Journal of Mathematical Logic
Vol. 18, No. 1 (2018) 1850001 (47 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0219061318500010

Degrees containing members of thin Π^0_1 classes are dense and co-dense

Rodney G. Downey
School of Mathematics, Statistics and Operational Research
Victoria University of Wellington, Wellington, New Zealand
Rod.Downey@msor.vuw.ac.nz

Guohua Wu
Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore 637371, Singapore
guohua@ntu.edu.sg

Yue Yang
Department of Mathematics, National University of Singapore
10 Lower Kent Ridge Road, Singapore 119076, Singapore
matyangy@nus.edu.sg

Received 20 November 2016
Accepted 30 October 2017
Published 19 December 2017

In [Countable thin Π^0_1 classes, *Ann. Pure Appl. Logic* **59** (1993) 79–139], Cenzer, Downey, Jockusch and Shore proved the density of degrees (not necessarily c.e.) containing members of countable thin Π^0_1 classes. In the same paper, Cenzer *et al.* also proved the existence of degrees containing no members of thin Π^0_1 classes. We will prove in this paper that the c.e. degrees containing no members of thin Π^0_1 classes are dense in the c.e. degrees. We will also prove that the c.e. degrees containing members of thin Π^0_1 classes are dense in the c.e. degrees, improving the result of Cenzer *et al.* mentioned above. Thus, we obtain a new natural subclass of c.e. degrees which are both dense and co-dense in the c.e. degrees, while the other such class is the class of branching c.e. degrees (See [P. Fejer, The density of the nonbranching degrees, *Ann. Pure Appl. Logic* **24** (1983) 113–130] for nonbranching degrees and [T. A. Slaman, The density of infima in the recursively enumerable degrees, *Ann. Pure Appl. Logic* **52** (1991) 155–179] for branching degrees).

Keywords: Π^0_1 classes; Turing degrees; density.

Mathematics Subject Classification 2010: 03D25, 03D80

1. Introduction

The study of Π^0_1 classes[a] and, in particular, the degrees of their members, is a longstanding branch of computability theory, which has many applications. One well-known example in this area is the *Low Basis Theorem* of Jocksuch and Soare [16] which asserts that any nonempty Π^0_1 class has a member of low (actually superlow) degree.

Π^1_0 classes can be thought of as the collections of paths $[T]$ through a computable subtree T of $2^{<\omega}$. There are many ties between areas of logic and effective mathematics, including logical theories of Peano Arithmetic, constructions in effective algebra and analysis, colorings of computable graphs, and algorithmic randomness. As an archetypal example, the many one-degree cone of orderings of a computable formally real field is in an effective one-to-one correspondence with the collections of members of nonempty Π^0_1 classes [19]. Π^0_1 classes also have deep connections with proof theory and reverse mathematics, and this Metakides–Nerode theorem has a re-interpretation that over RCA_0, the statement that "every countable formally real field has an ordering" is equivalent to the system Weak König's Lemma (WKL_0) (see [15]). There are many other examples and the reader is referred to the survey papers [2, 5] and Cenzer's book draft [6]. For basics of Π^0_1 classes, please also refer to Soare's book [22], or Cenzer *et al.*'s paper [3], Jockusch and Soare's paper [17].

Since a Π^0_1 class might be finite, in which case all the members are computable, little else can be said without additional hypotheses. One example of such an additional hypothesis is that a Π^0_1 class $\mathcal{P}$ has no computable members. Then not only must it have members of low degree, but it must have members of all possible jumps [16]. (That is for all $X \geq \emptyset'$, there exists $Y \in \mathcal{P}$ with $Y' \equiv_T X$.) The classes we are interested in this paper are *thin* Π^0_1 *classes*, where a Π^0_1 class $\mathcal{P}$ is *thin* if it is not clopen, and for all Π^0_1 subclasses $\mathcal{Q}$ of $\mathcal{P}$, there is a clopen set $\mathcal{U}$ such that $\mathcal{Q} = \mathcal{P} \cap \mathcal{U}$.

Thin Π^0_1 classes were introduced by Martin and Pour-El [18] (under duality), where Martin and Pour-El constructed an axiomatizable essentially undecidable theory T with few extensions, meaning that if T' extends T then T' is a finite (and hence a principal) extension of T. Thin Π^0_1 classes are the natural analog of hyperhypersimple sets when considered in the lattice of Π^0_1 classes, since Cholak, Coles, Downey and Herrmann [8] proved that a Π^0_1 class $\mathcal{P}$ is thin if and only if it is not clopen and the lattice of Π^0_1 subclasses of $\mathcal{Q}$ modulo finite differences forms a (Δ^0_2) Boolean algebra. Moreover, as an analog of Soare's theorem [21] that maximal sets form an orbit in the lattice of c.e. sets, Cholak *et al.* proved that the *perfect* thin classes (i.e. the lattice of subclasses is isomorphic to a free Boolean algebra) formed an orbit in the automorphism group of the lattice of Π^0_1 classes. In passing, we remark that if you assign a degree to a class as being the Turing degree of the sets of nonextendible nodes in a tree T which represents $\mathcal{P}$, then the degrees of Π^0_1

[a]We are using the convention that when we say "Π^1_0 classes", we mean "computably bounded Π^0_1 classes".

subclasses in the orbit are exactly the *array noncomputable* degrees (see [11]). This orbit is essentially the only orbit, as discovered by Downey and Montalban [12]. Thus thin Π^0_1 classes form a natural and also fascinating class with many natural associations with the c.e. degrees.

The study of the fine structure of degrees of members of thin Π^0_1 classes began with Cenzer *et al.* in [4] where the main focus was on the case when the Boolean subalgebra of subclasses was countable. The simplest such class is called a *minimal* class, where the Boolean algebra is the Boolean algebra of finite and co-finite subsets of $\mathbb{N}$. Equivalently, $\mathcal{P}$ is minimal if $\mathcal{P}$ is infinite and every subclass is either finite or co-finite.

In [4], it is shown that if $\mathbf{a}$ is a degree of a member of a thin class, then it has to be somewhat "lowly" in the generalized sense: $A' \leq_T A \oplus \emptyset''$ whenever A is a member of a thin Π^0_1 class (see [4, Theorem 2.10]). Of significant interest to this paper is the c.e. degrees of members of thin classes.

Theorem 1.1 (Cenzer, Downey, Jockusch and Shore [4]). (1) *The degrees* (*not necessarily c.e.*) *containing members of minimal Π^0_1 classes are dense in the c.e. degrees.*

(2) $\mathbf{0}'$ *contains a member of a minimal Π^0_1 class.*

(3) *There exists a c.e. degree $\mathbf{a}$ containing no member of thin Π^0_1 classes.*

In [13], Downey, Wu and Yang began to extend the results shown in [4]. The *rank* of a computable tree is its Cantor–Bendixson rank, and the rank of a real (i.e. a member of the Cantor space) is the least rank of a computable tree containing that real.

Theorem 1.2 (Downey, Wu and Yang [13]). *For any Δ^0_2 degree $\mathbf{a}$ and any computable ordinal α*:

(1) $\mathbf{a}$ *contains a real of rank α.* (*This improves earlier results of Cholak and Downey* [9], *and Cenzer and Smith* [7].)
(2) *If $\mathbf{a}$ contains a member of a thin Π^0_1 class*, *then $\mathbf{a}$ contains a member of a thin Π^0_1 class of rank α.*
(3) *This fails outside the Δ^0_2 degrees.*

The first result we prove in this paper is the following.

Theorem 1.3. *The c.e. degrees* not *containing members of thin classes are dense in the c.e. degrees. That is*, *for any c.e. degrees $\mathbf{b} < \mathbf{a}$*, *there is a c.e. degree $\mathbf{c}$ with $\mathbf{b} < \mathbf{c} < \mathbf{a}$ such that $\mathbf{c}$ contains no members of thin classes.*

By Sacks density theorem, we only need to require $\mathbf{c}$ with $\mathbf{b} \leq \mathbf{c} \leq \mathbf{a}$.

We will then improve a result of Cenzer *et al.* in [4], where it was shown that the degrees containing members of thin Π^0_1 classes are dense in the c.e. degrees. It was left open whether these degrees can be c.e. or not. We will show in the second half that the degrees can be constructed as c.e.

Theorem 1.4. *The c.e. degrees containing members of thin Π^0_1 classes are dense in the c.e. degrees. That is, for any c.e. degrees $\mathbf{b} < \mathbf{a}$, there is a c.e.degree $\mathbf{d}$ with $\mathbf{b} < \mathbf{d} < \mathbf{a}$ such that $\mathbf{d}$ contains a member of some thin Π^0_1 classes.*

Thus the c.e. "thin" degrees are really unusual. They do not depend on rank, and they form one of the very few (natural) classes of constructions which are both dense and co-dense in the c.e. degrees. The only other major class that springs to mind is the branching and nonbranching degrees (see [14] for density of nonbranching degrees and [20] for density of branching degrees.)

It is routine to show that the thin/nonthin degrees have no implications with the branching/nonbranching degrees. This can help us to understand the fundamental ideas of the main construction, which turns out to be fairly nonroutine. The proof of Theorem 1.3 can be viewed as a $\mathbf{0}'''$ argument, where dealing with the movement from right to left in the priority tree construction involves a feature of respecting certain historical "commitments" generated by "priority inversions" (a feature of $\mathbf{0}'''$ arguments involving local versus global priority, i.e. the priority of the mother versus that of the children) commitments in a way that is quite novel. This aspect will be seen in the construction. Furthermore, anyone who has worked with tree arguments, particularly with $\mathbf{0}'''$ ones (such as Ambos-Spies, Hirschfeldt and Shore [1] or Slaman [20]) in the setting of density theorems, is aware that synchronization of permitting and the tree machinery tends to be intricate. For this reason we will work up quite slowly to the construction.

The proof of Theorem 1.4 is also rather novel and the main complexity stems from the definition of the uses, and we believe that the method may be more widely applicable.

The paper is organized as follows. In Sec. 2, we give a brief description of CDJS's constructions of a single c.e. *NONTHIN* degree (a c.e. degree containing no member of any thin Π^0_1 class), and in Sec. 3, we will show how to combine CDJS's construction with the density arguments, and provide a full analysis of outcomes and interactions among strategies. In Sec. 4, we will give the construction of C, and show in Sec. 5 that the constructed set C satisfies all the requirements. In Sec. 6, we present the basic idea of the proof of Theorem 1.4, followed with the construction, and in Sec. 7, we will verify that the construction works.

2. CDJS's Construction of a NONTHIN Degree

We now construct a c.e. set C such that any set Turing equivalent to C cannot be a member of any thin Π^0_1 class. C is constructed to meet the following requirements:

$\mathcal{R}_e$: if $\Phi_e(C)$ and $\Psi_e(\Phi_e(C))$ are both total, such that $\Psi_e(\Phi_e(C)) = C$ and that $\Phi_e(C)$ is in Π^0_1 class $[T_e]$, then $[T_e]$ is not thin.

To satisfy one requirement $\mathcal{R}_e$, we will construct a subtree S_e of T_e, to witness that $[T_e]$ is not thin. We will satisfy $\mathcal{R}_e$ by satisfying the following subrequirements:

$\mathcal{S}_{e,i}$: There exists an interval $(x_{e,i}, z_{e,i})$ such that $[T_e]$ contains a branch (at least one) extending $\Phi_e(C) \upharpoonright x_{e,i}$, but not $\Phi_e(C) \upharpoonright z_{e,i}$. For convenience, we say that $[T_e]$ contains a branch in interval (or, region) $(x_{e,i}, z_{e,i})$.

- If i is even, then all nodes in T_e extending $\Phi_e(C) \upharpoonright x_{e,i}$, but not $\Phi_e(C) \upharpoonright z_{e,i}$, will be *put* on S_e. We will say that *all nodes on T_e in the region $(x_{e,i}, z_{e,i})$ are on S_e. These nodes are called plus-nodes.*
- If i is odd, then all nodes in T_e extending $\Phi_e(C) \upharpoonright x_{e,i}$, but not $\Phi_e(C) \upharpoonright z_{e,i}$, will be *terminated* on S_e. We will say that *all nodes on T_e in the region $(x_{e,i}, z_{e,i})$ are terminated on S_e. These nodes are called minus-nodes.*

Note that if all the $\mathcal{S}_{e,i}$-subrequirements are satisfied, then $[S_e]$ is a subclass of $[T_e]$, containing infinitely many branches of $[T_e]$ (guaranteed by plus-nodes), and at the same time, omitting infinitely many branches of $[T_e]$ (guaranteed by minus-nodes). This shows that $[T_e]$ is thin.

The construction of C will be conducted on a priority tree, T. Let τ be an $\mathcal{R}_e$ strategy, and τ has two outcomes, 1 and gw, where 1 indicates that the construction of the corresponding tree S_e is initiated, and gw denotes the case that a substrategy α eventually enumerates its y into C, securing a global win for τ.

Let α be an $S_{e,i}$-substrategy of τ. Define the length of agreement function $\ell(\alpha, s)$, where s is an α-stage, as usual:

$$\begin{aligned}\ell(\alpha, s) = \max\{x < s \colon \forall y < x[\Psi_e(\Phi_e(C))(y)[s]] = C_s(y) \text{ and}\\ \forall z < \psi_{e,s}(y)[\Phi_e(C)(z)[s] \text{ converges}] \text{ and } \Phi_e(C)[s]\upharpoonright\psi_{e,s}(y) \text{ is on } T_{e,s}\}.\end{aligned}$$

Suppose that α is a *minus*-substrategy, i.e. i is odd. α proceeds as follows:

(1) Select x_α, and wait for $\Phi_e(C)(x_\alpha)$ to converge, and then wait for $\ell(\alpha, s) > x_\alpha$.
(2) After seeing that $\ell(\alpha, s) > x_\alpha$, select y_α, bigger than $\varphi_e(x_\alpha)$, the use of $\Phi_e(C)(x_\alpha)$, and wait for $\Psi_e(\Phi_e(C))(y_\alpha)$ to converge to 0, and that $\Phi_e(C)$ to converge on all numbers $\leq \psi_e(y_\alpha)$.
(3) After seeing that $\ell(\alpha, s) > y_\alpha$, select z_α as a number bigger than $\psi_e(y_\alpha)$.
So far, we have selected a triple $(x_\alpha, y_\alpha, z_\alpha)$.
(4) Wait for $\Phi_e(C)(z_\alpha)$ to converge.
(5) When $\Phi_e(C)(z_\alpha)$ converges, declare that the nodes on S_e in the region (x_α, z_v) are terminated (recall that α is a *minus*-substrategy). Simultaneously, wait for a stage t such that all nodes in T_e in the region (x_α, z_α) become dead.
(*If there is no such a stage t, then $[T_e]$ contains at least a branch in the region (x_α, z_α).*)
(6) After we see that all nodes in the region (x_α, z_α) are dead, put y_α into C, and simultaneously, wait for $\Phi_e(C)(z_\alpha)$ to converge.
Note that if our assumption is correct, i.e. both $\Phi_e(C)$ and $\Psi_e(\Phi_e(C))$ are total, $C = \Psi_e(\Phi_e(C))$ and $\Phi_e(C)$ is in $[T_e]$, then we can never have a chance to put y_α into C. That is, if y_α is put into C, then $\mathcal{R}_e$ is satisfied via one of the following:

(a) $\Phi_e(C)(z_\alpha)$ does not converge anymore, then $\Phi_e(C)$ is not total.
Here, we assume that if $\Phi_e(C)(z_\alpha)$ converges, then $\Phi_e(C)(z)$ converges for all $z \leq z_\alpha$.

(b) $\Phi_e(C)(z_\alpha)$ converges later, but there exists a $z \in (x_\alpha, z_\alpha)$ such that $\Phi_e(C)(z)$ has a different value.
If so, $\Phi_e(C)$ will not be a branch in $[T_e]$, because a necessary condition of putting y_α into C is that $[T_e]$ contains no branch extending $\Phi_e(C)[t] \upharpoonright x_\alpha$, but not $\Phi_e(C)[t] \upharpoonright z_\alpha$.

(c) $\Phi_e(C)(z_\alpha)$ converges later, and the value is the same as those at stage t.
In this case, $C(y_\alpha) \neq \Psi_e(\Phi_e(C))(y_\alpha)$ as up to z_α (hence, up to $\psi_e(y_\alpha)$), $\Phi_e(C)$ [values here] is the same as before.
This implies that the computation $\Psi_e(\Phi_e(C))(y_\alpha)$ is the same as before, with value 0.

Each substrategy α is responsible for selecting a triple $(x_\alpha, y_\alpha, z_\alpha)$, or (x, y, z) for short, if clear from the context. Once such a triple is selected, α will check whether all the nodes in T_e in the interval (x_α, z_α), become dead, or not. If yes, then y_α will be put into C, which provides a global win for τ. If not, then all the substrategies will have their triples (x, y, z), and T_e will have infinite branches in each such intervals. Therefore, all these substrategies work together to make sure that $[S_e]$ is a subclass of $[T_e]$, witnessing that $[T_e]$ is not thin.

α has outcomes:

$$(x, w) < (y, w) < (z, w) < (ter),$$

where $(x, w), (y, w), (z, w)$ denote the outcomes that the construction waits at Steps (1), (2) and (4) respectively, i.e. the corresponding computation does not converge. Outcome (ter) denotes the outcome that this substrategy waits at Step (6) forever, i.e. this α-substrategy successfully finds an interval containing an infinite path of T_e.

Note that if α reaches Step (7), i.e. y is put into C, then this action provides a global win for τ, and τ will have outcome gw. Under this outcome, as either $C(y) \neq \Psi_e(\Phi_e(C))(y)$, or $\Phi_e(C)$ is not on $[T_e]$, the construction of subtree S_e is fully stopped, as there is no need to have $[S_e]$ to witness that $[T_e]$ is not a thin Π^0_1 class.

A *plus*-substrategy, i.e. when i even, proceeds in a similar way, with the change of (5) to the following (5′).

(5′) All the nodes on T_e in the region $(x_{e,i}, z_{e,i})$ are copied on S_e.

A *plus*-substrategy has the same outcomes as a *minus*-substrategy, except the outcome (ter), as *plus*-substrategies do not terminate nodes. It is the difference between (5) in *minus*-substrategies and (5′) in *plus*-substrategies.

The tree S_e is constructed by an $\mathcal{R}_e$-strategy τ, together with its substrategies, α say. Below τ's outcome gw, or α's outcomes $(x, w), (y, w), (z, w)$, the construction of S_e is inactive, as $\mathcal{R}_e$ is satisfied under these outcomes. That is, the construction of S_e needs to be continued only when each substrategy of τ has outcome (*ter*), which means that τ_1 is active under this outcome.

Consistency among strategies. We now consider interactions among strategies. First note that the interactions between substrategies of one requirement $\mathcal{R}_e$ are trivial, as once a substrategy enumerates a y into C, then $\mathcal{R}_e$ will get a global win and we will not continue the construction of the subtree S_e (it is not needed anymore).

We first consider the interactions between two $\mathcal{R}$-strategies, $\mathcal{R}_{e_1}$-strategy, τ_1, and $\mathcal{R}_{e_2}$-strategy, τ_2, say, and τ_1 has priority higher than τ_2. That is, τ_2 is below τ_1's outcome 1. A trivial case is when a substrategy α_1 of τ_1 enumerates a number into C first. Then, from now onwards, τ_1 will have outcome gw, and τ_2 will not be visited again in the remainder of the construction.

