A Minimal Set Low for Speed

Rod Downey Matthew Harrison-Trainor

November 18, 2020

Abstract

An oracle A is low-for-speed if it is unable to speed up the computation of a set which is already computable: if a decidable language can be decided in time $t(n)$ using A as an oracle, then it can be decided without an oracle in time $p(t(n))$ for some polynomial p. The existence of a set which is low-for-speed was first shown by Bayer and Slaman who constructed a non-computable computably enumerable set which is low-for-speed. In this paper we answer a question previously raised by Bienvenu and Downey, who asked whether there is a minimal degree which is low-for-speed. The standard method of constructing a set of minimal degree via forcing is incompatible with making the set low-for-speed; but we are able to use an interesting new combination of forcing and full approximation to construct a set which is both of minimal degree and low-for-speed.

1 Introduction

Almost since the beginning of computational complexity theory, we have had results about oracles and their effect on the running times of computations. For example Baker, Gill, and Solovay [BGS75] showed that on the one hand there are oracles A such that $\mathsf{P}^A = \mathsf{NP}^A$ and on the other hand there are oracles B such that $\mathsf{P}^B \neq \mathsf{NP}^B$, thus demonstrating that methods that relativize will not suffice to solve basic questions like P vs NP. An underlying question is whether oracle results can say things about complexity questions in the unrelativized world. The answer seems to be yes. For example, Allender together with Buhrman and Koucký [ABK06] and with Friedman and Gasarch [AFG13] showed that oracle access to sets of random strings can give insight into basic complexity questions. In [AFG13] Allender, Friedman, and Gasarch showed that $\bigcap_U \mathsf{P}^{R_{K_U}} \cap \mathsf{COMP} \subseteq \mathsf{PSPACE}$ where R_{K_U} denotes the strings whose prefix-free Kolmogorov complexity (relative to universal machine U) is at least their length, and COMP denotes the collection of computable sets. Later the "$\cap\mathsf{COMP}$" was removed by Cai, Downey, Epstein, Lempp, and J. Miller [CDE+14]. Thus we conclude that reductions to very complex sets like the random strings somehow gives insight into very simple things like computable sets.

One of the classical notions in computability theory is that of lowness. An oracle is low for a specific type of problem if that oracle does not help to solve that problem. A language A is low if the halting problem relative to A has the same Turing degree (and hence the same computational content) as the halting problem. Slaman and Solovay [SS91] characterized languages L where oracles are of no help in Gold-style learning theory: $EX^L = EX$ iff L

is low and 1-generic. Inspired by this and other lowness results in classical computability, Allender asked whether there were non-trivial sets which were “low for speed” in that, as oracles, they did not accelerate running times of computations by more than a polynomial amount. Of course, as stated this makes little sense since using any X as oracle, we can decide membership in X in linear time, while without an oracle X may not even be computable at all! Thus, what we are really interested in is the set of oracles which do not speed up the computation of *computable sets* by more than a polynomial amount. More precisely, an oracle X is *low for speed* if for any computable language L, if some Turing machine M with access to oracle X decides L in time f, then there is a Turing machine M' without any oracle and a polynomial p such that M' decides $\mathcal{L}$ in time $p \circ f$. (Here the computation time of an oracle computation is counted in the usual complexity-theoretic fashion: we have a query tape on which we can write strings, and once a string x is written on this tape, we get to ask the oracle whether x belongs to it in time $O(1)$.)

There are trivial examples of such sets, namely oracles that belong to P, because any query to such an oracle can be replaced by a polynomial-time computation. Allender's precise question was therefore:

Is there an oracle $X \notin \mathsf{P}$ which is low for speed?

Such an X, if it exists, has to be non-computable, for the same reason as above: if X is computable and low for speed, then X is decidable in linear time using oracle X, thus—by lowness—decidable in polynomial time without oracle, i.e., $X \in \mathsf{P}$.

A partial answer was given by Lance Fortnow (unpublished), who observed the following.

Theorem 1.1 (Fortnow)**.** *If X is a hypersimple and computably enumerable oracle, then X is low for polynomial time, in that if $L \in \mathsf{P}^X$ is computable, then $L \in \mathsf{P}$.*

Allender's question was finally solved by Bayer and Slaman, who showed the following.

