of $\Phi$. This ensures that $\Phi(A)$ computes $\Psi_n(A)$ for any $n \in \omega$.

**Lemma 4.2.** For any $A \in [T]$, $\Phi(A)$ is total if $\Psi(A)$ is total.

**Proof.** Assume that $A \in [T]$ and $\Psi(A)$ is total. Suppose that there exists some $\sigma$ and a stage $s$ after which $\Phi(\tau)$ is undefined for all $\tau \supset \sigma$ at all later stages $s' \geq s$. The totality of $\Psi(A)$ is ensured as follows whenever we define an axiom for $\Phi$: We take the least $n \in \omega$ such that $\Psi(\sigma; n) \uparrow$. We then make sure that for all $m < n$ and $j < m$, if $\Psi(\sigma; m) = \tau$ and if $\Phi(\sigma; j) = \eta$ such that $\eta \subset \tau$, then enumerate the axiom $\Phi(\sigma; j) = \tau$ for the longest string $\pi \in \Phi^+$ such that $\pi \subset \tau$. We then enumerate $\tau \ast 0$ and $\tau \ast 1$ into $\Phi^+$. If we follow this convention this runs contrary to
what we supposed for a contradiction. Hence, if there exists some $k \in \omega$ for which $\Phi(\sigma; k) \uparrow$, then there exists some $l < k$ for which $\Psi(\sigma; l) \uparrow$. □

We now prove the last lemma needed to complete the proof of the main theorem. We shall first describe the success criteria for the execution of requirements. Let us say that, for each $i \in \{0, 1, 2\}$, success conditions of the $S_i$ and $D$ requirements depend on the satisfaction of two conditions:

(i) Step 0 of the requirements are carried out. If step $s$ of the requirement is carried out, then there is a stage $t > s$ at which the requirement is carried out.

(ii) When a strategy is permanently satisfied after stage $t$, i.e., there is a stage $t$ such that the strategy is satisfied for every $t > s$, no string $\sigma$ above which the strategy is permanently satisfied is enumerated out of $T$.

**Lemma 4.3.** Suppose that the success conditions for the $S_i$-requirements, for each $i \in \{0, 1, 2\}$, are satisfied on infinite paths of $T$. Then any $A \in [T]$ in the $\Sigma$-region of $T$ for which $\Psi_n(A)$ is total is of r.e. degree.

**Proof.** The $S_0(n)$-strategy enumerates a $\Psi_n$-splitting subtree of $T$. So with the $S_0$-requirements we either end up with having a $\Pi_2$ outcome or $\Sigma_2$ outcome. Suppose that $A \in [T]$ and $A$ is in the $\Pi$-region of $T$, i.e., for every $\sigma \subset A$ there exist $\tau_1 \supset \sigma$ and $\tau_2 \supset \sigma$ in $T$ such that $\Psi_n(\tau_1; x) \downarrow \neq \Psi_n(\tau_2; x)$ for some $x \in \omega$. In this case, $\Psi_n(A) \supset T A$ and so the theorem is satisfied trivially. We now prove that if $A$ lies in the permanent $\Sigma$-region of $T$ such that $\Psi_n(A)$ is total, then $A$ is of r.e. on that infinite branch of $T$. For the $S_2$ requirements we have the following cases:

(i) If $\Theta$ appears finitely often, we have the Outcome 3 since in this case we win over $\Theta$.

(ii) If $\Theta$ appears infinitely often, we get Outcome 4. That is, we have that $\Psi_n(A)$ computes $A$ via $\Theta$ on a unique infinite path, and $\Psi_n$ is partial on any path incompatible with $A$. In this case, we argue below that $A$ is of r.e. degree.

We give the proof for the $S_2$ requirements, but the same argument can be adapted to $S_1$ requirements as well. Suppose that $A$ is the unique path in $[T]$ such that $\Psi_n(A)$ is total. We can construct an r.e. set $W$ which is Turing equivalent to $A$ as follows. Let $D_\sigma$ denote the node $\sigma$ with a $D(n)$-strategy placed on it. Define $W$ to be set of all $D_\sigma$ such that there exists some stage $s$ where $D_\sigma$ is in the $\Pi$-region with respect to $\Psi_n$ and $D_\sigma$ is active at stage $s$. More precisely, we let

$$W = \{ \sigma \subset A : \exists s \text{ such that } \sigma \text{ is on the subtree with } \Psi_n(A) \supset T A \text{ after stage } s \}.$$ 

Clearly $W$ is an r.e. set since $W$ is a $\Sigma^0_1$ set. We claim that $W \equiv_T A$.

(i) The fact that $W \geq_T A$ is clear since $A$ is the unique infinite path in $W$.