Our main concern arises when some substrategy of τ_2 puts a number into C first. We assume that τ_1 has priority higher than τ_2, and α_2 has (local) priority higher than α_1, and α_i is a substrategy of τ_i, for $i = 1, 2$, respectively. When α_2 say, enumerates a number, y say, into C, τ_2 is satisfied. However, this enumeration can injure τ_1's construction of S_{e_1}, in the following way:

- α_1 terminates all nodes in the region $(x_{e_1,i}, z_{e_1,i})$, at a stage s_1 say.
- At a stage $s_2 > s_1$, α_2 enumerates y_{α_2} into C, it can make $\Phi_{e_1}(C)$ to enter the region $(x_{\alpha_1}, z_{\alpha_1})$. *Thus, as $\Phi_{e_1}(C)$ comes back to $(x_{\alpha_1}, z_{\alpha_1})$, some nodes in this region should come back to be alive. This would make S_{e_1} not computable.*

To avoid this, we require that when α_2 selects its y_{α_2}, this selected number should be confirmed by τ_1 first. That is, after α_2 selects x_{α_2}, and sees that $\ell(\alpha_2)$ exceeds x_{α_2}, α_2 is ready to select y_{α_2}, and it first waits for a *plus*-substrategy of τ_1, α^* say, to select its $(x_{\alpha^*}, y_{\alpha^*}, z_{\alpha^*})$, just as the one given in the basic strategy above. After $\ell(\alpha^*)$ exceeds z_{α^*}, we let α_2 to take y_{α^*} as y_{α_2}, and follow the basic strategy described above. α^*, as a *plus*-substrategy, has outcomes $(x, w), (y, w), (z, w)$, and if one of them is true, then α^* shows that $\ell(\alpha^*)$ is bounded, giving a global win of τ_1. That is, under these outcomes, there is no construction of subtree S_{τ_1}.

Note that α^* is only responsible for the selection of $(x_{\alpha^*}, y_{\alpha^*}, z_{\alpha^*})$, and provides number y_{α^*} for α_2 as y_{α_2}. In the meanwhile, α^* is not responsible for securing an infinite path of $[T_e]$. In this sense, α^* has a big difference from a standard plus-strategy. Without loss of generality, we assume $\ell(\alpha^*)$ exceeds z_{α^*}, so α_2 will take y_{α^*} as y_{α_2}. Now, if α_2 enumerates y_{α_2} into C, then this is a global win for τ_2. Can this enumeration affect the construction of tree S_{τ_1}? In particular, can the enumeration of y_{α_2} bring $\Phi_{e_1}(C)$ back to the region $(x_{\alpha_1}, z_{\alpha_1})$? Here, we assume that α_1 reaches Step (6) and the nodes in the region $(x_{\alpha_1}, z_{\alpha_1})$ are terminated. We can also assume that α_1 is below α_2's outcome (*ter*) (otherwise, y_{α_2} is large enough and its enumeration into C could not bring $\Phi_{e_1}(C)$ back to the region $(x_{\alpha_1}, z_{\alpha_1})$). This means, x_{α_1} is selected bigger than z_{α^*}, and after y_{α_2} is enumerated into C causing $\Phi_{e_1}(C)$ to return to the region $(x_{\alpha_1}, z_{\alpha_1})$, $\Phi_{e_1}(C)$ already recovers up to

z_{α^*} and hence, $\Psi_{e_1}(\Phi_{e_1}(C))(y_{\alpha^*})$ equals to 0, giving

$$C(y_{\alpha^*}) = 1 \neq 0 = \Psi_{e_1}(\Phi_{e_1}(C))(y_{\alpha^*}),$$

a global win for τ_1.

We call this process of selecting y_{α_2} as a *confirmation gadget*. The idea is that when y_{α_2} is put into C, and this enumeration causes trouble for the construction of tree S_{τ_1}, this enumeration should also provide a global win for τ_1. This idea of confirming y_{α_2} can be iterated, when more $\mathcal{R}$-strategies are involved. Assume that $\tau_1, \dots, \tau_n$ are $\mathcal{R}$-strategies, in descending order of priority, and that α is a substrategy of τ_n. When α selects a number y, y needs to be confirmed by all these τ-strategies. What we do is to introduce an outcome, $(\alpha, y$ needed), with priority between outcomes (x, w) and (y, w). Below this outcome, we will have for each $j = n, \dots, 1$, one *plus*-substrategy α_j of τ_j, in a nested pattern. After α starts to look for y, these α_j's work as follows, starting with $j = n$, till α_n has a confirmed y and hands it over to α:

(1) Select x_j big, and wait for $\Phi_j(C)(x_j)$ to converge, with use $\varphi_j(x_j)$.
(2) Take outcome (y needed) and let α_{j-1} to act. [α_1 works in exactly the same way as the basic module.]
(3) Assume that α_j receives y from α_{j-1}, which has already been confirmed by α_{j-1}. Wait for $\Psi_j(\Phi_j(C))(y)$ to converge to 0, and that $\Phi_j(C)$ to converge on all numbers $\leq \psi_j(y)$.
(4) Select z_j as a number bigger than $\psi_j(y)$.
(5) Wait for $\Phi_j(C)(z_j)$ to converge.
(6) Declare that y is confirmed by τ_j and hands it over to α_{j+1} if $j < n$, and to α if $j = n$.

The *confirmation gadget* can be represented on the construction tree, with each α_j having its outcomes to indicate whether $\ell(\tau_j, s)$ exceeds x_j, or y, or z_j, or not. If α enumerates y into C at a later stage, then the enumeration of y will not cause trouble to the constructions of the corresponding trees S_{τ_j}, i.e. to cause $\Phi_{\tau_j}(BC)$ to be in $[T_{\tau_j}]\backslash[S_{\tau_j}]$, as otherwise, a disagreement between C and $\Psi_{\tau_j}(\Phi_{\tau_j}(C))$ at y would be created, and preserved.

We comment here that the idea of selecting y is a bit different from the one given in CDJS's paper [4]. Here, the set-up above of a confirmation gadget makes the confirmation more direct and clear.

Before we process to the density argument, we need to clarify a point that a substrategy α can be initialized, but the region being terminated by α is still valid in the remainder of the construction. It can happen that after α is initialized, some small number, y' say, being selected by a substrategy, α' say (with higher priority, of course), is put into C. Let τ and τ' be the mother nodes of α and α', respectively. The enumeration of y' can lead Φ_τ to enter a terminated region, a situation we always want to avoid. If τ' has priority higher than τ, then the enumeration of y' provides a global win for τ', and hence τ cannot be visited again, and we do not need to worry about whether $\Phi_\tau(C)$ enters a terminated region on S_τ or not. If τ has higher priority, then when α' select its y', y' should have been confirmed by τ, and when α' puts y' into C, as discussed above, either a disagreement at y' is created, and τ has a global win, or $\Phi_\tau(C)$ cannot come back to previous values up to x_α, and hence $\Phi_\tau(C)$ cannot come back to the region (x_α, z_α), which is terminated by α.

This point is obvious in CDJS's construction, as in CDJS's paper, they only need to construct one C, and the interactions among the $\mathcal{R}$-strategies are not complicated, as once a global win is achieved, it will be a win forever. However, in our density construction, it becomes a main concern, as a global win is not permanent anymore, due to the change of B. This means that after a substrategy is initialized, we need to make sure that the associated $\Phi_\tau(BC)$ cannot come back to the corresponding interval being terminated by this substrategy. In our construction, we will introduce a number for each substrategy (actually for each cycle, as we will see soon), and this number will be called "*a savior*" to make sure that once this substrategy is initiated, we can enumerate this savior number into C such that if later $\Phi_\tau(BC)$ comes back to this terminated region, either we can have a B-change, making a difference between B and $\Theta_\tau(\Phi_\tau(BC))$, or C and $\Phi_\tau(\Phi_\tau(BC))$ differ at this savior number. We will see more details in the next section.

3. On the Density of NONTHIN Degrees

We now prove that the NONTHIN degrees are dense in the c.e. degrees. Fix B and A with $B <_T A$. We will construct a c.e. set $C \le_T A \oplus B$ such that the degree of $B \oplus C$ does not contain any member of thin Π^0_1 classes. $C \le_T A \oplus B$ is a global requirement, and we will construct a p.r. functional Γ such that $\Gamma(AB)$ is total and equal to C. Here, we use $\Gamma(AB)$ for $\Gamma(A \oplus B)$. This also applies to other functionals with oracles.

Besides this, C will also satisfy the following requirements:

$\mathcal{R}_e$: if $\Phi_e(BC)$, $\Psi_e(\Phi_e(BC))$ and $\Theta_e(\Phi_e(BC))$ are all total, then either:

(a) $C \neq \Psi_e(\Phi_e(BC))$; or
(b) $B \neq \Theta_e(\Phi_e(BC))$; or
(c) $\Phi_e(BC)$ is not in $[T_e]$; or
(d) $[T_e]$ is not thin,

where $\{\langle \Phi_e, \Psi_e, \Theta_e, T_e\rangle : e \in \omega\}$ is an effective enumeration of partial computable functionals Φ, Ψ, Θ and primitive recursive trees T.

To satisfy one $\mathcal{R}_e$, we will construct a computable subtree S_e of T_e such that if $\Phi_e(BC)$, $\Psi_e(\Phi_e(BC))$, and $\Theta_e(\Phi_e(BC))$ are all total, with $C = \Psi_e(\Phi_e(BC))$, $B = \Theta_e(\Phi_e(BC))$, and $\Phi_e(BC)$ a branch in $[T_e]$, then $[S_e]$, as a subclass of $[T_e]$, witnesses that $[T_e]$ is not thin. The following subrequirements will be all satisfied.

$\mathcal{S}_{e,i}$: There exists a region $(x_{e,i}, z_{e,i})$ such that $[T_e]$ contains a branch in this interval.

- If i is even, then all nodes in T_e in region $(x_{e,i}, z_{e,i})$ will be *put* on S_e.
- If i is odd, then all nodes in interval $(x_{e,i}, z_{e,i})$ will be *terminated* on S_e.

Notation. Let α be an $\mathcal{S}_{e,i}$-strategy, and τ be the mother node of α. As mentioned earlier, if i is odd, α is a minus-substrategy, denoted as α^-, and if i is even, α is a plus-substrategy, denoted as α^+.

The basic module of satisfying one $\mathcal{S}$-strategy is basically the same as the one described above in Sec. 2, but with necessary modifications to deal with A-permissions and also the coding of B.

As before, suppose that the nodes in a region (x, z) are terminated, and we want to make sure that $\Phi_e(BC)$ will not come back to this region. Now B can change below a small number and can cause $\Phi_e(BC)$ to enter this region, which means that $\Phi_e(BC) \in [T_e]\backslash[S_e]$, and we fail to meet the requirement $\mathcal{S}_{e,i}$. Our idea is to use such B-changes as permissions to put an even smaller number into C to force $\Phi_e(BC)$ to enter a region of $[T_e]\cap[S_e]$ already prepared. That is, we select a number v first, and call this v a *savior number*, wait for $\Phi_e(BC)(v)$ to converge to 0. Then select an interval (x, z) (as in CDJS's construction). Suppose that after the nodes in the region (x, z) are terminated, and B changes, *which could lead $\Phi_e(BC)$ to come back into region (x, z)*, we enumerate this savior number v into C, so that if $\Phi_e(BC)$ recovers up to x, we will then have

$$\Psi_e(\Phi_e(BC))(v) = 0 \neq 1 = C(v).$$

As discussed in Sec. 2, we need to ensure the consistency between the enumeration of v and other $\mathcal{R}$-strategies, i.e. this v needs to be confirmed by all of these $\mathcal{R}$-strategies with higher priority. As C is constructed to be reducible to $A \oplus B$, the enumeration of v into C needs to be permitted by $A \oplus B$:

- An A-permission at n occurs, which is needed to terminate the associated region (x, z).

  [This A-change also undefines $\Gamma(AB)(v)$, and when we redefine $\Gamma(AB)(v)$, we define it with the B-part bigger than $\varphi_e(\psi_e(z))$, and hence when B changes below $\varphi_e(\psi_e(z))$, we are permitted to enumerate v into C.] This enumeration of v prevents $\Phi_e(BC)$ from entering the region (x, z), as otherwise, $\Phi_e(BC)$ covers up to x, and x is bigger than $\psi_e(v)$, and we then have

$$\Psi_e(\Phi_e(BC))(v) = 0 \neq 1 = C(v).$$

In the construction, α^- may take many tries to satisfy $\mathcal{S}_{e,i}$, and it can happen that all these tries fail because of the lack of A-permissions or further changes of B, and what we do is to use these changes to reduce A to B, threatening the assumption that $B <_T A$. α^- will construct two partial computable functionals Λ and Δ, to show that if α^- could not satisfy $\mathcal{S}_{e,i}$, then one of Λ^B and Δ^B would be total and equal to A, which is impossible.

α^- proceeds via cycles $(m, n) \in \omega \times \omega$, with cycle $(0, 0)$ starts first.

Cycle (m,n):

(1) Select $u_{m,n}$, denoted by u, for simplicity.
(2) Wait for $\Phi_e(BC)(u)$ to converge.
(3) Run the confirmation gadget to select $v_{m,n}$ big, denoted by v.
v is selected with the purpose of preventing the construction of tree S_e from returning to a terminated region (x,z) [Step (13)].

Define $\Gamma(AB)(v) = C(v) = 0$ with the A-part of the use equal to $n+1$, and the B-part of the use be a number large enough. [*This use will be kept the same, up to Step* (13), *even though A or B changes below the corresponding parts.*]

If Step (13) never happens, then $C(v) = 0$ and $\Gamma(AB)(v)$ will be defined forever, after a certain stage.

If Step (13) happens, i.e. n enters A, then $\Gamma(AB)(v)$ is undefined, and we can redefine it with $\gamma(v)$ anew. In particular, the B-part of the use $\gamma(v)$ is defined bigger than $\psi_e(\varphi_e(v))$. If B-changes on this (which could lead $\Phi_e(BC)$ to return to the region (x,z)), then $\Delta_m(B)(n)$ and $\Gamma(AB)(v)$ are both undefined and we enumerate v into C. Thus, if $\Phi_e(BC)$ returns to the interval (x,z), then we will have

$$\Psi_e(\Phi_e(BC))(v) = 0 \neq 1 = C(v).$$

(4) Wait for $\Psi_e(\Phi_e(BC))(v)$ to converge to 0, with use $\psi_e(v)$, and $\Phi_e(BC)(\psi_e(v))$ and $\Theta_e(\Phi_e(BC))(\varphi_e(\psi_e(v)))$ to converge.
[*We can ensure that when $\Phi_e(BC)$ comes back to a terminated region (x,z), then B, up to $\varphi_e(\psi_e(v))$, will not have any change, as $\theta_e(\varphi_e(\psi_e(v))) < x$ (otherwise, $\Theta_e(\Phi_e(BC))$ and B will differ at some point below $\varphi_e(\psi_e(v))$).*]
(5) Select $x_{e,i,m,n}$, denoted by x, as a big number. (*Especially, x is bigger than all the uses in the computations in* (4).)
(6) Wait for $\Phi_e(BC)(x)$ to converge.
[*When B has changes up to the use $\varphi_e(x)$, return to* (6). *It will change the selection of $y_{e,i,m,n}$ and $z_{e,i,m,n}$.*]
(7) Run the confirmation gadget to select $y_{e,i,m,n}$, denoted by y, bigger than $\varphi_e(x)$.
Define $\Gamma(AB)(y) = C(y) = 0$ with the A-part of the use equals to $m+1$ and the B-part of the use be a big number. This use will be kept the same, up to (15), *even though A or B changes below the corresponding part of the use. So if Step* (15) *never happens, then $C(y) = 0$ and $\Gamma(AB)(y)$ will be defined forever, after a certain stage. If Step* (15) *happens, then $\Gamma(AB)(y)$ becomes undefined, as the A-part of the use $\gamma(y)$ changes.*
(8) Wait for $\Psi_e(\Phi_e(BC))(y)$ to converge to 0, and that $\Phi_e(BC)$ to converge on all numbers $\leq \psi_e(y)$.
When B changes below the use $\varphi_e(\psi_e(y))$, return to (8). *It will change the selection of $z_{e,i,m,n}$. Of course, if B changes below $\varphi_e(x)$, return to* (6). *We will not mention this later, if it is clear from the context.*

(9) Select $z_{e,i,m,n}$, denoted by z, as a number bigger than $\psi_e(y)$.
[So far, we have selected a triple (x, y, z).]

(10) Wait for $\Phi_e(BC)(z)$ to converge.

(11) Define $\Delta_m^B(n) = A(n)$ with use $\delta_m(n)$ big.

(12) Wait for n to enter A, and simultaneously, start cycle $(m, n + 1)$.

(13) Declare that all the nodes in the region (x, z) are terminated in S_e.

[The nodes in this region are terminated only when A has a change at n. As mentioned above for the definition of Γ, this change of $A(n)$ undefines $\Gamma(AB)(v)$, and when we redefine $\Gamma(AB)(v)$ later, we will let the B-part of the use $\gamma(v)$ be a big number. Also we can define the A-part of the use bigger than m, and if m enters A later, we can still use this A-change as a permission to enumerate v into C.]

(14) Wait for a B-change below $\delta_m(n)$, and simultaneously, wait for a stage t such that all nodes in T_e in the region (x, z) become dead.

[If B changes below $\delta_m(n)$ later, we will redefine $\Delta_m(B)(n) = 1 = A(n)$, and start cycle $(m, n + 1)$.]

If B does not change below $\delta_m(n)$ and there is no such a stage t, then $[T_e]$ contains branches in the region (x, z), and $[S_e]$ will not contain these branches.

Otherwise, i.e. before we see a B-change below $\delta_m(n)$, we see that all nodes in T_e in the region (x, z) become dead, we define $\Lambda(B)(m) = A(m)$ with use $\lambda(m)$ big. This B-change also allows us to enumerate v into C, to prevent $\Phi_e(BC)$ from returning to the interval (x, z).

Wait for m to enter A, and simultaneously, start cycle $(m + 1, 0)$.

(15) Put y into C.

(*Note that the enumeration of y is permitted by the change of A at m. Also note that the change of $A(m)$ undefines $\Gamma(AB)(v)$, as the A-part of the use $\gamma(v)$ is defined bigger than m (after n enters A), and this $A(m)$-change allows us to enumerate v into C.*)

(16) Wait for $\Phi_e(BC)(z)$ to converge again, and simultaneously, wait for B to change below $\lambda(m)$.

If B changes below the use $\lambda(m)$, then, $\Lambda(B)(m)$ is rectified, and v is enumerated into C, to ensure that $\Phi_e(BC)$ will not come back to a terminated region (x, z). Cycle $(m + 1, 0)$ is started.

Otherwise, we will have a global win for $\mathcal{R}_e$ via one of the following ways:

(a) $\Phi_e(BC)(z)$ does not converge later. Then $\Phi_e(BC)$ is not total.

(b) $\Phi_e(BC)(z)$ converges later, but there is some $w \in (x, z)$ such that the value of $\Phi_e(BC)(w)$ is different from the previous one. Then $\Phi_e(BC)$ *cannot be a branch in $[T_e]$.*

(c) $\Phi_e(BC)(z)$ converges, but the values of $\Phi_e(BC) \upharpoonright z$ are the same as previous ones, then $C(y) = 1 \neq 0 = \Psi_e(\Phi_e(BC))(y)$.