Theorem 1.2 (Bayer-Slaman [Bay12])**.** *There are non-computable, computably enumerable, sets X which are low for speed.*

Bayer showed that whether 1-generic sets were low for speed depended on whether $\mathsf{P} = \mathsf{NP}$. In [BD18], Bienvenu and Downey began an analysis of precisely what kind of sets/languages could be low for speed. The showed for instance, randomness always accelerates some computation in that no Schnorr random set is low for speed. They also constructed a perfect Π^0_1 class all of whose members were low for speed. Among other results, they demonstrated that being low for speed did not seem to align very well to having low complexity in that no set of low computably enumerable Turing degree could also be low for speed.

From one point of view the sets with barely non-computable information are those of minimal Turing degree. Here we recall that $\mathbf{a}$ is a *minimal* Turing degree if it is nonzero and there is no degree $\mathbf{b}$ with $\mathbf{0} < \mathbf{b} < \mathbf{a}$. It is quite easy to construct a set of minimal Turing degree which is not low for speed, and indeed any natural minimality construction seems to give this. That is because natural Spector-forcing style dynamics seem to entail certain delays in any construction, even a full approximation one, which cause problems with the polynomial time simulation of the oracle computations being emulated. In view of this, Bienvenu and Downey asked the following question:

Question 1.3. Can a set A of minimal Turing degree be low for speed?

In the present paper we answer this question affirmatively:

Theorem 1.4. *There is a set A which is both of minimal Turing degree and low for speed.*

The construction is a mix of forcing and full approximation of a kind hitherto unseen. The argument is a complicated priority construction in which the interactions between different requirements is quite involved. In order to make the set A of minimal Turing degree, we must put it on splitting trees; and in order to make it low-for-speed, we must have efficient simulations of potential computations involving A. When defining the splitting trees, we must respect decisions made by our simulations, which restricts the splits we can choose. The splitting trees end up having the property that while every two paths through the tree split, two children of the same node may not split; finding splits is sometimes very delayed. This is a new strategy which does not seem to have been used before for constructing sets of minimal degree.

2 The construction with few requirements

We will construct a set A meeting the following requirements:

$\mathcal{M}_e$: If Φ_e^A is total then it is either computable or computes A.

$\mathcal{L}_{\langle e,i\rangle}$: If $\Psi_e^A = R_i$ is total and computable in time $t(n)$, then it is computable in time $p(t(n))$ for some polynomial p.

$\mathcal{P}_e$: $A \neq W_e$.

Here, R_i is a partial computable function. The requirements $\mathcal{P}_e$ make A non-computable, while the requirements $\mathcal{M}_e$ make A of minimal degree. The requirements $\mathcal{L}_{\langle e,i\rangle}$ make A low for speed.

When working with Spector-style forcing, it is common to define a tree to be a map $T\colon 2^{<\omega} \to 2^{<\omega}$ such that $\sigma \preceq \tau$ implies $T(\sigma) \preceq T(\tau)$. We will need our trees to be finitely branching; so for the purposes of this proof a tree will be a computable subset of $2^{<\omega}$ so that each node σ on the tree has finitely many children $\tau \succeq \sigma$. The children of σ may be of any length, where by length we mean the length as a binary string. Our trees will have no dead ends, and in fact every node will have at least two children. As usual, $[T]$ denotes the collection of paths through T. Recall that for a functional Φ_e, we will say that T is *e-splitting* if for any two distinct paths π_1 and π_2 through T, there is x with

$$\Phi_e^{\pi_1}(x)\downarrow \neq \Phi_e^{\pi_2}(x)\downarrow .$$

If τ_1 and τ_2 are initial segments of π_1 or π_2 respectively witnessing this, i.e., with

$$\Phi_e^{\tau_1}(x)\downarrow \neq \Phi_e^{\tau_2}(x)\downarrow,$$

and with a common predecessor σ, we say that they e-split over σ, or that they are an e-split over σ. The requirements $\mathcal{M}_e$ will be satisfied by an interesting new mix of forcing and full approximation. Following the standard Spector argument, to satisfy $\mathcal{M}_e$ we attempt to make A a path on a tree T with either:

- T is e-splitting, and so for any path $B \in [T]$, $\Phi_e^B \geq_T B$; or
- for all paths $B_1, B_2 \in [T]$ and all x, if $\Phi_e^{B_1}(x)\downarrow$ and $\Phi_e^{B_2}(x)\downarrow$ then $\Phi_e^{B_1}(x) = \Phi_e^{B_2}(x)$, and so Φ_e^B is either partial or computable for any $B \in [T]$.

Given such a tree, any path on T satisfies $\mathcal{M}_e$.