(ii) We argue that $W \leq_T A$. We see if $D_\sigma$ is used in the construction and so whether it remains in the tree at stage $|\sigma|$. If $D_\sigma$ is used then our tree at any stage will look like an antichain of strings on a path $A$. Now suppose that $\sigma$ is a string from which we start our strategy. Given $\gamma \in 2^{<\omega}$, there are three cases to find out whether or not $\gamma \in W$. First case is that $\gamma$ could be off the whole tree. So if $\gamma$ is incompatible with every $\sigma$, then we know that $\gamma \notin W$. Secondly, if it is the case that $\gamma \supset \sigma$ and $\gamma \subset A$, then $\gamma \in W$ and $\Psi(A) \geq_T A$ whenever $\gamma$ is on the infinite path on which $\Psi(A)$ is total. Finally, suppose that $\gamma \supset \sigma$ but $\gamma$ is incompatible with every $\tau \subset A$. In this case, we keep finding the node with the next highest priority...
D-strategy until, at some stage \( t \), we hit \( \gamma \) and we see whether \( \gamma \) is extended by an active node of length \( t \). For convenience we may suppose that the extension has the same length as the length of the stage of the construction. So a string is in \( W \) if it is in \( W[t] \) enumerated at stage \( t \) where we put active node compatible with \( A \). Then, \( D_\sigma \in W \) if and only if \( D_\sigma \in W \) by stage \( |\sigma| \). □

This completes the proof of Theorem 3.2. Hence, \([T]\) is a non-empty special \( \Pi^0_1 \) class in which no member is a minimal cover for any set. Thus, class of degrees of minimal cover is a not basis for \( \Pi^0_1 \) classes.

References

1. D. Cenzer: \( \Pi^0_1 \) Classes in Recursion Theory; in Handbook of Computability Theory, North-Holland, Studies in Logic and the Foundations of Mathematics, 140: 37-89 (1999).
2. S. B. Cooper: Computability Theory, Chapman & Hall, CRC Press, Boca Raton, FL, New York, London (2004).
3. A. Çevik: Antibasis theorems for \( \Pi^0_1 \) classes and the jump hierarchy. Archive for Mathematical Logic, 52(1-2): 137-142 (2013).
4. A. Çevik: \( \Pi^0_1 \) choice classes. Mathematical Logic Quarterly, 62(6): 563-574 (2016).
5. A. Çevik: An effectively closed set with no join property, Mathematical Logic Quarterly, 67(3): 313-320 (2021).
6. D. E. Diamondstone, D. D. Dzhafarov, R. I. Soare: \( \Pi^0_1 \) Classes, Peano Arithmetic, Randomness, and Computable Domination. Notre Dame J. Formal Logic, 51(1): 127-159 (2010).
7. R. Downey and D. Hirshfeldt: Algorithmic Randomness and Complexity, Springer-Verlag (2010).
8. A. Ehrenfeucht: Separable theories, Bull. Acad. Polon. Sci. Sér. Sci. Math. Astronom. Phys., 9(1961): 17-19 (1969).
9. M. J. Groszek and T. A. Slaman: \( \Pi^0_1 \) classes and minimal degrees, Annals of Pure and Applied Logic, 87(2): 117-144 (1997).
10. C. G. Jockusch and S. G. Simpson: Minimal degrees, hyperimmune degrees, complete extensions of arithmetic, Abstract no. 781-E10, Abstract of papers presented to the AMS, 1: 546 (1980).
11. C. G. Jockusch and R. I. Soare: \( \Pi^0_1 \) classes and degrees of theories, Trans. Amer. Math. Soc., 175: 33-56 (1972).
12. C. Jockusch, R. I. Soare: Degrees of members of \( \Pi^0_1 \) classes, Pacific J. Math., 40: 605-616 (1972).
13. T. Kent, A. E. M. Lewis: On the Degree Spectrum of a \( \Pi^0_1 \) Class, Trans. Amer. Math. Soc., 362: 5283-5319 (2010).
14. S. C. Kleene: Recursive predicates and quantifiers, Trans. Amer. Math. Soc., 53: 41-73 (1943).
15. G. Kreisel: Analysis of the Cantor-Bendixson theory by means of the analytic hierarchy, Bull. Acad. Polon. des Sciences, Sr. Math., Astronom. et Phys., 7: 621-626 (1959).
16. A. E. M. Lewis: \( \Pi^0_1 \) classes, strong minimal covers and hyperimmune-free degrees, Bulletin of the London Mathematical Society, 39(6): 892-910 (2007).
17. G. E., Sacks: A minimal degree below \( \theta' \), Bull. Amer. Math. Soc., 67: 416-419 (1961).
18. G. E., Sacks: The recursively enumerable degrees are dense, Annals of Mathematics, 80: 300-312 (1964).
19. J. R. Shoenfield: Degrees of models, Journal of Symbolic Logic, 25: 233-237 (1960).
20. I. Sasso: A Cornucopia of Minimal Degrees, Journal of Symbolic Logic, 35(3): 383-388 (1970).
21. R. I. Soare: Recursively Enumerable Sets and Degrees, Perspectives in Mathematical Logic. Springer-Verlag, Berlin (1987).
22. C. E. M. Yates: Initial segments of the degrees of unsolvability. Part II: Minimal degrees, Journal of Symbolic Logic, 35: 243-266 (1970).

Gendarmerie and Coast Guard Academy, Ankara, Turkey

Email address: a.cevik@hotmail.com