Confirmation gadgets. Now assume that a minus-substrategy α^- is running a confirmation gadget to look for a confirmed number v or y at Step (3) or Step (7) respectively, and in a nested manner for $\alpha_n, \dots, \alpha_1$, where α_j is a substrategy of τ_j in the gadget, and τ_{j_1} has priority higher than τ_{j_2} if $j_1 < j_2$. α_j runs as follows:

(1) Select x_{α_j}, denoted as x, as a big number.
(2) Wait for $\Phi_{\tau_j}(BC)(x)$ to converge.
If the corresponding length of agreement does not exceed x, then take outcome $(x\tau_j, k, f)$, if k is the current length of agreement of τ_j.
When B has changes up to the use $\varphi_{\tau_j}(x)$, return to (2). It will change the selection of y and z_{α_j}.
(3) If $j > 1$, then run Step (1) of α_{j-1}.
When B changes below the use $\varphi_{\tau_j}(\psi_{\tau_j}(x))$, return to α_j by taking outcome $(x\tau_j, k, d)$, where k is the current length of agreement of τ_j. It will change the selection of y.
If $j = 1$, then just select y as a big number, and wait for $\Phi_{\tau_1}(BC)(y)$ to converge and for $\Phi_{\tau_1}(BC)$ to converge on all numbers $\leq \psi_{\tau_1}(y)$.
If the corresponding length of agreement does not exceed y, then take outcome $(y\tau_j, k, f)$, if k is the current length of agreement of τ_1.
When the length of agreement exceeds y, run Step (4) of α_2.
(4) After receiving y from α_{j-1}, wait for $\Psi_{\tau_j}(\Phi_{\tau_j}(BC))(y)$ to converge to 0, and that $\Phi_{\tau_j}(BC)$ to converge on all numbers $\leq \psi_{\tau_j}(y)$.
When B changes below the use $\varphi_{\tau_j}(\psi_{\tau_j}(y))$, return to (4). It will change the selection of z_{τ_j}.
(5) Select z_{τ_j}, denoted by z, as a number bigger than $\psi_{\tau_j}(y)$.
Thus, we have selected a triple (x, y, z).
(6) Wait for $\Phi_{\tau_j}(BC)(z)$ to converge, and then run Step (4) of α_{j+1}, if $j < n$.
When B changes below the use $\varphi_{\tau_{\alpha_j}}(\psi_{\tau_{\alpha_j}}(z_{\alpha_j}))$, return to a Step (6) of α_j by taking outcome $(z\tau_j, k, d)$, where k is the current length of agreement of τ_j.
If $j = n$, then we declare that y is confirmed by $\tau_n, \dots, \tau_1$. That is, cycle (m, n) will take this confirmed y and continue.

Note that the strategies in a confirmation gadget do not run any cycle and never has intention to enumerate numbers into C. These strategies are actually different from the plus-substrategies, as they are not supposed to find infinite paths in the corresponding Π^0_1 class. Their only job is to look for a suitable y for α. The outcomes of such a confirmation gadget are called gadget outcomes. *Confirmation strategies will not be counted as plus-substrategies.*

We answer two questions related to the work of these substrategies in a confirmation gadget: can α eventually get the wanted v or y, and after such a number is selected and confirmed, will it be the last one that α takes? In CDJS's construction, both questions have positive answers, as the mother node is assumed to have infinitary outcome. In our density construction, as B is coded, there is no guarantee for a positive answer to each question. We elaborate this point by using y as an example, as the case for v is the same as that of y.

Cycle (m, n) can proceed further to Step (8) only after y is confirmed and selected. It can happen that B can change below some computation involved in the the confirmation gadget, then α^- will take outcome $(y\tau_j, k, d)$, say. If we assume that α^- take outcome $(y\tau_j, k, d)$ infinitely many times, then α^- could not proceed as in the description of α^-. However, in this case, we have a global win for τ_j and τ_j becomes inactive below divergence outcome $(y\tau_j, k, d)$. As in the standard $0'''$ arguments, α^- is injured at this outcome, and below the divergence outcome $(y\tau_j, k, d)$, we will reallocate this $\mathcal{S}$-subrequirement again, and the confirmation gadget of this new $\mathcal{S}$-strategy will not need to deal with τ_j. It is easy to show that the $\mathcal{S}$-subrequirement can be injured in this manner at most finitely many times, and hence, eventually, there will be a $\mathcal{S}$-strategy that can find the wanted v and y and proceed as described in the module above.

Summary: Cycle (m, n) has the following three categories of outcomes. We omit (m, n), as it is clear and these are outcomes of cycle (m, n).

- Standard density outcomes

$$(k, d) < (k, f),$$

where $k \in \omega$ and for $k_1 < k_2$, we always have for any $k_1 < k_2$,

$$(k_1, *) < (k_2, \dagger).$$

- The termination outcome (*ter*), which has the lowest priority among the outcomes of cycle (m, n). Under this outcome:
  - — Δ_m^B does not compute $A(n)$ correctly (A-permission at n occurs), and all nodes in the region (x, z) are declared to be terminated on S_e.
    So if α^- has (*ter*) as the true outcome, then the region (x, z) contains an infinite path in $[T_e]\backslash[S_e]$, and α^- succeeds in finding such a region.
  - — If later, B changes below $\delta_m(n)$, we can redefine $\Delta_m^B(n)$ as 1, which equals to $A(n)$, forever. This B-change also allows the enumeration of v into C, which shows that if later, $\Phi_e(BC)$ comes into the region (x, z) later, then a disagreement between C and $\Psi_e(\Phi_e(BC))$ at v will occur, which is a global win of τ, the mother node of α^-.
  - — For τ, α's success of finding an infinite path is a *local win*. So below outcome (*ter*), we need to arrange other substrategies.
  - — If later, we find that this region in $[T_e]$ does not contain infinite path, then $\Lambda(B)(m)$ will be defined, and a new cycle $(m + 1, 0)$ will be started. As a consequence, from this stage onwards, outcome (*ter*) can never be true again in the remainder of the construction.
- Gadget outcomes $(x\tau_i, k, d)$, $(x\tau_i, k, f)$, $(y\tau_i, k, d)$, $(y\tau_i, k, f)$, $(z\tau_i, k, d)$, $(z\tau_i, k, f)$.

These outcomes are kind of standard density outcomes, and these outcomes will provide global wins of the corresponding τ_i by either capturing a divergence point of τ_i or showing that the corresponding length of agreement is finite. We assume that $\tau_1, \tau_2, \dots, \tau_\ell$ are active at α^-, and we will use the selection of y to illustrate the main features of these gadget outcomes. That is, when cycle (m, n) selects y at Step (7). [The selection of v will have exactly the same feature of outcomes.] Remember that it can happen that we cannot get the desired y, and we need to figure out that in the confirmation gadget, which τ_i has the length of agreement not long enough, which makes the selection of y fail.

The gadget first selects x_ℓ at stage s_0, and then waits for τ_ℓ's length of agreement to exceed x_ℓ [it will then select $x_{\ell-1}$]. So, before the length of agreement, $l(\tau_\ell)$ exceeds x_ℓ, cycle (m, n) will have outcome $(x\tau_\ell, k, d)$, or $(x\tau_\ell, k, f)$, where $k \leq x_\ell$ is the current length of agreement of τ_ℓ, as in standard density argument. After $l(\tau_\ell)$ exceeds x_ℓ at stage s_1 say, it selects $x_{\ell-1}$. [If the $l(\tau_\ell)$ becomes less than x_ℓ, then it will have outcome $(x\tau_\ell, k, d)$ consequently, and $x_{\ell-1}$ will be cancelled.] Wait for the length of agreement, $l(\tau_{\ell-1})$ to exceed $x_{\ell-1}$, and in this period, cycle (m, n) will have outcome $(x\tau_{\ell-1}, k, d)$, or $(x\tau_{\ell-1}, k, f)$, where $k \leq x_{\ell-1}$ is the current length of agreement of $\tau_{\ell-1}$. This process iterates, till τ_1 selects x_1. So before the length of agreement $l(\tau_1)$ exceeds x_1, cycle (m, n) will have outcome $(x\tau_1, k, d)$, or $(x\tau_1, k, f)$, where $k \leq x_1$ is the current length of agreement of τ_1.

Assume that $l(\tau_1)$ exceeds x_1 at stage s_ℓ. Then this gadget selects y as a big number, and waits for $l(\tau_1)$ exceeds y, i.e. $\Psi_{\tau_1}(\Phi_{\tau_1})(y)$ to converge to 0. [*Note*: If some outcome $(x\tau_i, k, d)$ is true again, then y, and also $x_{i-1}, \dots, x_1$, will be cancelled automatically.] Suppose that $l(\tau_1)$ exceeds y at stage s_y, then we will select z_1 big, and wait for $l(\tau_1)$ exceeds z_1 at stage $s_{\ell+1}$, say.

Before stage $s_{\ell+1}$, cycle (m, n) will have outcome $(y\tau_1, k, d)$, or $(y\tau_1, k, f)$, where $k \leq y$ is the current length of agreement of τ_1.

After stage s_y, we wait for $l(\tau_1)$ exceeds z_1, at stage $s_{\ell+1}$ say. Before stage $s_{\ell+1}$, cycle (m, n) will have outcome $(y\tau_1, k, d)$, or $(y\tau_1, k, f)$, where $k \leq y$ is the current length of agreement of τ_1. z_2 is selected at stage $s_{\ell+1}$, and we wait for $l(\tau_2)$ to exceed z_2. Iterate this process, until a stage $s_{2\ell}$ at which $l(\tau_\ell)$ exceeds z_ℓ. The confirmation of y is completed at stage z_ℓ.

The priority of these gadget outcomes is arranged based on the process above.

- x-outcomes:
  - for $i < j$, $(x\tau_j, *, *)$ has priority higher than $(x\tau_i, *, *)$,
  - for a fixed i, $(x\tau_i, a, *)$ has priority higher than $(x\tau_i, b, *)$, for $0 \leq a < b \leq x_i$,
  - for fixed i and a, $(x\tau_i, a, d)$ has priority higher than $(x\tau_i, a, f)$.

We define y-outcomes and z-outcomes in a similar way, and in z-outcomes, for $i < j$, $(z\tau_i, *, *)$ has priority higher than $(z\tau_j, *, *)$.

- x-outcomes have priority higher than y-outcomes and y-outcomes have priority higher than z-outcomes.
- If this gadget cannot select a wanted y, then one of these gadget outcomes is true, showing that the corresponding $\mathcal{R}$-strategy τ_i has a global win at α^-.

Now we list all outcomes of cycle (m, n) with priority indicated:

- We already have standard density outcomes (k, d) and (k, f) with order $(k, d) < (k, f)$. Also for $k_1 < k_2$, outcome $(k_1, *)$ always has priority higher than $(k_2, \dagger)$.
- The gadget outcomes appear between outcomes (k, f) and $(k + 1, d)$ (standard density outcomes), for each k, and correspondingly, we have an order described above between these outcomes.
- (ter) has the lowest priority.

Remark. It is true that between outcomes (k, f) and $(k+1, d)$, there are infinitely many such gadget outcomes. These outcomes depend on k, and we may even denote these outcomes by appending k in front of it. We will not do this, as it will be clear from the locations of these outcomes.

During the construction, we will have many times of wanting numbers, u, v, x, y, z, etc. For convenience, we leave a noticeboard "number is needed" in front of any divergent outcome, including both density outcomes and gadget outcomes. So when we want to select a number, x say, we actually see computations we want to preserve, up to k say, and what we do is to put x on the noticeboard in front of $(k + 1, d)$, and at the next step, we select x, and from now onwards, we will focus on the outcomes of computations of $j \leq x$, until the length of agreement exceeds x, when we put y on the noticeboard in front of the outcome $(x + 1, d)$. So this noticeboard cannot be the true outcome of cycle (m, n), and the function of it is to make sure that the corresponding computations are preserved and before we come to outer outcomes, we can make a selection of x, which means that we will focus on the computations up to x. We comment here that after a noticeboard is visited and provides a wanted number, if it is visited again at a later stage, then between these two stages, an outcome of the left of this noticeboard is true. This means that the noticeboard can never be true outcome.

Obviously, if one of the standard density outcomes is true, then $\mathcal{R}$ is satisfied, and if one of the gadget outcomes is true, then we will not have a confirmed y for cycle (m, n), but in this case, one of the $\mathcal{R}$-strategies with priority higher than the mother node of α^- is satisfied.

(ter) is the outcome showing that cycle (m, n) reaches (a permission of A at n is received) and stops at Step (13), then Δ_m^B has an error at n when computing $A(n)$, and on the other hand, all the nodes in the region (x, z) get terminated and this subrequirement $\mathcal{S}$ is satisfied. Note that the change of $A(n)$ undefines $\Gamma(AB)(v)$, and hence, if at a later stage, A changes, moving the construction to the left, or B changes, showing that $\Phi(BC)$ may come back to region (x, z), then these A-changes or B-changes undefine $\Gamma(AB)(v)$ again, and we are allowed to enumerate v into C, preventing $\Phi(BC)$ from going back to the region (x, z), as described before, when we formulate the purpose of setting the savior number v.

Another outcome of cycle (m, n) is when the cycle reaches Step (15), i.e. A has a change at m, and y is enumerated into C. We denote this outcome as gw, as the enumeration of y provides a "*global win*" for τ, till B changes on small numbers. Under outcome gw:

- Λ^B does not compute $A(m)$ correctly (A-permission at m occurs), and y is enumerated into C, and this enumeration will ensure that C and $\Psi_e(\Phi_e(BC))$ cannot have length of agreement longer than z, if B does not change below the use $\lambda(m)$.

  If $\Phi_e(BC)$ returns to the region (x, z) later, then as we have already seen at Step (13) that all nodes in T_e in this region are dead, $\Phi_e(BC)$ could not be a path in $[T_e]$, which means that we force $\Phi_e(BC)$ to be out of $[T_e]$ successfully, another global win of τ.

  If later, B changes below $\lambda(m)$, we can redefine $\Lambda^B(m)$ as 1, which equals to $A(m)$, forever. Again, this B-change allows us to enumerate v into C.
- gw is a global win of τ, and below this outcome, we do not need to arrange other substrategies of τ as under this outcome, either we will have a global win, or B has a change on small number, changing the values (and hence the computations) of $\Phi_e(BC)$ up to z. This B-change undefines $\Lambda^B(m)$, and starts a new cycle $(m + 1, 0)$. Because of this, after α^- enumerates y into C, we can create a link between τ and α, and a further τ-stage will be a $\tau^\frown gw$-stage, till B has a change.

  Thus, τ has two outcomes, 1 and gw, and τ takes outcome gw, only after a link between τ and one of its substrategy α is created. This is a Σ^B_1 fact.

A typical point of a density construction is the so-called *delayed permission*. Here we use v as an example, and the same idea also applies to the enumeration of number y. In the construction, it can happen that when A changes below m or n (i.e. an A-permission is given), α^- is not visited at this stage (i.e. the construction is currently working on the right of α^-), and when we define $\gamma(v)$ again, we define it as a big number, so that whenever the construction moves to the left, $\Gamma(AB)(v)$ will be undefined, and this will ensure that when α^- is visited later, $\Gamma(AB)(v)$ is (again) undefined and α^- can enumerate v into C. In a word, the enumeration of v can be far behind the permission of A at n, and the permission of AB is always there, if (m, n) is visited at a later stage.

Now come back to the outcomes of α, which consist of all possible outcomes of various cycles (m, n) with $m, n \in \omega$. The priority of these outcomes is given as follows:

- If $m_1 < m_2$, then outcomes of the form $(m_1, *, *, *)$ have higher priority than those outcomes of the form $(m_2, *, *, *)$.
- If $m_1 = m_2$, $n_1 < n_2$, then outcomes of the form $(m_1, n_1, *, *)$ have higher priority than those outcomes of the form $(m_2, n_2, *, *)$.
- If $m_1 = m_2$, $n_1 = n_2$, then outcomes of the form $(m_1, n_1, *, *)$, $(m_2, n_2, *, *)$ are just outcomes of cycle (m_1, n_1), and the priority of these outcomes follows from the one we provided before, for a single cycle.

We now consider the definition of the p.r. functional $\Delta_m(B)$, and we assume that $n_1 < n_2$. Note that when cycle (m, n_2) is started, cycle (m, n_1) already has its definition of $\Delta_m(B)(n_1)$. It can happen that after cycle (m, n_2) declares that its region is terminated (an A-permission at n_2 occurs), some $n_1 < n_2$ enters A. According to the basic strategy, cycle (m, n_1) declares that the region associated to cycle (m, n_1) is terminated. Here the change of $A(n_1)$ is an A-permission, allowing α^- to give up the region of cycle (m, n_2) (recall that the very basic target of α^- is to find a region containing an infinite branch of $[T_\tau]$), and the $A(n_1)$ change is a permission to enumerate the savior number v_{m,n_2} into C, to ensure that $\Phi_e(BC)$ will not come into the region being terminated by cycle (m, n_2). In this case, cycle (m, n_1) terminates the associated region and α^- will have outcome $(m, n_1, (ter))$.

Another reason for cycle (m, n_1) to act, after $\Delta_m(B)(n_2)$ is defined, is that B changes below $\delta_m(n_1)$, hence below $\delta_m(n_2)$. This B-change undefines $\Delta_m(B)(n_1)$, and also $\Delta_m(B)(n_2)$. This change of B also allows us to enumerate the savior number v_{m,n_2} into C, if cycle already has an A-permission to terminate the associated region.

One more situation is when n_1 enters A, cycle (m, n_2) defines $\Lambda(B)(m)$ and starts cycle $(m + 1, 0)$. If so, then cycle (m, n_1) will have no action (termination), until B changes, undefining $\Lambda(B)(m)$. If there is no such a B-change, then there will be no more cycles of the form $(m, -)$, and $\Delta_m(B)(n_1)$ does not compute $A(n_1)$ correctly. On the other hand, we will have either $\Lambda(B)(m) = A(m)$ or an enumeration of y into C, a global win for τ. If B changes, then this change will undefine $\Delta_m(B)(n_2)$ and $\Lambda(B)(m)$, and what we do is to terminate the region of cycle (m, n_1), which is a delayed termination. We can do so, as when n_1 enters A, $\Gamma(AB)(v_{m,n_1})$, and we define it later with big use, so when B-change occurs, $\Gamma(AB)(v_{m,n_1})$ is undefined again, which enables us to terminate this region (that is, if later, the construction moves to the left of cycle (m, n_1), we are able to enumerate v_{m,n_1} into C to ensure that $\Phi(BC)$ does not come into this region again). So after this B-change, we will continue to define $\Delta_m(B)$ (this definition was stopped when cycle (m, n_2) defines $\Lambda(B)(m)$). This explains that there is no competition among cycles $(m, -)$ for the definition of $\Lambda(B)(m)$.

As $B <_T A$, $\Delta_m(B)$ could not have definition on (almost) all n with $\Delta_m(B)(n) = A(n)$. This means that either α^- runs only finitely many cycles of the form $(m, -)$ in the whole construction (so only finitely many n can have $\Delta_m(B)(n)$ defined), or α runs infinitely many cycles, and one of these cycles, (m, n) say, runs infinitely often, corresponding to a divergence outcome. In this case, $\Delta_m(B)(n)$ is not to be defined.

We can apply a similar argument to show that $\Lambda(B)$ could not have definition on each m with $\Lambda(B)(m) = A(m)$.

Remark. We summarize how cycle (m, n) works consistently with the construction of Γ. For the number v being selected by cycle (m, n), when we define $\Gamma(AB)(v)$, we let the A-part of use $\gamma(v)$ be defined as a number bigger than n, and when Step

(13) is reached, $\Gamma(AB)(v)$ is undefined, and then, when we redefine it, we let the use $\gamma(v)$ even bigger, especially, the A-part of the use is bigger than m (so when the construction moves to left toward α^-, due to A-changes or B-changes, $\Gamma(AB)(v)$ is undefined, again and again, and if eventually, α^- is visited again, then $\Gamma(AB)(v)$ is undefined, allowing us to enumerate v into C), and the B-part of the use is bigger than $\delta_m(n)$. When B has a change below $\delta_m(n)$, and hence below $\gamma(v)$, we can enumerate v into C as wanted. The same is true for the enumeration of y.

When we move from left to right, we have certain computations confirmed, $\Phi(BC)(w)$ say, which, when we come to the right, the previous computation of $\Phi(BC)(w)$ is confirmed by computations $\Theta(\Phi(BC))$, i.e. B and $\Theta(\Phi(BC))$ agree on $\varphi(w)$. That is, when we move to the right, the first thing is to have each $\mathcal{S}$-strategy to have length agreement bigger than the B-part involved (i.e. to have B and $\Theta(\Phi(BC))$ agree on $\varphi(w)$) and in particular, when we choose a new y, y should be bigger than the θ-use $\theta(\varphi(w))$. Thus, if we move to the left again, due to B-changes below the use $\varphi(x)$, and this change actually prevents $\Phi(BC)$ from being back to a previous region being terminated before (up to y), as otherwise, we will have a disagreement between B and $\Theta(\Phi(BC))$ below $\varphi(w)$. Another reason of moving from right to left is because of A-permissions, which undefine $\Gamma(AB)(v)$ and $\Gamma(AB)(y)$, allowing us to enumerate some v or y into C as wanted.