The standard argument for building a minimal degree is a forcing argument. Suppose that we want to meet just the $\mathcal{M}$ and $\mathcal{P}$ requirements. We can begin with a perfect tree T_{-1}, say $T_{-1} = 2^{<\omega}$. Then there is a computable tree $T_0 \subseteq T_{-1}$ which is either 0-splitting or forces Φ_0^A to be either computable or partial. We can then choose $A_0 \in T_0$ such that A_0 is not an initial segment of W_e. Then there is a computable tree $T_1 \subseteq T_0$ with root A_0 which is either 1-splitting or forces Φ_1^A to be either computable or partial. We pick $A_1 \in T_1$ so that A_1 is not an initial segment of W_1, then $T_2 \subseteq T_1$ with root A_1, and so on. Then $A = \bigcup A_i$ is a path through each T_i, and so is a minimal degree. Though each T_i is computable, they are not uniformly computable; given T_i, to compute T_{i+1} we must know whether T_{i+1} is to be $(i+1)$-splitting, to force partiality, or to force computability.

We cannot purely use forcing the meet the lowness requirements $\mathcal{L}_{\langle e,i \rangle}$. We use something similar to the Slaman-Beyer strategy from [Bay12]. The entire construction will take place on a tree T_{-1} with the property that it is polynomial in $|\sigma|$ to determine whether $\sigma \in T_{-1}$, and that moreover, for each n, there are polynomially many in n strings of length n on T_{-1}. For example, let $\sigma \in T_{-1}$ if it is of the form

$$a_1^{2^0} a_2^{2^1} a_3^{2^2} a_4^{2^3} \cdots$$

where each $a_i \in \{0,1\}$. So, for example, $100111100000000 \in T_{-1}$.

First we will show how to meet $\mathcal{L}_{\langle e,i \rangle}$ in the absence of any other requirements. For simplicity drop the subscripts e and i so that we write $\Psi = \Psi_e$ and $R = R_i$. The idea is to construct a computable simulation Ξ of Ψ^A, with $\Xi(x)$ computable in time polynomial in the running time of $\Psi^A(x)$, so that if $\Psi^A = R$ then $\Xi = \Psi^A$. We compute $\Xi(x)$ as follows. We computably search over $\sigma \in T_{-1}$ (i.e. over find potential initial segments σ of A) and simulate the computations $\Psi^\sigma(x)$. When we find σ with $\Psi^\sigma(x)\downarrow$, we set $\Xi(x) = \Psi^\sigma(x)$ for the first such σ. Of course, σ might not be an initial segment of A, and so Ξ might not be equal to Ψ^A; this only matters if $\Psi^A = R$ is total, as otherwise $\mathcal{L}$ is satisfied vacuously. If x is such that $\Xi(x)\downarrow \neq R(x)\downarrow$, then there is some $\sigma \in T_{-1}$ witnessing that $\Psi^\sigma(x) = \Xi(x)$; the requirement $\mathcal{L}$ asks that A extend σ, so that $\Psi^A(x) \neq R(x)$ and $\mathcal{L}$ is satisfied. So now we need to ensure that if $\Xi = \Psi^A = R$, then Ξ is only polynomially slower than Ψ^A. We can do this by appropriately dovetailing the simulations so that if $\Psi^\sigma(x)\downarrow$ in time $t(x)$, the simulation Ξ will test this computation in a time which is only polynomially slower than $t(x)$, and we will have $\Xi(x)\downarrow$ in time which is only polynomially slower than $t(x)$. For example, we might start by simulating one stage of the computation $\Psi^\sigma(x)$ for σs of length one, then simulating two stages for σs of length at most two, then three stages for σs of length at most three, and so on. It is important here that T_{-1} has only polynomially many nodes at height n and we can test membership in T_{-1} in polynomial time; so the nth round of simulations takes time polynomial in n.

Think of the simulations as being greedy and taking any computation that they find; and then, at the end, we can non-uniformly choose the initial segment of A to force that either the simulation is actually correct, or to get a diagonalization.

The interactions between the requirements get more complicated. Consider now two requirements $\mathcal{M} = \mathcal{M}_e$ and $\mathcal{L}$. If $\mathcal{L}$ is of higher priority than $\mathcal{M}$, there is nothing new going on—$\mathcal{M}$ knows whether $\mathcal{L}$ asked to have A extend some node σ, and if it did, $\mathcal{M}$ tries to build a splitting tree extending σ. So assume that $\mathcal{M}$ is of higher priority than $\mathcal{L}$.