We now describe how a plus-substrategy α^+ (i.e. an $\mathcal{S}_{e,j}$-substrategy, j even) works. Remember that an α^+-strategy only copies part of T_e on S_e. This is much simpler than a minus strategy. Most of the idea is the same as those described in α^--strategy.

α^+ proceeds via cycles (m), with $m \in \omega$, starting with cycle (0) starts first.

Cycle (m):

(1) Select $x_{\alpha,m}$, denoted by x, as a big number.

(2) Wait for $\Phi_e(BC)(x)$ to converge.
When B has changes up to the use $\varphi_e(x)$, return to (2). *It will change the selection of $y_{\alpha,m}$ and $z_{\alpha,m}$.*

(3) Run the confirmation gadget to select $y_{\alpha,m}$, denoted by y, bigger than $\varphi_e(x)$.
Define $\Gamma(AB)(y) = C(y) = 0$ with the use of A-part equals to $m+1$ and let the use of B-part be any big number.

(4) Wait for $\Psi_e(\Phi_e(BC))(y)$ to converge to 0, and for $\Phi_e(BC)$ to converge on all numbers $\leq \psi_e(y)$.
When B changes below the use $\varphi_e(\psi_e(y))$, return to (4). *It will change the selection of $z_{\alpha,m}$.*

(5) Select $z_{\alpha,m}$, denoted by z, as a number bigger than $\psi_e(y)$.
(*Thus, we have selected a triple* (x, y, z).)

(6) Wait for $\Phi_e(BC)(z)$ to converge.
(*Copy all the nodes on T_e in the region (x, z) on S_e, in the sense that at any stage, when we see new nodes on T_e, we also add these nodes on S_e.*)

(7) Wait for a stage t such that all nodes in T_e in the region (x, z) become dead.

(8) Define $\Xi(B)(m) = A(m)$ with use $\xi(m)$ big. Wait for m to enter A, and simultaneously, start cycle $(m+1)$.

(9) Put y into C.

(*The enumeration of y is permitted by the change of $A(m)$.*)

(10) Wait for $\Phi_e(BC)(z)$ to converge again, and simultaneously, wait for B to change below $\xi(m)$.

If B changes below the use $\xi(m)$, then $\Xi(B)(m)$ is rectified.

(11) Start cycle $(m+1, 0)$.

Otherwise, we will have a global win for τ via one of the following cases:

(a) $\Phi_e(BC)(z)$ does not converge later. Then $\Phi_e(BC)$ is not total.

(b) $\Phi_e(BC)(z)$ converges later, but there is some $w \in (x, z)$ such that the value of $\Phi_e(BC)(w)$ is different from the previous one. Then $\Phi_e(BC)$ *cannot be a branch in* $[T_e]$.

(c) $\Phi_e(BC)(z)$ converges, but the values of $\Phi_e(BC) \upharpoonright z$ are the same as previous ones, then $C(y) = 1 \neq 0 = \Psi_e(\Phi_e(BC))(y)$.

Cycle (m) has two kinds of outcomes: standard density outcomes and gadget outcomes, which are the same as those in α^--strategies. When cycle (m) reaches Step (9), we will have a global win outcome, *gw*. Under this outcome,

- $\Xi(B)$ does not compute $A(m)$ correctly, and y is enumerated into C, so that if $\Phi_e(BC)$ comes back to the same values below z later, then we will have a disagreement between C and $\Psi_e(\Phi_e(BC))$ at y. This will be a global win for τ, until B changes below $\xi(m)$, when we will redefine $\Xi^B(m)$ as 1, which equals to $A(m)$, forever.

We are now ready to give a full description of the construction of C.

4. Construction

We first define the construction tree T, which is defined by recursion. The priority of the requirements is defined as follows:

$$\mathcal{R}_0 < \mathcal{S}_{0,0} < \mathcal{R}_1 < \mathcal{S}_{0,1} < \mathcal{S}_{1,0} < \mathcal{S}_{1,1} < \mathcal{R}_2 < \cdots < \mathcal{R}_e < \cdots$$
$$< \mathcal{S}_{0,e} < \mathcal{S}_{1,e} < \cdots < \mathcal{S}_{e-1,e} < \mathcal{S}_{e,0} < \mathcal{S}_{e,1} < \cdots < \mathcal{S}_{e,e} < \mathcal{R}_{e+1} < \cdots.$$

We use τ to denote $\mathcal{R}$-strategies and α to denote $\mathcal{S}$-strategies.

The top node on T is labeled as $\mathcal{R}_0$. Assume that τ is a node on T, there are two edges leaving τ, 1 and *gw*.

Assume that α is a node on T. There are infinitely many edges leaving α, i.e. α has infinitely many outcomes. If α is a minus-strategy, then these outcomes fall into the following three categories, related to cycle (m, n):

- Termination outcome $(m, n, (ter))$, which is the rightmost among outcomes of cycle (m, n).

- Standard density outcomes $(m,n,k,d),(m,n,k,f)$, where $k\in\omega$, and these outcomes take lexicographic order.
- Gadget outcomes between $(m,n,k-1,f)$ and (m,n,k,d), where $k\in\omega$,

  x-outcomes: $(m,n,x\tau_j,p,d),(m,n,x\tau_j,p,f)$,
  y-outcomes: $(m,n,y\tau_j,p,d),(m,n,y\tau_j,p,f)$,
  z-outcomes: $(m,n,z\tau_j,p,d),(m,n,z\tau_j,p,f)$,

  where $p\in\omega$.

  — x-outcomes are on the left of y, z-outcomes.
  — Among x-outcomes, $x\tau_{j_1}$-outcomes are on the left of $x\tau_{j_2}$-outcomes, if $\tau_{j_2}\subset\tau_{j_1}$.
  — Among y, z-outcomes, for a particular τ, $y\tau$-outcomes are on the left of $z\tau$-outcomes, and among different τ's. $y\tau_{j_2},z\tau_{j_2}$-outcomes are on the left of $y\tau_{j_1},z\tau_{j_1}$-outcomes, if $\tau_{j_2}\subset\tau_{j_1}$.
  — Before each $(-,d)$-outcome, there is a "noticeboard", which can be used to indicate which number is needed.

If α is a plus-strategy, then the outcomes fall into the following three categories, related to cycle (m,n):

- Standard density outcomes $(m,k,d),(m,k,f)$, where $k\in\omega$, and these outcomes take lexicographic order.
- Gadget outcomes between $(m,k-1,f)$ and (m,k,d), where $k\in\omega$,

  x-outcomes: $(m,x\tau_j,p,d),(m,x\tau_j,p,f)$,
  y-outcomes: $(m,y\tau_j,p,d),(m,y\tau_j,p,f)$,
  z-outcomes: $(m,z\tau_j,p,d),(m,z\tau_j,p,f)$,

  where $p\in\omega$.

  — x-outcomes are on the left of y, z-outcomes.
  — Among x-outcomes, $x\tau_{j_1}$-outcomes are on the left of $x\tau_{j_2}$-outcomes, if $\tau_{j_2}\subset\tau_{j_1}$.
  — Among y, z-outcomes, for a particular τ, $y\tau$-outcomes are on the left of $z\tau$-outcomes, and among different τ's. $y\tau_{j_2},z\tau_{j_2}$-outcomes are on the left of $y\tau_{j_1},z\tau_{j_1}$-outcomes, if $\tau_{j_2}\subset\tau_{j_1}$.
  — Before each $(-,d)$-outcome, there is a "noticeboard", which can be used to indicate which number is needed.

$(-,k,d)$ is a divergence outcome (at k), and $(-,k,f)$ shows that k is the current length of agreement, and when α has these outcomes, the mother node τ actually has a global win, and the strategies between τ and α are injured, and as usual, we will have backup versions of these strategies.

Fix $\alpha\in T$. Let $L(\alpha)$ denote the list of requirements that are not satisfied at α:

- Let $L(\lambda)$ be the set of all requirements, where λ is the root of T. Assign $\mathcal{R}_0$ to λ, and say that $\mathcal{R}_0$ becomes active at λ.

- For any $\alpha \in T$ with $\alpha = \beta^\frown \mathcal{O}$, and assume that $L(\beta)$ is given. Recall that β is assigned with the requirement in $L(\beta)$ with highest priority, $\mathcal{P}$ say.

  If β is an $\mathcal{R}$-strategy, then $\mathcal{O}$ is either 1 or gw. If $\mathcal{O}$ is 1, then we say that $\mathcal{R}$ is active at β, and we let $L(\alpha) = L(\beta)\backslash\{\mathcal{R}_e\}$. If $\mathcal{O}$ is gw, then we let $L(\alpha) = L(\beta)\backslash\{$all substrategies of $\beta\}$.

  If β is an $\mathcal{S}$-strategy $\mathcal{S}_{e,i}$ say, then $\mathcal{O}$ has the following possibilities. Here we assume that β is a minus-substrategy, and we assume that β is running cycle (m, n). In case that it is a plus-substrategy, it has $(m, *, d)$ and $(m, *, f)$ instead, and it runs cycle (m).

  — $\mathcal{O}$ is $(m, n, (ter))$. Then let $L(\alpha) = L(\beta)\backslash\{\mathcal{S}_{e,i}\}$.

  — $\mathcal{O}$ is a standard density outcome (m, n, k, d), or (m, n, k, f). Then let

  $$L(\alpha) = L(\beta) \cup \{\mathcal{U} : \mathcal{U} \text{ is a requirement allocated between } \tau \text{ and } \beta\}.$$

  Here τ is the mother node of β. *All the requirements allocated between β and τ are injured at α and if $\mathcal{U}$ is such a requirement, and is an $\mathcal{R}$-requirement, then $\mathcal{U}$ becomes inactive at α.*

  — $\mathcal{O}$ is a gadget outcome. We assume that it is an x-outcome, $(m, n, x\tau_j, k, d)$, or $(m, n, x\tau_j, k, f)$, and the cases that it is a y-outcome, or z-outcome will be the same. Then let

  $L(\alpha) = L(\beta) \cup \{\mathcal{U} : \mathcal{U}$ is a requirement allocated between τ_j and $\beta\}$.

  All the requirements allocated between β and τ_j are injured at α and if $\mathcal{U}$ is such a requirement, and is an $\mathcal{R}$-requirement, then $\mathcal{U}$ becomes inactive at α.
- Let $\mathcal{Q}$ be the requirement in $L(\alpha)$ with the highest priority, and assign $\mathcal{Q}$ to α. If $\mathcal{Q}$ is an $\mathcal{R}_j$-requirement, then say that $\mathcal{R}_j$ becomes active at α.

This completes the construction of tree T. Note that the construction ensures that on each infinite path f of the tree T, each $\mathcal{R}$-requirement can be injured at most finitely many times, and that for each e, there is a longest node σ on f on which $\mathcal{R}_e$ is assigned, and no strategy below σ on f can injure σ. Thus, $\mathcal{R}_e$ becomes active at σ, and along f, it can happen that $\mathcal{R}_e$ is active at all nodes, or it becomes inactive at some node σ' say. In the former case, all the strategies, $\mathcal{S}_{e,i}$, can be injured at most finitely many times, and that for each i, there is a longest node σ'' on f on which $\mathcal{S}_{e,i}$ is assigned, and all of these substrategies along f have outcomes $(m, n, (ter))$. In the latter case, either σ' is just σ, with outcome gw, or σ' is a substrategy of σ, with outcome either (m, n, k, d), or (m, n, k, f), for some k, or σ' is a substrategy of some $\mathcal{R}$-strategy between σ and σ', with outcome either $(m, n, x\sigma, k, d)$, or $(m, n, x\sigma, k, f)$, for some k, or $(m, n, y\sigma, k, d)$, or $(m, n, y\sigma, k, f)$, or $(m, n, z\sigma, k, d)$, or $(m, n, z\sigma, k, f)$.

We now describe the full construction of C, together with a computable functional Γ such that $C = \Gamma(AB)$.

Construction of C and Γ. Let $\{a_s\}_{s\in\omega}$ and $\{b_s\}_{s\in\omega}$ be effective enumerations of A and B, respectively. The construction will proceed by stages, and at the end of each stage s, define δ_s as the current approximation of the true path, TP. A stage s is called an α-stage if $\alpha \subseteq \delta_s$.

Stage 0: Initialize all nodes on the construction tree T, and let $C = \emptyset$, and also $\Gamma = \emptyset$. Let δ_0 be λ, the root of T.

Stage $s > 0$: Stage s consists of three phrases:

- **Phase 1** decides δ_s, the approximation of the true path TP at stage s,
- **Phase 2** works on the construction of S_τ, where τ is an $\mathcal{R}$-strategy on δ_s,
- **Phase 3** extends the definition of Γ.

Phase 1: Definition of δ_s.

Substage 0: Let $\delta_s(0) = \lambda$, the root of T.

Substage $t \leq s$. Given $\zeta = \delta_s \upharpoonright t$. Initialize all nodes $>_L \zeta$.

If $t = s$, then define $\delta_s = \zeta$, initialize all nodes $>\zeta$ and go to stage $s + 1$.

Otherwise, for $t < s$, define $\delta_s(t)$ and we will take action for $\zeta = \delta_s(t)$ accordingly. [For those cycles or strategies on the right of $\delta_s(t)$, if the associated savior numbers have received A-permissions by stage s, and they are not in C yet, then enumerate these numbers into C.] Initialize all the nodes and cycles on the right of $\delta_s(t + 1)$.

- ζ is an $\mathcal{R}$-strategy.

If no link between ζ and any substrategy exists at the last ζ-stage, then let $\delta_s(t) = \zeta^\frown 1$.

If there is a link between ζ and some substrategy, and the link is still there, i.e. B has no changes on small numbers, and the disagreement created by β is still valid, then $\delta_s(t) = \zeta^\frown gw$.

If there is a link between ζ and a substrategy σ via cycle (m, n), and B-changes on small numbers, then we remove this link and enumerate the savior number of cycle (m, n) into C. Let $\delta_s = \sigma$.

- ζ is an $\mathcal{S}$-strategy.

Suppose that ζ is a minus substrategy.

If ζ has no cycles started, then start cycle $(0, 0)$ by selecting u big (u is put on the noticeboard on the left of outcome $(0, 0, 0, d)$). Define $\delta_s(t) = \zeta^\frown(0, 0, 0, b)$, and let it be δ_s.

Otherwise, suppose that ζ is running cycle (m, n) at stage s^-, the last stage at which ζ was visited [running cycle (m, n)], and assume that ζ has outcome $\mathcal{O}$ at stage s^-, if any.

(i) Check whether one of the following happens between stages s^- and s.

  (a) B-changes from stage s^- to stage s have changed some computations associated to a cycle on the left of cycle (m, n), or have changed some computations associated to an outcome of cycle (m, n) whose priority is not lower than outcome $\mathcal{O}$.

(b) A has changes (during the period from stage s^- to stage s), which are needed as permissions by some (least) cycle on the left of cycle (m, n), or by (m, n) itself.

If (a) happens, assume that (m', n') is the corresponding cycle, and define $\delta_s(t)$ as $\xi^\frown(m', n', k, d)$ or $\xi^\frown(m', n', p\tau', d, d)$, where $p \in \{x, y, z\}$ and τ' is an $\mathcal{R}$-strategy with $\tau'^\frown 1 \subset \tau$. Here we assume that τ is the mother note of ζ and τ' is active at ζ. Go to substage $t + 1$.

If (b) happens, assume that (m', n') is the corresponding cycle of substrategy ξ, and perform accordingly:

(b1) If n' enters A, then terminate the nodes in the associated region (x, z) on the tree $S_{\tau'}$ at stage s, where τ' is the mother node of ξ. Also request that the associated savior number v be enumerated into C, once cycle (m', n') is initiated. Let $\delta_s = \xi^\frown(m', n', (ter))$ and go to stage $s + 1$.

(b2) If m' enters A, then enumerate the corresponding y into C, and create a link between ξ and τ', and let $\delta_s = \tau'^\frown gw$. Go to stage $s + 1$.

If both (a) *and* (b) *appear to be true at stage* s, *then let the one with higher priority act.*

(ii) Suppose that neither (a) nor (b) appears to be true at stage s.

If $m \in A_s$, then cycle (m, n) has not requested A-permission at m yet (otherwise, (b) above applies), and we define $\Lambda(B)(m) = 1 = A_s(m)$ with use $\lambda(m) = \lambda(m - 1)$, and start cycle $(m + 1, 0)$.

If $n \in A_s$, then cycle (m, n) has not requested A-permission at n yet (otherwise, (b) above applies), and we define $\Delta_m(B)(n) = 1 = A_s(n)$ with use $\delta_m(n) = \delta_m(n - 1)$, and start cycle $(m, n + 1)$.

If none of the above is true, then check for cycle (m, n) which of the following applies:

(c0) If u is not defined, then define u as a big number and let $\delta_s = \zeta^\frown (m, n, 0, f)$.

(c1) If u is defined and $\ell(\tau, s) < u + 1$ [we take it as a divergence if a computation changes from stage s^-]. Let $k = \ell(\tau, s)$. If at k, the associated computation does not converge, then let $\delta_s(t) = \zeta^\frown(m, n, k, d)$. Otherwise, let $\delta_s(t) = \zeta^\frown(m, n, k, f)$. Go to substage $t + 1$.

(c2) $\ell(\tau, s) \geq u$, and v is not selected yet. Let $\delta_s = \zeta^\frown(m, n, u + 1, b)$, and indicate that v is needed.

(c3) ζ has outcome $(m, n, u + 1, b)$ at stage s^-, then start the confirmation gadget to find v.

(c3.1(n)) We first define $x\tau_n$ as a big number (here we assume that $\tau_n, \ldots, \tau_1$ are $\mathcal{R}$-strategies active at ζ, with ascending order of priority) and let $\delta_s = \zeta^\frown(m, n, u+1, f)$, and go to stage $s+1$.

(c3.2(n)) $x\tau_n$ is defined, and $\ell(\tau_n, s) < x\tau_n + 1$. Let $k = \ell(\tau_n, s)$. If at k, the associated computation does not converge, then let $\delta_s(t) = \zeta^\frown(m, n, x\tau_n, k, d)$, and go to substage $t+1$. Otherwise, if ζ has the outcome $(m, n, x\tau_n, k, f)$ at stage s^-, then let $\delta_s(t) = \zeta^\frown(m, n, x\tau_n, k, f)$ and go to substage $t+1$. If ζ has other outcome at stage s^-, then let $\delta_s = \zeta^\frown(m, n, x\tau_n, k, f)$, and go to stage $s+1$.

(c3.3(n)) $x\tau_n$ is defined, and $\ell(\tau_n, s) \geq x\tau_n + 1$. Let $\delta_s(t) = \zeta^\frown(m, n, x\tau_{n-1}, 0, b)$. [The confirmation gadget is switched to (c3.1($n-1$)). (c3.3(1)) has a session of choosing v.]

(c3.4(n)) After seeing $\ell(\tau_{n-1}, s) \geq z\tau_{n-1} + 1$, let $\delta_s = \zeta^\frown(m, n, z\tau_n, 0, b)$, and go to stage $s+1$.
[The confirmation gadget has been switched back from (c3.1($n-1$)).]

(c3.5(n)) Define $z\tau_n$ as a big number and let $\delta_s = \zeta^\frown(m, n, z\tau_n, 0, f)$, and go to stage $s+1$.

(c3.6(n)) $z\tau_n$ is defined, and $x\tau_n < \ell(\tau_n, s) \leq z\tau_n$. Let $k = \ell(\tau_n, s)$. If at k, the associated computation does not converge, then let $\delta_s(t) = \zeta^\frown(m, n, z\tau_n, k, d)$. Otherwise, let $\delta_s = \zeta^\frown(m, n, z\tau_n, k, f)$, and go to stage $s+1$.

(c3.7(n)) $z\tau_n$ is defined, and $\ell(\tau_n, s) > z\tau_n$. Then declare that the selected v is confirmed, and let $\delta_s = \zeta^\frown(m, n, u+1, f)$, and go to stage $s+1$.

(c4) v is defined and $\ell(\tau, s) < v+1$. Let $k = \ell(\tau, s)$. If at k, the associated computation does not converge, then let $\delta_s(t) = \zeta^\frown(m, n, k, d)$. Otherwise, let $\delta_s(t) = \zeta^\frown(m, n, k, f)$. Go to substage $t+1$.

(c5) v is selected and $\ell(\tau, s) \geq v+1$. Then let $\delta_s(t) = \zeta^\frown(m, n, v+1, b)$ and go to stage $s+1$.

(c6) ζ has outcome $(m, n, v+1, b)$ at stage s^-, choose x big, and let $\delta_s = \zeta^\frown(m, n, v+1, f)$. Go to stage $s+1$.

(c7) x is defined and $\ell(\tau, s) < x+1$. Let $k = \ell(\tau, s)$. If at k, the associated computation does not converge, then let $\delta_s(t) = \zeta^\frown(m, n, k, d)$. Otherwise, let $\delta_s(t) = \zeta^\frown(m, n, k, f)$. Go to substage $t+1$.

(c8) $\ell(\tau, s) \geq x$, and y is not selected yet. Let $\delta_s = \zeta^\frown(m, n, x+1, b)$, and indicate that y is needed.

(c9) ζ has outcome $(m, n, x+1, b)$ at stage s^-, then start the confirmation gadget to find y.

(c9.1(n)) We first define $x\tau_n$ as a big number (recall that we are assuming that $\tau_n, \ldots, \tau_1$ are $\mathcal{R}$-strategies active at ζ, with ascending order of priority) and let $\delta_s = \zeta^\frown(m, n, u+1, f)$, and go to stage $s+1$.

(c9.2(n)) $x\tau_n$ is defined, and $\ell(\tau_n, s) \leq x\tau_n$. Let $k = \ell(\tau_n, s)$. If at k, the associated computation does not converge, then let $\delta_s(t) = \zeta^\frown(m, n, x\tau_n, k, d)$. Otherwise, let $\delta_s(t) = \zeta^\frown(m, n, x\tau_n, k, f)$, and go to substage $t+1$.

(c9.3(n)) $x\tau_n$ is defined, and $\ell(\tau_n, s) > x\tau_n$. Let $\delta_s(t) = \zeta^\frown(m, n, x\tau_{n-1}, 0, b)$. [The confirmation gadget is switched to (c9.1($n-1$)). (c9.3(1)) has a session of choosing y.]

(c9.4(n)) After seeing that $\ell(\tau_{n-1}, s) > z\tau_{n-1}$, let $\delta_s = \zeta^\frown(m, n, z\tau_n, 0, b)$ and go to stage $s+1$.

(c9.5(n)) Define $z\tau_n$ as a big number and let $\delta_s = \zeta^\frown(m, n, z\tau_n, 0, f)$, and go to stage $s+1$.

(c9.6(n)) $z\tau_n$ is defined, and $x\tau_n < \ell(\tau_n, s) \leq z\tau_n$. Let $k = \ell(\tau_n, s)$. If at k, the associated computation does not converge, then let $\delta_s(t) = \zeta^\frown(m, n, z\tau_n, k, d)$. Otherwise, let $\delta_s(t) = \zeta^\frown(m, n, z\tau_n, k, f)$, and go to substage $t+1$.

(c9.7(n)) $z\tau_n$ is defined, and $\ell(\tau_n, s) > z\tau_n$. Then declare that the selected y is confirmed, and let $\delta_s = \zeta^\frown(m, n, x+1, f)$ and go to stage $s+1$.

(c10) y is defined and $\ell(\tau, s) \leq y$. Let $k = \ell(\tau, s)$. If at k, the associated computation does not converge, then let $\delta_s(t) = \zeta^\frown(m, n, k, d)$. Otherwise, let $\delta_s(t) = \zeta^\frown(m, n, k, f)$. Go to substage $t+1$.

(c11) y is defined and $\ell(\tau, s) > y$. Let $\delta_s = \zeta^\frown(m, n, y+1, b)$, and indicate that z is needed.

(c12) ζ has outcome $(m, n, y+1, b)$ at stage s^-, choose z big, and let $\delta_s = \zeta^\frown(m, n, y+1, f)$. Go to stage $s+1$.

(c13) z is defined and $\ell(\tau, s) < z+1$. Let $k = \ell(\tau, s)$. If at k, the associated computation does not converge, then let $\delta_s(t) = \zeta^\frown(m, n, k, d)$. Otherwise, let $\delta_s(t) = \zeta^\frown(m, n, k, f)$. Go to substage $t+1$.

(c14) z is selected and $\ell(\zeta, s) \geq z+1$. Define $\Delta_m(B)(n)[s] = A_s(n)$ with use $\delta_m(n) = s$, and start cycle $(m, n+1)$, by letting $\delta_s = \zeta^\frown(m, n+1, 0, b)$. Go to stage $s+1$.

(c15) $\Delta_m(B)(n)[s]$ is defined as 0 and n is in A_s. Then terminate all the nodes in the region (x, z), and let $\delta_s = \xi^\frown(m, n, ter)$.

(c16) If ζ has outcome $(m, n, (ter))$ at the last ζ-stage, and some nodes on T_τ in the region (x, z) are still alive at stage s, then let $\delta_s(t) = \zeta^\frown(m, n, (ter))$, and go to substage $t+1$.

(c17) If ζ has outcome $(m, n, (ter))$ at the last ζ-stage, $\Delta_m(B)(n)[s] = 0 \neq 1 = A_s(n)$, and all nodes on T_τ in the region (x, z) are dead (i.e. they have no further extensions on T_τ), then define $\Lambda(B)(m)[s] = A_s(m)$ with use $\lambda(m) = s$, and start cycle $(m+1, 0)$, by letting $\delta_s = \zeta^\frown(m+1, 0, 0, b)$. Go to stage $s+1$.

(c18) If ζ has outcome $(m, n, (ter))$ at the last ζ-stage, and B has change below $\delta_m(n)$, then redefine $\Delta_m(B)(n)[s] = 1 = A_s(n)$, and enumerate the savior number v of cycle (m, n) into C. Start cycle $(m+1, 0)$, by letting $\delta_s = \zeta^\frown(m+1, 0, 0, b)$. Go to stage $s+1$.

(c19) $\Lambda(B)(m)[s]$ is defined as 0 and m is in A_s. Then enumerate y into C, and create links between ζ and τ. Let $\delta_s = \tau^\frown gw$ and go to stage $s+1$.

(c20) After the enumeration of y into C, B changes below $\lambda(m)$, then, at the next τ-stage, the savior number v is put into C, and redefine $\Lambda(B)(m) = 1 = A(m)$. Start cycle $(m+1, 0)$.

If ζ is a plus-substrategy, then ζ works in a similar, but a simpler way, as indicated in the description before the construction part.

Phase 2: Construction of subtrees S_τ

For those τ which are active at δ_s, continue the construction of tree S_τ, a subtree of T_τ, as follows: for a node η on T_τ of length s, check whether η is in a region being terminated by a substrategy of τ. If yes, then we do not put η on S_τ. If not, then we put η (with its initial segments) on S_τ.

Phase 3: Definition of Γ

Find the least j such that $\Gamma(AB)(j)[s]$ has no definition. [*Without loss of generality, we assume that j is a number being selected by a minus $\mathcal{S}$-strategy α^- (via cycle (m, n) say) as its parameter, v or y, as otherwise, we can find a biggest $j' < j$ with such a property, and define the use $\gamma(j)$ the same as $\gamma(j')$.*]

Define $\Gamma(AB)(j)$ as $C_s(j)$ with the use $\gamma(j)$ based on the location of cycle (m, n) (of α^-) on the construction tree T.

If s is the first stage at which $\Gamma(AB)(j)$ is defined, then:

- If δ_s has priority higher than α^- or cycle (m, n), then we just define the use $\gamma(j)$ as $\gamma(j-1)[s]$.
- Otherwise, i.e. cycle (m, n) has higher priority, we define:
  - — The A-part of $\gamma(j)$ as the maximum of $m' + n'$, where (m', n') is a cycle on $\delta_{s'}$, where $s' \le s$ and $\delta_{s'}$ has priority higher than δ_s.
  - — The B-part of $\gamma(j)$ is the maximum of the B-uses in the computations associated to a node with priority not lower than δ_s.

If $\Gamma(AB)(j)$ has been defined before, then we assume that $\Gamma(AB)(j)[s']$ has definition, where $s' < s$ is the last such a stage, and we check whether $\Gamma(AB)(j)[s]$ is undefined due to A-changes or because of B-changes.

If it is because of B-changes (but not because of A-changes), then we just let $\gamma(j)[s] = \gamma(j)[s']$.

If it is because of A-changes, then we define $\Gamma(AB)(j)[s] = C_s(j)$ with use $\gamma(j)[s]$ as follows:

- The A-part of $\gamma(j)$ is the same as A-part of $\gamma(j)[s']$.
- The B-part of $\gamma(j)$ is the maximum of the B-uses in the computations associated to a node with priority not lower than δ_s.

This completes the construction of stage s.

End of construction.

5. Verification

We now prove that C and Γ constructed above satisfy all the requirements. Define TP as the leftmost path that has been visited infinitely many times. That is, $TP = \liminf_s \delta_s$ — the so-called *true path of the construction.*

Lemma 5.1. *TP is well-defined and has infinite length.*

We use $TP \upharpoonright k$ to denote the initial segment of TP of length k. We prove the following statement, which implies Lemma 5.1 immediately.

Lemma 5.2. *For each k, let $\xi_k = TP \upharpoonright k$. Then ξ_k has outcome $\mathcal{O}_k$ such that $\xi_k{}^\frown\mathcal{O}_k$ is on TP, i.e. $\xi_k{}^\frown\mathcal{O}_k \subset TP$. That is*:

(1) *ξ_k can be initialized at most finitely often.*
(2) *$\xi_k{}^\frown\mathcal{O}_k$ can be visited infinitely many times during the construction, and there are at most finitely many stages s such that δ_s is on the left of $\xi_k{}^\frown\mathcal{O}_k$.*
(3) *If ξ_k is an $\mathcal{S}$-substrategy, then $\mathcal{O}_k$ is an outcome of some cycle and this cycle can act and initialize those strategies at most finitely often.*

Proof. We prove Lemma 5.2 by induction on k.

When $k = 0$, ξ_0 is the root of the construction tree T, λ, which can never be initialized. As an $\mathcal{R}_0$-strategy, λ has outcomes, 1 and gw. If in the construction, at a stage s say, some substrategy puts a number into C, then λ will have outcome gw, until B changes and destroys such computations. If there is no such change from B, then λ has outcome gw on TP. If B does have such a stage, then at the next stage (note that each stage is a λ-stage), λ will have outcome 1. Thus, if λ cannot have outcome gw on TP, then λ has outcome 1 on TP. Also as λ is an $\mathcal{R}_0$-strategy, it has no action in the construction. Thus, the statement is true for $k = 0$.

Now suppose that the statement is true for all $\ell < k$, and we prove that it is also true for k.

As given, $\xi_k = \xi_{k-1}{}^\frown\mathcal{O}_{k-1}$. By the induction hypothesis, we can assume that after a stage s_0 large enough, δ_s cannot be on the left of ξ_k, and as a consequence, ξ_k can only be initialized by ξ_{k-1}, under outcome $\mathcal{O}_{k-1}$, at most finitely often, by (3) of the induction hypothesis for ξ_{k-1}. Thus, (1) is true for ξ_k.

For (2), if ξ_k is an $\mathcal{R}$-strategy, (2) is obviously true, as in this case, ξ_k has only two outcomes. If at a stage s say, some substrategy puts a number into C, then ξ_k will have outcome gw, until B changes and destroys such computations. By the same argument for λ, the root of T, we can show that the statement is true for ξ_k in this case.

So we assume that ξ_k is an $\mathcal{S}$-strategy. We assume that ξ_k is a minus-strategy, as the cases for plus-strategies are much simpler, and what we prove here for minus-strategies can be applied to plus-strategies with a bit of modification.

In the construction, ξ_k runs (possibly, infinitely many) cycles (m, n), $m, n \in \omega$, at ξ_k-stages. We first show the existence of a cycle (m, n) with an outcome $\mathcal{O}$ (either a termination outcome, or a standard density outcome, or a gadget outcome) on TP. That is, cycle (m, n) has outcome $\mathcal{O}$ infinitely often, and only finitely many cycles with priority higher than (m, n) can be initiated, each of which, once initiated, can be active at most finitely often. For this purpose, we need to show that the auxiliary partial computable functionals $\Lambda_{\xi_k}(B)$ and $\Delta_{\xi_k,m}(B)$, $m \in \omega$, are well-defined. For simplicity, we will omit the subscript ξ_k in the discussion, as it will make no confusion.

Note that in the construction, for any m, if $\Lambda(B)(m)$ is defined, with value different from $A(m)$, then at the next ξ-stage, the corresponding number y is enumerated into C, and a link between ξ and its mother node τ is created. This link will not be removed until B has changes, undefining $\Lambda(B)(m)$. When ξ is visited again, $\Lambda(B)(m)$ is defined as 1, equal to $A(m)$. As we are assuming $B <_T A$, Λ^B cannot be totally defined [otherwise, B would compute A correctly], which means that there is a least m such that either $\Lambda(B)(m)$ is not defined or $\Lambda(B)(m) \downarrow\neq A(m)$. In the former case, i.e. a cycle (m, n) for some n defines $\Lambda(B)(m)$ at a stage, then when B changes, the B-changes undefine both $\Lambda(B)(m)$ and $\Delta_m(B)(n)$, and when ξ is accessible again, cycle (m, n) defines $\Delta_m(B)(n)$ as 1, and as a consequence, cycle (m, n) will have no more definitions of $\Lambda(B)(m)$ — the job of defining $\Lambda(B)(m)$ is left to other cycles. If such a process repeats infinitely many times, and $\Lambda(B)(m)$ is not defined by any cycle, then this means that $\Delta_m(B)$ is defined as a total function, computing A correctly, which is impossible. So there is a least n'' such that either cycle (m, n'') does not define $\Lambda(B)(m)$ in the construction, or if it is defined, then $\Lambda(B)(m) \downarrow\neq A(m)$. In this case, ξ will have one of the outcomes of cycle (m, n) as true outcome.

Now fix m. We assume that $\Delta_m(B) \upharpoonright n$ is well-defined and equal to $A \upharpoonright n$. We consider the definition of $\Delta_m(B)(n)$. Recall that ξ_k's target is to find an infinite path in T_τ [here, τ is the mother node of ξ_k], and ensure that this path is not on S_τ.

- We assume that cycle (m, n) selects the region (x, z), at a stage s say.

If n is already in A_s, then $\Delta^B_m(n)$ is defined as 1, and equals to $A(n)$. In this case, $\Delta^B_m(B)(n)$ can be undefined only when B has small changes, and eventually $\Delta_m(B)(n)$ is defined. Note that cycle $(m, n+1)$ is initiated whenever cycle (m, n) defines $\Delta_m(B)(n)$ as $A(n)$, as whenever such a small B-change occurs, $\Delta_m(B)(n+1)$ [defined by cycle $(m, n+1)$] is undefined. *This makes $\Delta_m(B)$ well-defined.*

If n is not in A_s, then before n enters A, whenever B changes below $\delta_m(n)$, we redefine $\delta_m(n)$ as a number bigger than all the uses in the computations involved.

It can happen that n is not in A at all, and some computation involved in cycle (m, n) diverges (including those computations in the confirmation gadget), then $\delta_m(n)$ goes to infinite. In this case, $\Delta_m(B)(n)$ is not defined, but, on the other hand, cycle (m, n) finds a divergent computation (hence a divergence outcome) which is involved in the section of the region (x, z). If so, the $\mathcal{O}$ is the least divergence outcome (either a standard density outcome, or a gadget outcome).

If n enters A later, then at the next ξ_k-stage $s' > s$, say, we already see $n \in A_{s'}$, then the action at this stage s' is to terminate the nodes in the region (x, z). Again, after stage s', when B changes on small numbers, $\Delta_m(B)(n)$ can be undefined, and when we redefine it, we just define it as 1, with the use the same as before. This shows that cycle (m, n) succeeds in defining $\Delta_m(B)(n) = A(n)$, provided that the corresponding computations converge, and cycle $(m, n + 1)$ is started. *According to the construction, v is enumerated into C. The change of $A(n)$ is actually an A-permission for us to enumerate v into C — could be delayed as usual, to prevent $\Phi(BC)$ from entering the terminated region (x, z).* Note that once a cycle (m, n) succeeds in defining $\Delta_m(B)(n) = A(n)$, cycle (m, n) will not care whether it can find an infinite path in T_τ or not, and what it does is to hand such a task to cycle $(m, n + 1)$.

Thus, for n, either $\Delta_m(B)(n)$ is defined and equal to $A(n)$ (in this case, cycle $(m, n + 1)$ is started), or $\Delta_m(B)(n)$ is defined and is not equal to $A(n)$ (in this case, if cycle $(m, n + 1)$ is started before n enters A, then it will be stopped when n enters A, as A's change at n is a permission for the terination of nodes in the region (x, z)), or $\Delta_m(B)(n)$ is undefined (as a divergence outcome is found). In the case that $\Delta_m(B)(n)$ is defined and is not equal to $A(n)$, ξ will take outcome $(m, n, (ter))$, till B changes below $\delta_m(n)$. By $B <_T A$, $\Delta_m(B)$ cannot be a total well-defined function, computing A correctly. This shows the existence of a number n (least) such that either $\Delta_m(B)(n)$ is not defined, or $\Delta_m(B)(n)$ is defined but not equal to $A(n)$. In the latter case, ξ will take outcome $(m, n, (ter))$ forever, from the first ξ-stage after n enters A. The former case shows that ξ has either standard density outcome or gadget outcome. Let u be defined at stage s_u.

(1) If ξ has outcome (m, n, k, d) or (m, n, k, f) infinitely often, with $k \leq u$, then the least outcome which is true infinitely often is on TP.
(2) If ξ never has outcome (m, n, k, d) or (m, n, k, f), with $k \leq u$, after certain stage, then $x\tau_j$ is defined and will be kept the same after a stage big enough. [In the following, when we say that a number is defined and will be kept the same, we mean that the outcome $(m, n, x\tau_j, k, d)$ or $(m, n, x\tau_j, k, f)$ infinitely often, with $k \leq x\tau_j$, then the least outcome which is true infinitely often is on TP.]
(3) If $x\tau_{j-1}$ is defined and will be kept the same after a stage big enough, and ξ has outcome $(m, n, x\tau_{j-1}, k, d)$ or $(m, n, x\tau_{j-1}, k, f)$ infinitely often, with $k \leq x\tau_{j-1}$, then the least outcome which is true infinitely often is on TP.

$\vdots$

[*Iterate this process until τ_1 is considered.*]

(4) If v is defined and will be kept the same after a stage big enough, and ξ has outcome $(m, n, x\tau_1, k, d)$ or $(m, n, x\tau_1, k, f)$ infinitely often, with $x\tau_1 \geq k \leq v$, then the least outcome which is true infinitely often is on TP.

(5) If $z\tau_1$ is defined and will be kept the same after a stage big enough, and ξ has outcome $(m, n, z\tau_1, k, d)$ or $(m, n, z\tau_1, k, f)$ infinitely often, with $v \geq k \leq z\tau_1$, then the least outcome which is true infinitely often is on TP.

(6) If ξ never has outcome $(m, n, z\tau_1, k, d)$ or $(m, n, z\tau_1, k, f)$ after certain stage, and ξ has outcome (m, n, v, k, d) or (m, n, v, k, f) infinitely often, with $v \geq k \geq x\tau_2$, then the least outcome which is true infinitely often is on TP.