Write $\Phi = \Phi_e$. Assume that for each $\sigma \in T_{-1}$, there are x and $\tau_1, \tau_2 \succeq \sigma$ such that $\Phi^{\tau_1}(x) \downarrow \neq \Phi^{\tau_2}(x) \downarrow$; and that for each $\sigma \in T_{-1}$ and x there is $\tau \succeq \sigma$ such that $\Phi^{\tau}(x) \downarrow$. Otherwise, we could find a subtree of T_{-1} which forces that Φ^A is either not total or is computable, and satisfy $\mathcal{M}$ by restricting to that subtree. This assumption implies that we can also find any finite number of extensions of various nodes that pairwise e-split, e.g. given σ_1 and σ_2, there are extensions of σ_1 and σ_2 that e-split. Indeed, find extensions τ, τ^* of σ_1 that e-split, say $\Phi^{\tau_1}(x) \downarrow \neq \Phi^{\tau_2}(x) \downarrow$, and an extension ρ of σ_2 with $\Phi^{\tau}(x) \downarrow$. Then ρ e-splits with one of τ_1 or τ_2.

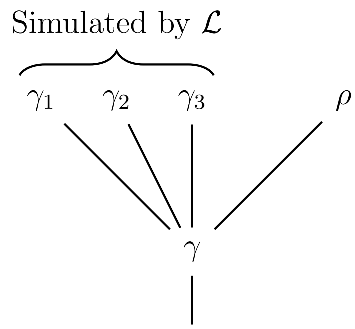
The requirement $\mathcal{L}$ non-uniformly guesses at whether or not $\mathcal{M}$ will succeed at building an e-splitting tree. Suppose that it guesses that $\mathcal{M}$ successfully builds such a tree. $\mathcal{M}$ begins with the special tree T_{-1} described above, and it must build an e-splitting tree $T \subseteq T_{-1}$.

While building the tree, Ξ will be simulating Ψ^A by looking at computations Ψ^σ. The tree T might be built very slowly, while Ξ has to simulate computations relatively quickly. So when a node is removed from T, Ξ will stop simulating it, but Ξ will have to simulate nodes which are extensions of nodes in T as it has been defined so far, but which have not yet been determined to be in or not in T. This leads to the following problem: Suppose that γ is a leaf of T at stage s, ρ extends γ, and Ξ simulates $\Psi^\rho(x)$ and sees that it converges, and so defines $\Xi(x) = \Psi^\rho(x)$. But then the requirement $\mathcal{M}$ finds an e-split $\tau_1, \tau_2 \succeq \gamma$ and wants to set τ_1 and τ_2 to be the successors of γ on T, with both τ_1 and τ_2 incompatible with ρ. If we allow $\mathcal{M}$ to do this, then since $\mathcal{M}$ has higher priority than $\mathcal{L}$, $\mathcal{M}$ has determined that A cannot extend ρ as $\mathcal{M}$ restricts A to be a path through T. So $\mathcal{L}$ has lost its ability to diagonalize and it might be that $\Psi^A = R$ (because this happens on all paths through T) but $\Psi^A(x) \neq \Psi^\rho(x) = \Xi(x)$.

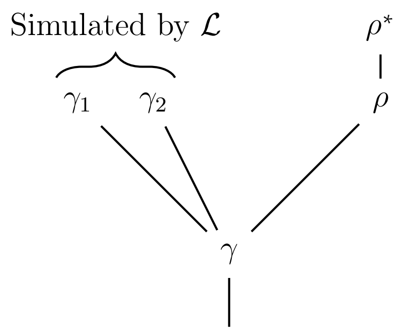
This means that $\mathcal{M}$ needs to take some action to keep computations that $\mathcal{L}$ has found on the tree. We begin by describing the most basic strategy for keeping a single node ρ on the tree.

Suppose that at stage s the requirement $\mathcal{M}$ wants to add children to a leaf node γ on T. First, look for $\gamma_1, \gamma_2, \gamma_3$ extending γ such that they pairwise e-split: for any two γ_i, γ_j, there is x such that $\Phi_e^{\gamma_i}(x) \downarrow \neq \Phi_e^{\gamma_j}(x) \downarrow$. By our earlier assumption that each node on T_{-1} has an e-splitting extension we will eventually find such elements, say at stage t. But it might be that by stage t, we have simulated $\Psi_t^\rho(x) \downarrow$ and set $\Xi(x)$ to be equal to this simulated computation. So for the sake of $\mathcal{L}$, we must keep ρ on the tree. (Later, we will have to extend the strategy to worry about what happens if we have simulated multiple computations ρ, but for now assume that there is just one.)