(7) If $z\tau_2$ is defined and will be kept the same after a stage big enough, and ξ has outcome $(m, n, z\tau_2, k, d)$ or $(m, n, z\tau_2, k, f)$ infinitely often, with $z\tau_2 \geq k \leq v$, then the least outcome which is true infinitely often is on TP.

$$\vdots$$

[*Iterate this process until* τ_j *is considered.*]

(8) If $z\tau_n$ is defined and will be kept the same after a stage big enough, and ξ has outcome $(m, n, z\tau_n, k, d)$ or $(m, n, z\tau_n, k, f)$ infinitely often, with $z\tau_n \geq k \leq v$, then the least outcome which is true infinitely often is on TP.

Thus, for ξ, we let m, n as above, the description above shows that ξ has an outcome on TP, and (2) is proved.

Let $\mathcal{O}_k$ be the outcome on TP, and (m, n) is the cycle. If there is no ξ-stage with $\Delta_m(B)(n) \neq A(n)$, then ξ has a standard density outcome or a gadget outcome, and in this case, this cycle has no action during the construction. If at a ξ-stage, we see $\Delta_m(B)(n)$ is defined with $\Delta_m(B)(n) \neq A(n)$, then at this ξ-stage, the action at this stage is to terminate region $(x, z)d$, and ξ takes outcome $(m, n, (ter))$. After this, by the choice of m, n, no change of B can undefine $\Delta_m(B)(n)$, and hence ξ will not take any further action, and (3) is proved for $\xi^\frown\mathcal{O}_k$. □

Lemma 5.3. *Let τ be any node on T. Then each version of S_τ is a computable subtree of T_τ.*

Proof. Fix τ. It is clear that S_τ is a subtree of T_τ, as all the nodes on S_τ are from T_τ. We can assume that at any stage s, all nodes on T_τ of length s are put on T_τ. The basic idea of constructing S_τ is to put a node on T_τ of length s on S_τ only at stage s (thus a node of length s is not put on S_τ at stage s, then this node will never be put on S_τ in the remainder of the construction). In our terminology, it just says that we need to ensure that any terminated node will be kept as terminated. All of these guarantee that S_τ is computable. □

Lemma 5.4. *All $\mathcal{R}$-requirements are satisfied along the true path TP.*

Proof. We prove it by induction on e. Let τ be the last $\mathcal{R}_e$ strategy on TP. τ exists by the construction of T. Let s_τ be the last stage at which τ is initialized. Then a subtree S_τ will be constructed by all substrategies below $\tau^\frown 1$.

If there is a link between a substrategy σ and τ created at a stage, showing a global win of τ, and this link is there forever (i.e. B does not change below the uses involved), then σ creates a global win for τ, such that either $C(y) \neq \Psi_\tau(\Phi_\tau(BC))(y)$ for some y, where w is selected by σ, or $\Phi_\tau(BC)$ is not a path of $[T_\tau]$.

So we assume that no such a link exists permanently. If some substrategy of τ has a standard density outcome on TP, or of some $\mathcal{R}$-strategy τ' below $\tau^\frown 1$ shows a gadget outcome for τ on TP, then these outcomes show that either the length agreement for τ has a finite limit, or some computation involved in the calculation of the length agreement for τ diverges. These outcomes show that τ is satisfied, and below these outcomes, τ has no further substrategies on TP.

Now we assume that τ has infinitely many substrategies on TP. Then each such substrategy σ is devoted to find an infinite path in $[T_\tau]$. We will show that each such σ will find such a path successfully. To see this, by our assumption, σ is on TP, and we assume that cycle (m, n) is the one with an outcome on TP. Cycle (m, n) will select the wanted numbers u, v, x, y, z (if not, then σ will provide an outcome showing that τ is satisfied at σ), and then $\Delta_m(B)(n)$ is defined after a stage big, which turns to be different from $A(n)$, as if $A(n)$ is the same as $\Delta_m(B)(n)$, then cycle $(m, n+1)$ will be started, contradicting the assumption of (m, n). As a consequence, the region (x, z) will be terminated, and T_τ does have an infinite path in this region, as otherwise, cycle (m, n) will notice this at the next σ-stage, and define $\Lambda(B)(m)$, starting cycle $(m + 1, 0)$, which is again impossible, by the choice of cycle (m, n).

This shows that all substrategies, i.e. all plus-substrategies and all minus-substrategies, are satisfied, and both $[S_\tau]$ and $[T_\tau]\backslash[S_\tau]$ are infinite. This shows that $[T_\tau]$ is not a thin class. It means that if $\Phi_\tau(BC)$ is Turing equivalent to $B \oplus C$, and $[T_\tau]$ contains $\Phi_\tau(BC)$ as a branch, then $[T_\tau]$ is not thin. Thus, $\mathcal{R}_e$ is satisfied.

This completes the proof of Lemma 5.4. $\square$

To complete the verification, we will prove that $C \leq_T A \oplus B$.

Lemma 5.5. $\Gamma(AB)$ *is well-defined with* $\Gamma(AB) = C$*. Thus* $B \leq_T B \oplus C \leq_T B \oplus A$*.*

Proof. We first show that Γ is total. Fix k and assume that after a stage s_0 large enough such that for any $k' < k$, $\Gamma(AB)(k')$ is defined and no A- or B-change can undefine it later. We now show that there exists a stage $s' > s_0$ such that $\Gamma(AB)(k)$ is defined and no A- or B-change can undefine it later.

Note that if k is not assigned to any node on T as a savior v or as a parameter y, then s_0 is the wanted stage, as $\Gamma(AB)(k)$ is defined as 0 and $\gamma(k)$ is defined to equal $\gamma(k-1)$.

So we consider that k is assigned to a strategy α, in particular a cycle (m, n), as a savior v or as a parameter y. Without loss of generality, we assume that k is assigned as y. Then when we define $\Gamma(AB)(y)$ as a stage s, we define $\Gamma(AB)(y)$ as 0 with the A-part of the use the maximum of the numbers in cycles (m', n') along δ_s, the B-part of the use the maximum of the B-part of the *use of* $\Gamma(AB)(y-1)$, *and* *also the B-uses involved in computations in those strategies $\leq\delta_s$, and the B-uses involved in computations associated with those saviors attached to some nodes with priority higher than δ_s.*

- Check whether A or B changes on the corresponding part of the use $\gamma(k)$.

If there is no such a change, then $\Gamma(AB)(y)$ is 0 and in this case, $C(y)$ is also 0, and we will show it below. If δ_s has priority higher than cycle (m, n), then at stage s, (m, n) is initialized and then y cannot be enumerated into C later. So we assume that δ_s has priority lower than cycle (m, n).

If A and B do not have changes below the A-part or the B-part of $\gamma(y)$, then $\Gamma(AB)(y)$ is 0, and δ never moves to the left of δ_s, so y cannot be enumerated into C. Thus $\Gamma(AB)(y) = 0 = C(y)$.

If such a change exists, at stage s' say, we check whether y is in $C_{s'}$ or not (i.e. check whether cycle (m, n) is accessible at stage s'). If y is enumerated into C at stage s', then $\Gamma(AB)(y)$ is redefined as 1, with use $\gamma(y)$ the same as the previous one. If not, $\Gamma(AB)(y)$ is redefined as 0, with use $\gamma(y)$ the same as the previous one (we keep it the same just because we will keep the γ-use nondecreasing). In any case, $\Gamma(AB)(y)$ is redefined with value $C(y)$, and the use $\gamma(y)$ has new definition with $\gamma(y)[s'] = \gamma(y)[s]$. By repeating the process above, we either see that y is enumerated into C, at a stage when $\Gamma(AB)(y)$ is undefined, or eventually, we will see a stage s^* at which A and B will have no changes below the corresponding parts of use $\gamma(y)$, which means that $y \in C$ if and only $x \in C_{s^*}$. Thus, $\Gamma(AB)(y)$ cannot be undefined after stage s^*, and

$$\Gamma(AB)(y) = \Gamma(AB)(y)[s^*] = C_{s^*}(y) = C(y).$$

By induction, we have that $\Gamma(AB)$ is total, and equals C. □

This completes the verification part and hence the proof of Theorem 1.3.

6. Proof of Theorem 1.4

In this section, we prove Theorem 1.4, which is an improvement of an earlier density result of Cenzer, Downey, Jockusch and Shore [4], where the wanted members of thin classes were not required to have c.e. degrees. To prove Theorem 1.4, we will construct a c.e. D in stages, such that its complement, $\overline{D}$, will be the wanted member of a thin Π^0_1 class, $[T]$ (constructed by us).

As in paper [4], we assume that B is the even part of A, so we replace A with $A \sqcup B$, where A consists of only odd numbers and B consists of only even numbers. D will be constructed by stages, and D_s is the enumeration of D by stage s. $\overline{D}_s$ denotes the complement of D_s, which is of the form $\{d_{0,s} < d_{1,s} < d_{2,s} < \cdots\}$.

We also construct a computable tree T such that $[T]$ is a thin Π^0_1 class. Again, T will also be constructed in stages, and we denote T_s as the approximation of T by stage s, and $T = \bigcup_s T_s$. Here, as in paper [4], we assume that on T, 0 is on the right of 1 and that all the paths (except those deadends) on T_s have length $d_{s,s}$.

In the construction, at a stage s, our only action is to let $D_{s+1} = D_s \cup \{d_{j,s} : k \leq j \leq s\}$ for some k, a "dumping action", which was developed for the construction of co-retraceable sets.

Our target is to use the rightmost path of T_s to code $\overline{D}_s$: on T_s, we have nodes $\sigma_{j,s}$ coding the elements $d_{j,s}$ in $\overline{D}_s$ for $j \leq s$ (We assume here that 0 is in D so $\sigma_{0,s}$ is not empty). That is,

- $\sigma_{0,s} = 1^{d_{0,s}}$,
- having $\sigma_{j,s}$, $j < s$, we let $\sigma_{j+1,s} = \sigma_{j,s}{}^\frown 0^\frown 1^{d_{j+1,s}-d_{j,s}-1}$.

All other paths (except for those deadends, of course) on T_s will be of the form $\sigma_{j,s}{}^\frown 1^{d_{s,s}-d_{j,s}}$ for $j \leq s$, and we will denote it by $\sigma_{j,s}{}^\frown 1^*$, if we do not care about the exact number of 1's after $\sigma_{j,s}$. When $d_{i,s}$ is enumerated into D at a stage s, then any path on T_s extending $\sigma_{i,s}{}^\frown 0$ is terminated forever. A property of $[T]$ is that the rightmost path of $[T]$ will be the only one limit point in $[T]$, i.e. the limit of all other paths. So, to make $[T]$ thin, we will construct T as in [4] to make $[T]$ a minimal Π^0_1 class, with exactly one limit path.

The following requirements will guarantee that $[T]$ is minimal, and will call these minimality requirements:

$\mathcal{R}_e$: If $[P_e] \subseteq [T]$, then either $[P_e]$ is finite or $[T]\backslash[P_e]$ is finite.

Here, P_e is the eth-primitive recursive tree.

When B is empty, a basic case. We first consider this simple case, when B is empty, and give a brief description of constructing T.

For P_e, we will use $P_{e,s}$ to denote the approximation of P_e by stage s. We assume that each path in $P_{e,s}$ is of length $d_{s,s}$. As in paper [4], to satisfy $\mathcal{R}_e$, what we will do at every stage s is to check whether some $\sigma_{i,s}{}^\frown 1^* \notin P_{e,s}$, and if so, we will re-route the construction through this $\sigma_{i,s}{}^\frown 1^*$. This action of re-routing is actually an enumeration of numbers into D.

We now describe how to implement the idea above, incorporated with the A-permission.

Say that $\mathcal{R}_e$ is *active at* $i > e$ (*i.e. active via* $\sigma_{i,s}$) *at a stage* s, if:

(1) $\sigma_{i,s}{}^\frown 1^* \notin P_{e,s}$, and
(2) for all j with $e \leq j \leq i$, $\mathcal{R}_e$ has not been currently declared *to be met* at j via $\sigma_{j,s}$.

Say that $\mathcal{R}_e$ *requires attention* at a stage s if it is active at this stage at some i.

As in standard permitting argument, we will define a p.r. function Δ as an approximation of A by stages, whenever $\mathcal{R}_e$ requires attention. That is, if $\mathcal{R}_e$ requires attention at i at stage s, then we define $\Delta(k) = A_s(k)$ for those $k < i$ such that $\Delta(k)$ has no definition by stage s.

Suppose that at stage $t > s$, p is the least number entering A. We check whether $\Delta(p)$ has defined or not, i.e. whether some $\mathcal{R}_e$ has been active at some $i \geq p$ at a previous stage. If not, do nothing. Otherwise, we let $\mathcal{R}_e$ *receive attention at* i, by *re-routing* the construction through $\sigma_{i,t}{}^\frown 1^*$, i.e. we will let $\sigma_{i,t+1}$ extend $\sigma_{i,t}{}^\frown 1^*$, and terminate those nodes extending $\sigma_{i,t}{}^\frown 0$. If so, we *declare* that $\mathcal{R}_e$ *is satisfied at* i via $\sigma_{i,t}$ at stage t. Once $\mathcal{R}_e$ receives attention, it will be satisfied forever (remember that B is empty in this simple case), and by definition, it will not be active anymore.

Of course, if later, a smaller number q enters A at a stage $t' > t$ say, and $\mathcal{R}_e$ was also active at some q with $q \le i' < i$, then $\mathcal{R}_e$ will receive attention at i' at stage t'.

We will show soon that if $\mathcal{R}_e$ is not met, then there exist infinitely many i at which $\mathcal{R}_e$ is active, which means that Δ has infinitely many extensions and hence is a total computable function. This would show that A is computable, a contradiction. Thus $\mathcal{R}_e$ cannot be active at infinitely many i's.

So we let i be the largest at which $\mathcal{R}_e$ is active, and we will show now that $\mathcal{R}_e$ is met. We assume that $[P_e] \subset [T]$, and further, $[P_e]$ is infinite, then by the construction, $\mathcal{R}_e$ cannot receive attention at any stage, and the rightmost path of $[T]$ is also in $[P_e]$. This means that above σ_i, we cannot have any node in $T\backslash P_e$, and hence all nodes above σ_i in T and in P_e are the same. Thus, all paths in $[T]\backslash[P_e]$ extend some node σ_j, $j < i$. By the construction of T, there are only finitely many such paths and hence $[T]\backslash[P_e]$ is finite, and $\mathcal{R}_e$ is met.

Thus, for the case when $B = \emptyset$, the construction of D is a standard direct permitting construction, and the reduction of D to A, via Γ say, is obvious: whenever a number is enumerated $x = d_{i,s}$ into D, some number $p < i$ enters A at that stage.

For the case when B is nonempty, we will have the definition of *permanently active at* i, where we say that $\mathcal{R}_e$ is *permanently active at* i if $\mathcal{R}_e$ is active at i at stage s via $\sigma_{i,s}$ for almost all stages s. For B empty, it is clear that once $\mathcal{R}_e$ is active at i at stage s, then it will be active at i at any stage after s, and hence it is permanently active at i.

When B is nonempty, a general case. We now come to the general case, when B is nonempty. We will construct a p.r. functional Γ such that $\Gamma(A \sqcup B) = D$, a global requirement.

For the part of coding B into D, if a number i enters B at stage s, then we will enumerate $d_{i,s}$ into D. *We assume here that $i \le s$ if i enters B at stage s.* In the construction, we can put in $d_{i',s}$, to satisfy $\mathcal{R}$-requirements, and what we will guarantee is that $\lim_s d_{i,s} = d_i$ exists for each i. Having this, we can prove $B \le_T D$ as follows: for a number i, to decide whether i is in B or not, we use D as oracle to find a stage s_i such that $d_i = d_{i,s_i}$, and then $i \in B$ if and only if $i \in B_{s_i}$.

Let us come back to the discussion of how to satisfy a single $\mathcal{R}_e$ requirement. The main point is that the coding of B can fail the basic module given above, *perhaps several times*, in the way that after we declare that $\mathcal{R}_e$ is satisfied at i at stage s, some $j < i$ enters B later, causing us to re-route the construction through $\sigma_{j,s}{}^\frown 1^*$. As in a standard density argument, this kind of failure will provide opportunities to extend the definition of a p.r. functional Δ to show that $A = \Delta(B)$, which, of course, is not true, as we are assuming that $B <_T A$. In particular, if $[P_e]$ is an infinite subclass of $[T]$ and $[T]\backslash[P_e]$ is infinite, then, after finitely many tries, some win of $\mathcal{R}_e$ will be obtained and this win cannot be undone by coding of B.

Say that $\mathcal{R}_e$ *requires attention* at a stage s if it is active at this stage at some i, where we say that $\mathcal{R}_e$ is *active at* $i > e$ (*i.e. active via* $\sigma_{i,s}$) *at a stage* s, if:

(1) $\sigma_{i,s}\frown 1^* \notin P_{e,s}$, and
(2) for all j with $e \leq j \leq i$, $\mathcal{R}_e$ has not been currently declared *to be met* at j via $\sigma_{j,s}$.

Here, $\sigma_{i,s}$ codes the first i elements of $\overline{D}_s$. As B is coded into D as described above, $\sigma_{i,s}$ always contains information of B_s (up to some number, of course). So in the construction of T, at each stage s, we have $\sigma_{i,s}$ for $i \leq s$, coding an initial segment of B_s, and eventually, as we will guarantee that $\overline{D}$ is infinite, each initial segment of B will be coded in some initial segment of $\overline{D}$, and hence B is reducible to D. To be specific, we can construct a p.r. functional Θ such that $B = \Theta(D)$ as follows: for any x, whenever we define $\Theta(D)(x)$ at stage s, we always define it the same as $B(x)$ with use $\theta(x)[s] = d_{x,s}$, and $\Theta(D)(x)$ can be undefined at a stage t if D has a change below $d_{x,s}$. *In particular, when* x *enters* B, *our action of coding is to put* $d_{x,s}$ *into* D, *which undefines* $\Theta(D)(x)$. $\Theta(D)(x)$ can also be undefined by the bumping actions and also the actions of satisfying $\mathcal{R}$-requirements. All together, these actions will undefine $\Theta(D)(x)$ finitely many times, and eventually, we will have $B(x) = \Theta(D)(x)$. We will guarantee that $\overline{D}$ is infinite, making $\Theta(D)$ total. The construction of Θ is fairly standard, and we will not include it in the construction part.

Now we consider how to satisfy $\mathcal{R}_e$-requirement, with coding of B and A-permissions involved. The basic module runs the following actions several times (infinitely, perhaps). Recall that when $\mathcal{R}_e$ *requires attention* at stage s, $\mathcal{R}_e$ is active at this stage via $\sigma_{i,s}$, and B's change at some $j < i$ will drive $\sigma_{i,s}$ out of T. We will run the following procedures, perhaps infinitely many times.

(1) At stage s, check whether $\mathcal{R}_e$ requires attention at some $i \leq s$. We do nothing if no such i exists. Otherwise, let i be the least one, and
    — for $j \leq i$, if $\Delta(B)(j)$ is not defined, define $\Delta(B)(j)[s] = A(j)[s]$ with use $\delta(j) = i + 1$.
(2) Wait for A to change below $i + 1$. Declare that the $\mathcal{R}_e$-requirement is waiting for a A-change below $i + 1$ (i.e. an A-permission at i).
    [*If* B *change below* $\delta(i)$ *first, then* $d_{i,s}$ *is enumerated into* D (*by the dumping action*) *and hence* $\sigma_{i,s}\frown 1^*$ *is abandoned.*]
(3) When some $j \leq i$ enters A at stage $t+1$, we consider the least i' with $j \leq i' \leq i$ such that $\mathcal{R}_e$ is active at some i' at stage $t+1$, and let $\sigma_{i',t+1}$ extend $\sigma_{i',t}\frown 1^*$, terminate those nodes extending $\sigma_{i',t}\frown 0$. We declare that $\mathcal{R}_e$ has a win via $\sigma_{i',t+1}$.

[*Note that here, $\mathcal{R}_e$-requirement acts at i', instead of i. $\mathcal{R}_e$-requirement may not be active at stage $s+1$.* Also $A(j)[s] \neq \Delta(B)(j)[s]$].

(4) Wait for B to change below i' at stage w. Then $\sigma_{i',t}{}^\frown 1^*$ becomes B-incorrect and hence is abandoned.

[*If there is no such a stage w, then $\mathcal{R}_e$ has a win.*]

Here when we define $\Delta(B)(j)$ at stage s, this means that $\mathcal{R}_e$ is active at stage s. We do this because in case that j enters A, $\mathcal{R}_e$ will act at i' and consequently, we will either have a win for $\mathcal{R}_e$ or have a B-change below i' (so below $i+1$), i.e. the change of B at stage w above, undefining $\Delta(B)(j)$.

We now check whether $\mathcal{R}_e$ is also active at some $i'' < i'$ at stage w.

- If no, then we just wait till next stage at which $\mathcal{R}_e$ is active again at $i^\dagger > i$ at a stage $w' > w$, and then for those j with $\Delta(B)(j)$ defined at stage s, define $\Delta(B)(j)[w'] = A(j)[w']$ with use $\delta(j)[w'] = i^\dagger$.
- If $\mathcal{R}_e$ is active at some $i'' < i'$ at stage w, then for those $j \leq i''$, we will redefine $\Delta(B)(j)[w] = A(j)[w]$ with the same use $\delta(j)[w] = i$, and for those $j > i''$, we will wait till a stage $w' > w$ at which $\mathcal{R}_e$ is active again at $i^\dagger > i$ and define $\Delta(B)(j)[w'] = A(j)[w']$ with use $\delta(j)[w'] = i^\dagger$.

We say that $\mathcal{R}_e$ is *permanently active* at i, if $\mathcal{R}_e$ waits for A's permission at i at all stages after a certain stage s. By $B <_T A$, we can show that $\mathcal{R}_e$ can be *permanently active* at at most finitely many i, and as a consequence, $\mathcal{R}_e$ is met from some stage onwards. Suppose for a contradiction that there are infinitely many i's at which $\mathcal{R}_e$ is permanently active. We will show that for any k, we will compute $A \upharpoonright k$, recursive in B.

Fix k. By induction, we assume the existence of $s(k-1)$ after which $A(j) = \Delta(B)(j)$ is true for each $j < k$. We now show the existence of $s(k) > s(k-1)$ after which $A(k) = \Delta(B)(k)$ is true.

We let s be a stage at which $\Delta(B)(k)$ is first defined, and then search for an i_k and a stage $s_k > s$ such that $\mathcal{R}_e$ is active at i_k and $B \upharpoonright i_k = B_{s_k} \upharpoonright i_k$. The existence of s_k is guaranteed by the assumption that $\mathcal{R}_e$ is permanently active at infinitely many i's. Then we have $A(k) = \Delta(B)(k)[s]$ for all stage $s > s_k$, because otherwise, we will have a win for $\mathcal{R}_e$, and this win cannot be undone by changes of B. Now we take $s(k) = \max\{s(k-1), s_k\}$. Thus $\Delta(B)$ computes $A(k)$ correctly. This completes the induction step.

This proves $A = \Delta(B)$, if we assume that $\mathcal{R}_e$ is permanently active at infinitely many i's, which is a contradiction. Therefore, $\mathcal{R}_e$ can be permanently active at only finitely many i's.

This definition of reduction Δ is standard, and we will not specify it in the construction part. What we will indicate is that at any stage, we will check whether $\mathcal{R}_e$ is active at certain i at a stage s, and whether it has permissions from A and B to act at stage s.

Remark. We have seen that $\mathcal{R}_e$ can be permanently active at only finitely many i's. It is possible that $\mathcal{R}_e$ can require attention infinitely many times, and almost all such requests are undone because of B's early changes.

Construction of Γ. For the construction of p.r. functional Γ, making $D = \Gamma(A \sqcup B)$, at stage s, we define $\Gamma(A \sqcup B)(x)[s] = D(x)[s]$ for $x \leq d_{s,s}$ with the use $\gamma(x)$ as follows: if $x = d_{i,s}$ for some $i \leq s$, we define $\gamma(x) = i$, and if x is not of the form $d_{i,s}$, we either redefine it as the previous definition, if it has definition before, or let it be the same as $\gamma(y)$, where y is the largest $d_{j,s} < x$. Note that $x = d_{i,s}$ can be enumerated into D by bumping actions (so we have either an A-permission or a B-change less than i), and $\Gamma(A \sqcup B)(x)$ is undefined in both cases. Again, by bumping actions, when $d_{i,s}$ enters D, all numbers between $d_{i,s}$ and $d_{s,s}$ are enumerated into D, if they are not in D yet.

We will see soon that when interactions between $\mathcal{R}$ requirements are considered, our construction of D will actually utilizes delayed permissions.

Interactions between two $\boldsymbol{\mathcal{R}}$ requirements and delayed permissions. The A-permissions for the basic case when $B = \emptyset$ is a direct permission, as once a permission is given, one requirement is met, until it is injured by meeting a requirement with higher priority. That is, in this basic case, the interaction among $\mathcal{R}$ requirements is simple and direct. It is not true anymore, when B is nonempty. We first consider the interactions between *two* $\mathcal{R}$ requirements, say $\mathcal{R}_e$ and $\mathcal{R}_f$ with $e < f$. That is, $\mathcal{R}_e$ has higher priority.

If $\mathcal{R}_e$ is satisfied at i via some $\sigma_{i,s}{}^\frown 1^*$, we would not try $\mathcal{R}_e$ again in the cone above $\sigma_{i,s}{}^\frown 1^*$, and $\mathcal{R}_f$ will be satisfied exactly as in the basic module. This is the point of last clause in the definition of being active, since $\mathcal{R}_e$ cannot be active via any string in this cone.

The case that needs more consideration is when both $\mathcal{R}_e$ and $\mathcal{R}_f$ (hence, more requirements) are ready to act at the same stage. For instance, at stage s, we might have that $\mathcal{R}_e$ is ready to act at i, via $\sigma_{i,s}$ say, and $\mathcal{R}_f$ is also ready to act at j via $\sigma_{j,s}$. There are two cases, depending on which one is bigger, i or j.

If $i < j$, then, in some sense, there is no problem if both are permitted by $i' \leq i$ entering A at stage t. We will re-route the construction of T through $\sigma_{i,s}{}^\frown 1^*$ at this stage, and the nodes above $\sigma_{j,s}$ will be terminated, or abandoned. We will follow this idea. *Of course, this is also true if we might first have been satisfying $\mathcal{R}_f$ at j via $\sigma_{j,s}$, and then a smaller number entering A will allow us to work toward satisfying $\mathcal{R}_e$ at i.*

However, the case when $i > j$ is a bit tricky. Suppose that some $i' \leq j$ enters A first at stage t. This argument also encompasses the scenario that we were meeting $\mathcal{R}_e$ at i first and then, when $i' \leq j$ enters A later, causing us to have a close look at $\mathcal{R}_f$ and j. We consider both cases combined here. As both are permitted by i', it is reasonable to re-route the construction through $\sigma_{i,s}{}^\frown 1^*$, by priority. But it can happen that some q enters B later, with $j < q < i$, causing us to re-route the construction of T through $\sigma_{q,s}{}^\frown 1^*$. Thus, the temporary satisfaction of $\mathcal{R}_e$ fails, and on the other hand, we have no action to satisfy $\mathcal{R}_f$, even though an A-permission at i' at stage t is given for $\mathcal{R}_f$ to act at j via $\sigma_{j,s}$, which means that the given A-permission is not used by $\mathcal{R}_f$ properly. This cycle could repeat infinitely often and we would fail to meet $\mathcal{R}_f$ because we always try to satisfy $\mathcal{R}_e$ first, who has higher priority.

Note that the enumeration of q into B actually provides another opportunity for $\mathcal{R}_f$ to receive attention, and this is the idea of delayed permission in the construction. That is, when i' enters A, then this change undefines $\Gamma(A \sqcup B)(d_{j,s})$, and we can define $\Gamma(A \sqcup B)(d_{j,s})$ again, at stage t say, by letting $\gamma(d_{j,s})$ be i, so, when q enters B at stage w, $\Gamma(A \sqcup B)(d_{j,s})$ is undefined again, and we can enumerate $d_{j,s}$ into D, i.e. we meet $\mathcal{R}_f$ by re-routing the construction through $\sigma_{j,s}\frown 1^*$ [$\mathcal{R}_e$ *at* i *is no longer there to stop* $\mathcal{R}_f$]. Thus, $\mathcal{R}_f$ receives attention at stage w. So, even though $\mathcal{R}_f$ does not act at j via $\sigma_{j,s}$ immediately at stage t, when i' enters A, this enumeration of i' allows us to redefine $\Gamma(A \sqcup B)(d_{j,s})$ with a bigger use, i.e. i, and hence allows us to have a deferred action at stage w, when q enters B. For this process, we will name it as $\mathcal{R}_f$ *at* j *defers to* $\mathcal{R}_e$ *at* i *at stage* t.

For convenience, we define a function g to record this process. That is, when we set the use $\gamma(d_{j,s}, s)$ as i (we are assuming that $i > j$), we define $g(j, s) = i$. So:

- when we define $\Gamma(A \sqcup B)(d_{j,s})$ for the first time, at stage s, we set the use $\gamma(d_{j,s}, s)$ as j, and we let $g(j, s) = j$, and
- this use can be lifted up to i later at stage t say, i.e. $g(j, t) = i$, when $\mathcal{R}_f$ at j defers to $\mathcal{R}_e$ at i at stage t.

In the construction, we will guarantee that $\gamma(d_{j,s})$ can be lifted up at most finitely many times, and hence ensure that $\Gamma(A \sqcup B)(d_{j,s})$ is defined. *We drop the oracle* $A \sqcup B$ *if there is no confusion.*

The argument above assumes that no other things happen from stage s to t to stage w, which is a simple case. It can happen that after stage s, $\mathcal{R}_e$ becomes active at some $k < i$ with $j < k < q$. Thus, $\mathcal{R}_f$ can defer to $\mathcal{R}_e$ once again, and this time at k for the least such k. [*Note that* $k \leq i$, *and since* R_e *is never active in the cone above somewhere, it is apparently met. This delay can happen at most* $i - j$ *many times.*]

We will guarantee that $\lim_s d_{i,s} = d_i$ exists, which implies $D = \Gamma(A \sqcup B)$. This will be done by analyzing the reason that makes $d_{i,s}$ and $g(i, s)$ move.

Now we explain how both $\mathcal{R}_e$ and $\mathcal{R}_f$ are met in this set-up.

- $\mathcal{R}_e$ is met by exactly the same argument as above, and hence $\mathcal{R}_e$ has at most finitely many permanently active i.
  Let i_0 be the largest one such that $\mathcal{R}_e$ is permanently active i_0 and suppose that we are now only working after stage s_0 above $\sigma_{i_0,s_0} = \sigma^*_{i_0}$ (and hence $A_{s_0} \upharpoonright i_0 = A \upharpoonright i_0$).
- $\mathcal{R}_f$ is also met. There are two possibilities:

(a) If there is a cone above some $\sigma_i{}^\frown 1^*$, in which we have permanently met $\mathcal{R}_e$, then the argument for $\mathcal{R}_f$ is *exactly* the same as the one in the basic module in this cone.

(b) If there is no such a cone, then no attack on $\mathcal{R}_e$ can succeed. In this case, we can also show that $\mathcal{R}_f$ can be permanently active at at most finitely many j's and $\mathcal{R}_f$ is met.
Suppose for a contradiction that $\mathcal{R}_f$ has infinitely many permanently active j. To compute $A(p)$, we wait for a stage $s > s_0$ where $\mathcal{R}_f$ has at least p many active $j_1 < \cdots < j_p$, which are all B-correct. These numbers exist by our assumption.
We claim that $A(p) = A_s(p)$. If not, then p enters A at a later stage $t > s$. Since $p \leq j_p$, at stage t, if we do not meet $\mathcal{R}_f$ at one of these $j_k \leq j_p$, it can only be the case that this is deferring to $\mathcal{R}_e$ at some $i > j_p$. But since this satisfaction of $\mathcal{R}_e$ is not permanent, after some stage, it will be undone by coding of B. As argued above, this can happen at most $i - j_p$ times, and all such attacks on $\mathcal{R}_e$ on $i \leq g(j_p)$ will be canceled. Thus, we can meet $\mathcal{R}_f$ at j_p or below j_p eventually, a contradiction.
Thus A is reducible to B, a contradiction again.

When more requirements are considered. The idea above can be extended to handle several requirements, with more complexity, of course, due to the subtlety of tracking the movement of use of $\gamma(d_{j,s}, s)$, i.e. the approximation of the value of $g(j)$. Let us have a look at a few scenarios first. We will take the interactions among $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$ as an example. Note that the priority among these requirements is decreasing.

- Initially, there were only $\mathcal{R}_2$ and $\mathcal{R}_3$ which are active at j and k with $k < j$ respectively. As argued in the two-requirement case, if A permits us to re-route the construction through $\sigma_{k,s}{}^\frown 1^*$ (hence also permits through $\sigma_{j,s}{}^\frown 1^*$), then we would raise $g(k,s)$ to j and re-route the construction through $\sigma_{j,s}{}^\frown 1^*$. *We assume this case first.*

  Then the deferred $\mathcal{R}_3$ at k waits for a B-change, which would remove $\mathcal{R}_2$ at j, and allow us to re-route the construction through $\sigma_{k,s}{}^\frown 1^*$.

  — Next, at some stage $t > s$, $\mathcal{R}_0$ becomes active at $\sigma_{i,t}$ which *extends* $\sigma_{j,s}{}^\frown 1^*$, and A permits the construction to re-route the construction through $\sigma_{k,s}{}^\frown 1^*$ at stage t and we will meet $\mathcal{R}_0$ at $\sigma_{i,t}{}^\frown 1^*$.
  In this case, we would still define $g(k,t) = j = g(k,s)$, i.e. we keep the γ-use $\gamma(d_{k,s})$ as j, because the $\mathcal{R}_3$ would work above $\sigma_{j,s}{}^\frown 1^*$, until the B-coding removes the satisfaction of $\mathcal{R}_2$ at $\sigma_{j,t}$, which means that $\Gamma(d_{k,s})$ is undefined again, and $\mathcal{R}_0$ and $\mathcal{R}_2$ cannot be met by their previous actions. This will be called *extending the deferring* of $\mathcal{R}_3$ at k to $\mathcal{R}_0$ at i.

  — However, the situation would have been different if there is an $\mathcal{R}_1$ having now an active position l between k and j, which was permitted to act at t (so $\mathcal{R}_0$

at i is also simultaneously permitted, as $k < l < j < i$). In this case we would need to defer $\mathcal{R}_1$ at l to $\mathcal{R}_0$ at i by setting $g(l,t) = i$. Once $\mathcal{R}_0$ had been found to fail, then the construction would be re-routed through $\sigma_{l,t}{}^\frown 1^*$.
In this case, there is no need to defer $\mathcal{R}_2$ at j to $\mathcal{R}_1$ at l, because l is less than j, and once $\mathcal{R}_1$ fails at $\sigma_{l,t}{}^\frown 1^*$, B must have change below l and hence below j, and then the construction would re-route through $\sigma_{l,t}{}^\frown 1^*$.
For $\mathcal{R}_3$, we need to defer $\mathcal{R}_3$ at k to $\mathcal{R}_1$ at l, because k is less than l. Once $\mathcal{R}_1$ fails at $\sigma_{l,t}{}^\frown 1^*$, B's change below l (hence below j) allows us to re-route the construction through $\sigma_{l,t}{}^\frown 1^*$. This will be called *shifting upward the deferring* of $\mathcal{R}_3$ at $\sigma_{k,s}$ from $\mathcal{R}_2$ at j via $\sigma_{j,t}$ to $\mathcal{R}_0$ at i via $\sigma_{i,t}$.

- Returning to the three requirements $\mathcal{R}_0, \mathcal{R}_2$ and $\mathcal{R}_3$ with the same positions, but with a slight different scenario. That is, initially, $\mathcal{R}_2$ at j deferring to $\mathcal{R}_0$ at i [*instead of being met at j as above*], and the construction would be re-routed through $\sigma_{i,t}{}^\frown 1^*$, which extends $\sigma_{j,t}{}^\frown 0^\frown 1^*$. Then, when k was A-permitted, we would raise $g(k,s)$ to i, because, according to the assumption of this case, we do not have a win for $\mathcal{R}_2$ at j via $\sigma_{j,t}{}^\frown 1^*$ yet, and once $\mathcal{R}_0$ fails at i via $\sigma_{i,t}{}^\frown 1^*$ later, we could not expect for a change of B below j, but we need to use a (delayed) A-permission at k to re-route the construction through $\sigma_{k,t}{}^\frown 1^*$.

  Suppose that $\mathcal{R}_0$ fails at $\sigma_{i,t}{}^\frown 1^*$ later at a stage w (so B changes below i, undefining $\Gamma(d_{k,w})$), and also $\mathcal{R}_2$ becomes active at j' via $\sigma_{j',t}{}^\frown 1^*$ ($j' < i$), then, as both $\Gamma(d_{k,w})$ and $\Gamma(d_{j',w})$ are undefined, $\mathcal{R}_3$ at k defers to $\mathcal{R}_2$ at j'. This will be called *shifting downward the deferring* of $\mathcal{R}_3$ at $\sigma_{k,s}$ from $\mathcal{R}_0$ at i via $\sigma_{i,t}$ to $\mathcal{R}_2$ at j' via $\sigma_{j',w}$.

  Note that at stage w, $\mathcal{R}_3$ at $\sigma_{k,s}$ defers to $\mathcal{R}_2$ at j' via $\sigma_{j',w}$, which is exactly the first case above.

Thus, among these four requirements, the deferring actions of $\mathcal{R}_3$ at k, i.e. extending, shifting upward and shifting downward, can happen at most finitely many times. As $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2$ can be permanently active at at most finitely many times, and eventually, the requests of $\mathcal{R}_3$ can be deferred to higher priority requirements at most finitely many times. This will guarantee that $\mathcal{R}_3$ can be satisfied after a certain big stage.

This explanation can be extended to several requirements of arbitrary size easily, where the most important point is to clarify the actions of extending and shifting the deferrings, upward and downward. In general, if we have n many $\mathcal{R}$ requirements, Case 4 (shifting downward) above can repeat a few times, and forming a sequence of $n-1$ deferrings, all waiting for a B-change to invalidate $\mathcal{R}_0$'s action, and the shifting of deferrings upward is caused by changes of A. Once we have such a B-change, we check which $\mathcal{R}_e$-requirement with the highest priority can use this B-change at k to act and enumerate numbers into D (with pumping actions). In this case, the deferring of $\mathcal{R}_e$ to higher priority requirements is released, and $\mathcal{R}_e$ has opportunity to take action to get satisfied. Once B has an even smaller change (less than k, of course), which can invalidate $\mathcal{R}_e$'s action just now, and providing $\mathcal{R}_{e'}$ to take action. So for $\mathcal{R}_n$, after an A-permission is given, either it keeps in deferring state, and it will be satisfied in a cone where some higher priority requirement is satisfied, or after all the deferrings are released, it will eventually have a chance to act (satisfied), or invalidated by a B-change (so the corresponding part of $\Delta(B)$ is rectified).

Construction of D and T. We are now ready to present the construction. We assume that at each stage, at most one element enters $A \sqcup B$, and we let c_s be the number entering $A \sqcup B$ at stage s, if any. We always assume that $c_s < s$.

Stage 0: Set $D_0 = \{0\}$, $T_0 = \emptyset$ and Γ not defined yet everywhere.

Stage $s+1$: Having D_s, T_s, and also $\Gamma(A_s \sqcup B_s)$ defined up to $d_{s,s}$. We have $\overline{D}_s = \{d_{0,s} < d_{1,s} < \cdots < d_{s,s}\}$ and T_s contains s non-terminal leaves, with the rightmost one as $\sigma_{s,s}$, defined by induction on $j < s$ as follows:

$$\sigma_{0,s} = 1^{d_{0,s}} \quad \text{and} \quad \sigma_{j+1,s} = \sigma_{j,s}{}^\frown 0^\frown 1^{d_{j+1,s} - d_{j,s} - 1}$$

and the other non-terminal leaves are of the form $\sigma_{j,s}{}^\frown 1^{d_{s,s} - d_{j,s}}$ for $j < s$.