To begin, we stop simulating any computations extending ρ. This means that we are now free to extend the tree however we like above ρ without worrying about how this affects the simulations. We also stop simulating any other computations not compatible with γ_1, γ_2, or γ_3.



Now look for an extension ρ^* of ρ that e-splits with at least two of γ_1, γ_2, and γ_3. We can find such a ρ^* by looking for one with Φ^{ρ^*} defined on the values x witnessing the e-splitting of γ_1, γ_2, and γ_3, e.g., if $\Phi^{\gamma_1}(x) \neq \Phi^{\gamma_2}(x)$, and $\Phi^{\rho^*}(x)\downarrow$, then ρ^* must e-split with either γ_1 or γ_2. Say that ρ^* e-splits with γ_1 and γ_2. Then we define the children of γ to be γ_1, γ_2, and ρ^*.

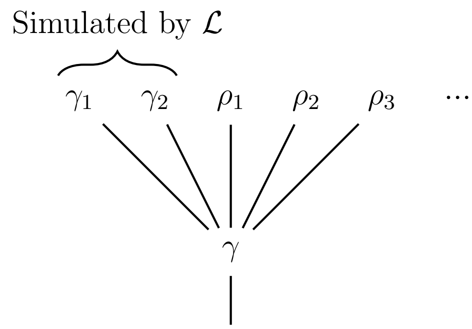


So of the extensions of γ, some are simulated by $\mathcal{L}$, and others are not. If $\mathcal{L}$ has the infinitary outcome, where it never finds the need to diagonalize, then it will have A extend either γ_1 or γ_2. It is only if $\mathcal{L}$ needs to diagonalize that it will have A extend ρ^*—and in this case, $\mathcal{L}$ is satisfied and so does not have to simulate Ψ^A.

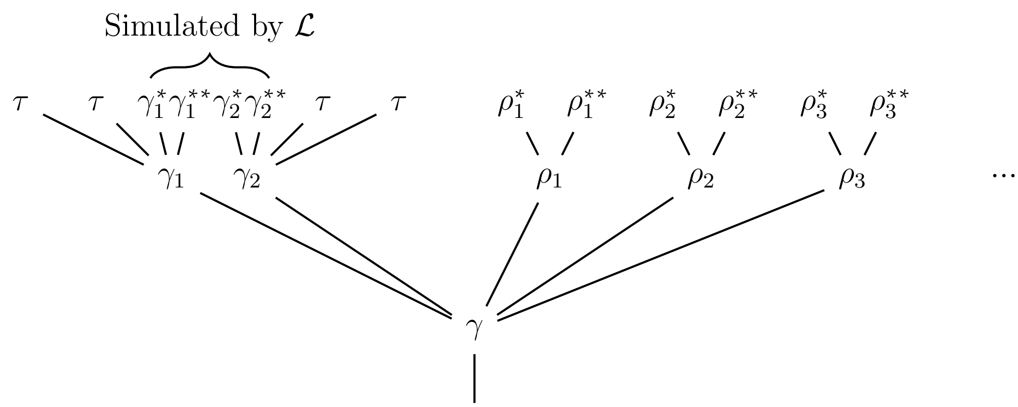
There is still an issue here. What if, while looking for ρ^*, we simulate a computation $\Psi^{\gamma_3}(y)\downarrow$, and set $\Xi(y) = \Psi^{\gamma_3}(y)$, and then only after this find that ρ^* e-splits with γ_1 and γ_2? We can no longer remove γ_3 from the tree. Moreover, there might be many different nodes ρ that we cannot remove from the tree—indeed, it might be that around stage s, we cannot remove any nodes at height s from the tree, because each of them has some computation that we have simulated.

To deal with this, we have to build e-splitting trees in a weaker way. It will no longer be the case that every pair of children of a node σ e-split, but we will still make sure that every pair of paths e-splits. (It might seem that this violates compactness, but in fact thinking more carefully it does not—the set of pairs of paths (π_1, π_2) that e-split is an open cover of the non-compact topological space $[T] \times [T] - \Delta$ where Δ is the diagonal.)

So suppose again that we are trying to extend γ. Look for a pair of nodes γ_1, γ_2 that e-split. Suppose that $\rho_1, \ldots, \rho_n$ are nodes that have been simulated, that we must keep on the tree. (We might even just assume that $\rho_1, \ldots, \rho_n$ are all of the other nodes at the same level as γ_1, γ_2.) We stop simulating computations above $\rho_1, \ldots, \rho_n$.