Also we assume that for $x \le d_{s,s}$, $\Gamma(A \sqcup B)(x)[s]$ is defined.

There are two steps at stage $s+1$:

Step 1. Consider the enumeration of c_{s+1} into $A \sqcup B$. There are three cases.

Case 1: No number is enumerated into $A_{s+1} \sqcup B_{s+1}$, i.e. c_{s+1} does not exist.

In this case, we let $D_{s+1} = D_s$ and extend $\overline{D}_{s+1} = \overline{D}_s \cup \{d_{s,s} + 1\}$, i.e. let $d_{s+1,s+1} = d_{s,s} + 1$. Extend $\sigma_{j,s}$ to $\sigma_{j,s}{}^\frown 1$ for all $j < s$ and extend $\sigma_{s,s}$ by $\sigma_{s,s}{}^\frown 1$ and $\sigma_{s,s}{}^\frown 0$, to form tree T_{s+1}. Also extend the definition of Γ to include Γ $(A \sqcup B)(d_{s+1,s+1})[s+1] = 0$ with use $d_{s+1,s+1}$ and set $g(j, s+1) = g(j, s)$ for all $j \le s$ and $g(s+1, s+1) = s+1$.

Case 2: c_{s+1} is enumerated into B_{s+1}. Let $p = c_{s+1}$, for simplicity.

In this case, we *re-route the construction through* $\sigma_{p,s}{}^\frown 1^*$, which means that enumerate all $d_{k,s}$ with $p \le k \le s$ into D_{s+1}. Let

$$\overline{D}_{s+1} = \{d_{0,s}, \ldots, d_{p-1,s}, d_{s,s} + 1, d_{s,s} + 2, \ldots, d_{s,s} + (s + 1 - p)\}.$$

That is, for k with $p \le k \le s+1$, $d_{k,s+1} = d_{s,s} + (k + 1 - p)$.

Terminate all leaves on T_s extending $\sigma_{p,s}{}^\frown 0$. We form T_{s+1} as follows: the rightmost path of T_{s+1}, i.e. $\sigma_{s+1,s+1}$, is $(\sigma_{p,s} \restriction d_{p,s}){}^\frown 1^{d_{s,s} - d_{p,s}}{}^\frown 0^{d_{s+1,s+1} - d_{s,s}}$, and the other paths of T_{s+1} are formed in the following pattern: $(\sigma_{s+1,s+1} \restriction d_{k,s+1}){}^\frown 1^{d_{s+1,s+1} - d_{k,s}}$ for $k \le s+1$.

Find the requirement $\mathcal{R}_e$ (with the highest priority) whose previous action at i say, is invalidated by the enumeration of p, if any, and declare that is not satisfied yet. Check whether there is a requirement $\mathcal{R}_{e'}$ at some j defers to $\mathcal{R}_e$ at i. If yes, then find $\mathcal{R}_{e'}$ (at j') with the highest priority, let it act, and defer those $\mathcal{R}_{e''}$ at j'' to $\mathcal{R}_{e'}$ at j' where $e'' > e'$ and $p < j'' < j'$. Correspondingly, for these j'', define $g(j'', s+1) = j'$ and define $\Gamma(A \sqcup B)(d_{j'',s+1})[s+1] = D(d_{j'',s+1})[s+1]$ with use bigger than $\sigma(j,s)$. Also define $\Gamma(A \sqcup B)(x)[s+1]$ for other $x < s+1$ if it has not definition, according to the rule of Γ.

Case 3: c_{s+1} is enumerated into A_{s+1}. Let $p = c_{s+1}$, for simplicity.

In this case, we find $\mathcal{R}_e$ with the highest priority that can act at i where $i > p$, if any, and let $\mathcal{R}_e$ act and declare that $\mathcal{R}_e$ is satisfied.

If so, then $d_{i,s}, \ldots, d_{s,s}$ are enumerated into D, and consequently,

$$\overline{D}_{s+1} = \{d_{0,s}, \ldots, d_{i-1,s}, d_{s,s}+1, d_{s,s}+2, \ldots, d_{s,s}+(s+1-i)\}.$$

That is, for k with $i \le k \le s+1$, $d_{k,s+1} = d_{s,s} + (k+1-p)$.

For T, we *re-route the construction through* $\sigma_{i,s}{}^\frown 1^*$, and terminate all leaves on T_s extending $\sigma_{i,s}{}^\frown 0$. We form T_{s+1} as follows: the rightmost path of T_{s+1}, i.e. $\sigma_{s+1,s+1}$, is $(\sigma_{i,s} \restriction d_{i,s})^\frown 1^{d_{s,s}-d_{i,s}}{}^\frown 0^{d_{s+1,s+1}-d_{s,s}}$, and the other paths of T_{s+1} are formed in the following pattern: $(\sigma_{s+1,s+1} \restriction d_{k,s+1})^\frown 1^{d_{s+1,s+1}-d_{k,s}}$ for $k \le s+1$.

For those requirements $\mathcal{R}_{e'}$, $e' > e$, that are active at $i' > i$ say, defer $\mathcal{R}_{e'}$ at i' to $\mathcal{R}_e$ at i, and we define $g(i', s+1) = i$. Define $\Gamma(A \sqcup B)(d_{j',s+1})[s+1] = D(d_{j',s+1})[s+1]$ with use bigger than $\sigma(i,s)$. Also define $\Gamma(A \sqcup B)(x)[s+1]$ for other $x < s+1$ if it has not been defined, according to the rule of Γ.

Step 2. For each $e \le s+1$, find whether $\mathcal{R}_e$ is active at some i, i.e. via $\sigma(i, s+1)$, $i \le s+1$. If yes, declare that $\mathcal{R}_e$ requests for A-permission below i to act.

End of stage $s+1$

End of construction

7. Verification

We now verify that the constructed c.e. set D and tree T satisfy all requirements. It is obvious that D is a c.e. set and T is a computable tree.

Lemma 7.1. *For all i, the coding of B can change $d_{i,-}$ at most finitely many times.*

Proof. It is obvious, if $d_{i,s} \ne d_{i,s+1}$, and it is because of the coding of B, then B must have a change on $j \le i$ and hence the coding of B can change $d_{i,-}$ at most $i+1$ many times. □

Lemma 7.2. *For each e:*

(1) *$\mathcal{R}_e$ has at most finitely many permanently active i's, and $\mathcal{R}_e$ is satisfied.*
(2) *If the construction of T cannot be eventually re-routed into a cone with empty intersection with P_e, then any deferring being made by $\mathcal{R}_e$ will be released after finitely many stages.*

Proof. We prove it by induction on e. Suppose that (1) and (2) are true for all $e' < e$, we will prove (1) and (2) are also true for $\mathcal{R}_e$.

Let s_0 be the least stage after which no $\mathcal{R}_{e'}$, $e' < e$, can be permanently active at more i's. By induction hypothesis, we can have a string σ' in T such that after stage s_0, σ' is always on the rightmost path on T. Thus after stage s_0, $\mathcal{R}_e$ works only above σ'. Note that at stage s_0, $\mathcal{R}_e$ at some i can be deferred to some $\mathcal{R}_{e'}$, $e' < e$, at some j with $j > i$. Let i be the largest such a number, and without loss of generality, assume that after stage s_0, A has no changes below i. According to our assumption on σ', these deferrings will keep going on forever.

We first show that (1) is true for $\mathcal{R}_e$. At any further stage s, we will check whether some $\sigma_{j,s}$, $j > i$, is not in $P_{e,s}$, and if any, check whether A has permissions to allow the construction of T to re-route through $\sigma_{j,s}$. Remember that before the construction part, we explained about how to construct a p.r. functional Δ: when $\Delta(j)$ is defined at a stage t, we should have seen some $\sigma_{j',t}$, $(j' > j)$, is not in $P_{e,t}$, and the change of $D(j)$ will actually provide a permission for the re-routing through $\sigma_{j',t}$. Note that this re-routing action could be deferred by some $\mathcal{R}_{e'}$, $e' < e$, and by our assumption of stage s_0, we know that this deferring will be released after finitely many stages, and $\mathcal{R}_e$ will eventually succeed in performing the re-routing action, if B does not change to invalidate this. This shows that we either satisfy $\mathcal{R}_e$ by re-routing the construction of T to a cone with empty intersection with $[P_e]$, or we force a change of B below j', allowing us to redefine $\Delta(B)(j)$. If $\mathcal{R}_e$ has infinitely many permanent active k's, then as discussed before, $\Delta(B)$ would be total and computes A correctly. As $A >_T B$, $\Delta(B)$ cannot be total and computes A correctly. This shows the existence of $j > i$ such that either $\Delta(B)(j)$ is defined but with $\Delta(B)(j) \neq A(j)$, or $\Delta(B)(j)$ is not defined at all. In the former case, $\mathcal{R}_e$ acts by re-routing the construction of T to a cone. As B has no change below j', i.e. below $\delta(j)$, T will work in this cone forever and $\mathcal{R}_e$ is met. In the latter case, there are two possibilities. One possibility is that after a certain stage w, $\Delta(B)(j)$ becomes undefined and cannot have definition at any further stage. This means that after some stage, $\mathcal{R}_e$ cannot be active at any string above $\sigma_{j',w}$, and hence strings above $\sigma_{j',w}$ will be always in P_e, showing that $[T]\backslash[P_e]$ is finite, and $\mathcal{R}_e$ is met. The other possibility is that $\mathcal{R}_e$ can be active at infinitely many k's. As we already know that $\mathcal{R}_e$ can be permanently active at only finitely many k's, we know that the changes of B will invalidate almost all of these, and hence the rightmost path in $[T]$ is in $[P_e]$, and hence, $[T]\backslash[P_e]$ is finite, so $\mathcal{R}_e$ is met.

Now we prove (2), by showing that at any stage $s > s_0$, $\mathcal{R}_e$ may wait for A-permission to act at j, i.e. to re-route the construction of T through $\sigma_{j,s}$, and when A has a permission at stage t, some $\mathcal{R}_{e'}$, with $e' < e$, at $j' > j$, may defer $\mathcal{R}_e$ at j. According to our assumption on s_0, this deferring will be released later by a B-change, and we can show that after a stage big enough, all of these deferrings will be released, allowing $\mathcal{R}_e$ to re-route the construction of T through $\sigma_{j,s}$. This action may also defer other $\mathcal{R}$-strategies with lower priority, and if the construction of T eventually comes back to P_e, $\mathcal{R}_e$'s action at j will be invalidated by changes of B, which means that this deferring to other lower priority $\mathcal{R}$-requirements will be released eventually. (2) is true.

This completes the proof of Lemma 7.2. □

Lemma 7.3. *For all $i, \lim_s d_{i,s} = d_i$ exists. As a consequence, $B \leq_T D$, as we specified before by constructing a p.r. functional Θ.*

Proof. Fix i. We have seen in Lemma 7.1 that every change of B below i also changes $d_{i,-}$. Another way of having $d_{i,s} \neq d_{i,s+1}$ is due to some actions of $\mathcal{R}_e$ with $e < i$ by re-routing the construction of T to a cone above $\sigma_{j,s}$ with $j \leq i$. This action is initiated by A-permissions below j, and hence below i (the re-routing actions can be delayed by some $\mathcal{R}_{e'}$ for awhile (maybe forever) where $e' < e$). So once A has no more changing below $(i+1)$, after stage s say, there is no more possibility of initiating further enumeration of $\sigma_{i,-}$. That is, all the possible changes of $\sigma_{i,-}$ are initiated by stage s. This means that $\sigma_{i,-}$ can change at most $i+1$ many times by actions of $\mathcal{R}$-requirements during the construction.

There is no other way of changing $\sigma_{i,-}$. Hence, together with Lemma 7.1, $\sigma_{i,-}$ changes only finitely many times, and has a limit.

Immediately, by the description of Θ we mentioned before, $B \leq_T D$. □

Lemma 7.4. $\Gamma(A \sqcup B)$ *is total and* $D = \Gamma(A \sqcup B)$.

Proof. We prove the lemma by induction on x.

Fix x and assume that for any $y < x$, $\Gamma(A \sqcup B)(y)$ is defined with $\Gamma(A \sqcup B)(y) = D(y)$. We need to show that $\Gamma(A \sqcup B)(x)$ is defined with $\Gamma(A \sqcup B)(x) = D(x)$.

Note that in the construction, when we define $\Gamma(A \sqcup B)(x)$ for the first time, at a stage s say, and x is already in D_s, then we just define $\Gamma(A \sqcup B)(x) = 1$ with use $\gamma(x)[s] = \gamma(x-1)[s]$, and keep $\gamma(x)$ the same as $\gamma(x-1)$. So once $\gamma(x-1)$ becomes fixed, $\gamma(x)$ is also fixed.

So we assume that when $\Gamma(A \sqcup B)(x)$ is defined for the first time at stage s, $x \notin D_s$. Then $x = d_{i,s}$ for some $i \leq s$, and the A-part use and B-part use are defined as $i+1$. When $\Gamma(A \sqcup B)(x)$ becomes undefined during the construction, due to changes of A or B below the use, we choose another use $\gamma(x)$ for $\Gamma(A \sqcup B)(x)$ only when A changes below $i+1$. [*If B changes below $i+1$ first, we just keep the use the same.*]

Suppose that A changes below $i+1$ at stage s' say. If at this stage, no $\mathcal{R}_e$-requirements, $e \leq i$, acts by re-routing the construction of T via $\sigma_{j,s'}$, where $j \geq i$, then just keep use the same, for both A-part and B-part. Otherwise, without loss of generality, we assume that $\mathcal{R}_{e_2}$ at j_2 defers to $\mathcal{R}_{e_1}$ at j_1, with $e_1 < e_2$ and $j_2 \leq i \leq j_1$. Then we keep the A-part of the use as $i+1$, but increase the B-part of the use as j_1. If A changes below $i+1$ or B changes below j_1, then $\Gamma(A \sqcup B)(x)$ becomes undefined again, and we check whether x is enumerated into D at this stage, and if not, we check whether the deferring has been shifted upward or downward or extended to an $\mathcal{R}$-requirements with even higher priority. In the former case, as x is now in D, we just define $\gamma(x)[s+1]$ the same as $\gamma(x)[s]$. In the latter case, the A-part of the use will be kept the same, i.e. $i+$, and the B-part of the use will be changed to an even bigger number, depending on whether the deferring is extended (the B-part of the use will be kept the same), or shifted upward or downward (the B-part of the use could be increased). For this i, only requirements $\mathcal{R}_e$, with $e < i$, can initiate these deferrings and for each e, after $\mathcal{R}_e$ initiates such a deferring at j, $\mathcal{R}_e$ can have deferrings at j' only when $j' < j$ [*we can explain this in terms of function* $g(i, t)$], and hence, for this i, this deferring can happen at most finitely many times, and $\gamma(x)$ will have a fixed definition after these stages, $\Gamma(A \sqcup B)(x)$ is defined. In this process, x can be enumerated into D only when A changes below $i + 1$ or changes of B release the deferrings after several steps, and at each such a step, $\Gamma(A \sqcup B)(x)$ is undefined, and x can be enumerated into D only at these steps.

This shows that $\Gamma(A \sqcup B)(x)$ is defined everywhere and hence is a total function, and also $\Gamma(A \sqcup B)(x) = D(x)$.

By induction, $\Gamma(A \sqcup B)$ is a total function and $\Gamma(A \sqcup B) = D$. $\square$

This completes the proof of Theorem 1.4.

Acknowledgments

Downey's research is partially supported by the Marsden Fund New Zealand.

Wu's research is partially supported by Singapore Ministry of Education Academic Research Fund Tier 2 Grant MOE2016-T2-1-083 (M4020333) and NTU Tier 1 Grants RG29/14 (M4011277) and RG32/16 (M4011672).

Yang's research is partially supported by the Singapore Ministry of Education Academic Research Fund Tier 2 Grant MOE2016-T2-1-019/R146-000-234-112 and NUS Research Grant WBS R-146-000-231-114 and NSSFC 17BZX089 (China).

References

[1] K. Ambos-Spies, D. Hirschfeldt and R. A. Shore, Undecidability and 1-types in intervals of the computably enumerable degrees, *Ann. Pure Appl. Logic* **106** (2000) 1–47.

[2] D. Cenzer, Π_1^0 classes in computability theory, in *Handbook of Computability Theory* (Elsevier, Amsterdam, 1999), pp. 37–85.

[3] D. Cenzer, P. Clote, R. Smith, R. Soare and S. Wainer, Members of countable Π_1^0 classes, *Ann. Pure Appl. Logic* **31** (1986) 145–163.

[4] D. Cenzer, R. Downey, C. G. Jockusch, Jr. and R. A. Shore, Countable thin Π_1^0 classes, *Ann. Pure Appl. Logic* **59** (1993) 79–139.

[5] D. Cenzer and J. B. Remmel, Π_1^0 classes in mathematics, *J. Symbolic Logic* **54** (1989) 975–991.

[6] D. Cenzer and J. B. Remmel, *Effectively Closed Sets*, Perspective in Logic (Cambridge Univ. Press, to appear).

[7] D. Cenzer and R. Smith, The ranked points of a Π_1^0 set, *J. Symbolic Logic* **54** (1989) 975–991.

[8] P. Cholak, R. Coles, R. Downey and E. Herrmann, Automorphisms of the lattice of Π_1^0 classes: Perfect thin classes and ANC degrees, *Trans. Amer. Math. Soc.* **353** (2001) 4899–4924.

[9] P. Cholak and R. Downey, On the Cantor–Bendixon rank of recursively enumerable sets, *J. Symbolic Logic* **58** (1993) 629–640.

[10] R. Downey, Maximal theories, *Ann. Pure Appl. Logic* **33** (1987) 245–282.

[11] R. Downey, C. G. Jockusch and M. Stob, Array nonrecursive sets and multiple permitting arguments, in *Recursion Theory Week*, eds. K. Ambos-Spies, G. H. Müller and G. E. Sacks, Lecture Notes in Mathematics, Vol. 1432 (Springer-Verlag, Heidelberg, 1990), pp. 141–174.

[12] R. Downey and A. Montalban, Slender classes, *J. London Math. Soc.* **78** (2008) 36–50.

[13] R. Downey, G. Wu and Y. Yang, The members of thin and minimal Π^0_1 classes, their ranks and Turing degrees, *Ann. Pure Appl. Logic* **166** (2015) 755–766.

[14] P. Fejer, The density of the nonbranching degrees, *Ann. Pure Appl. Logic* **24** (1983) 113–130.

[15] H. Friedman, S. Simpson and R. Smith, Countable algebra and set existence axioms, *Ann. Pure Appl. Logic* **25** (1983) 141–181.

[16] C. G. Jockusch, Jr. and R. I. Soare, Π^0_1 classes and degrees of theories, *Trans. Amer. Math. Soc.* **173** (1972) 33–56.

[17] C. G. Jockusch, Jr. and R. I. Soare, Degrees of members of Π^0_1 classes, *Pacific J. Math.* **40** (1972) 605–616.

[18] D. A. Martin and M. B. Pour-El, Axiomatizable theories with few axiomatizable extensions, *J. Symbolic Logic* **35** (1970) 205–209.

[19] G. Metakides and A. Nerode, Effective content of field theory, *Ann. Math. Logic* **17** (1979) 289–320.

[20] T. A. Slaman, The density of infima in the recursively enumerable degrees, *Ann. Pure Appl. Logic* **52** (1991) 155–179.

[21] R. I. Soare, Automorphisms of the lattice of recursively enumerable sets. Part I: Maximal sets, *Ann. Math.* **100** (1974) 80–120.

[22] R. I. Soare, *Recursively Enumerable Sets and Degrees* (Springer-Verlag, Berlin, 1987).