Now at the next step we need to add extensions to γ_1 and γ_2 just as we added extensions of γ. We look for extensions γ_1^* and γ_1^{**} of γ_1, γ_2^* and γ_2^{**} of γ_2, and ρ_i^* and ρ_i^{**} of ρ_i such that all of these pairwise e-split. While we are looking for these, we might simulate more computations at nodes τ above γ_1 and γ_2, but there will be no more computations simulated above the ρ_i.



Now at the next step of extending the tree we need to extend γ_1^*, γ_1^{**}, γ_2^*, and γ_2^{**}, and make sure that we extend τ to e-split with these extensions and with extensions of the ρ^*; but in doing so we will introduce further extensions that do not e-split. So at no finite step do we get that everything e-splits with each other, but in the end every pair of paths e-splits.

3 Multiple requirements and outcomes

Order the requirements $\mathcal{M}_e$, $\mathcal{L}_e$, and $\mathcal{P}_e$ as follows, from highest priority to lowest:

$$\mathcal{M}_0 > \mathcal{L}_0 > \mathcal{P}_0 > \mathcal{M}_1 > \mathcal{L}_1 > \mathcal{P}_1 > \mathcal{M}_2 > \cdots.$$

Each requirement has various possible outcomes:

- A requirement $\mathcal{M}_e$ can either build an e-splitting tree, or it can build a tree forcing that Φ_e is either partial or computable. In the former case, when $\mathcal{M}_e$ builds an e-splitting tree, we say that $\mathcal{M}_e$ has the infinitary outcome ∞. In the latter case, there is a node σ above which we do not find any more e-splittings. We say that $\mathcal{M}_e$ has the finitary outcome σ.

- A requirement $\mathcal{L}_{\langle e,i\rangle}$ can either have the simulation Ξ of Ψ_e be equal to R_i whenever they are both defined, or $\mathcal{L}_{\langle e,i\rangle}$ can force A to extend a node σ, with $\Psi_e^\sigma(x) \neq R_i(x)$ for some x. In the first case, we say that $\mathcal{L}_{\langle e,i\rangle}$ has the infinitary outcome ∞, and in the latter case we say that $\mathcal{L}_{\langle e,i\rangle}$ has the finitary outcome σ.

- A requirement $\mathcal{P}_e$ chooses an initial segment σ of A that ensures that A is not equal to the eth c.e. set W_e. This node σ is the outcome of $\mathcal{P}_e$.

The *tree of outcomes* is the tree of finite strings η where $\eta(3e)$ is an outcome for $\mathcal{M}_e$, $\eta(3e+1)$ is an outcome for $\mathcal{L}_e$, and $\eta(3e+2)$ is an outcome for $\mathcal{P}_e$, and so that η satisfies the coherence condition described below. For convenience, given a requirement $\mathcal{R}$ we write $\eta(\mathcal{R})$ for the outcome of $\mathcal{R}$ according to η: $\eta(\mathcal{M}_e) = \eta(3e)$, $\eta(\mathcal{L}_e) = \eta(3e+1)$, and $\eta(\mathcal{P}_e) = \eta(3e+2)$. Using this notation allows us to avoid having to remember exactly how we have indexed the entries of η. Given a requirement $\mathcal{R}$, we say that η is a *guess by* $\mathcal{R}$ if η has an outcome for each requirement of higher priority than $\mathcal{R}$, e.g. a guess by $\mathcal{L}_e$ is a string η of length $3e+1$ with

$$\eta = \langle \eta(\mathcal{M}_0), \eta(\mathcal{L}_0), \eta(\mathcal{P}_0), \dots, \eta(\mathcal{M}_{e-1}), \eta(\mathcal{L}_{e-1}), \eta(\mathcal{P}_{e-1}), \eta(\mathcal{M}_e)\rangle.$$

We put one more extension condition on these guesses. Define $\mathrm{root}(\varnothing) = \varnothing$. Given an outcome η, define $\mathrm{root}(\eta\hat{\ }\sigma)$ inductively by $\mathrm{root}(\eta\hat{\ }\sigma)$ to be $\mathrm{root}(\eta)$ if $\sigma = \infty$, or σ otherwise; and we require that $\sigma \succeq \mathrm{root}(\eta)$ for $\eta\hat{\ }\sigma$ to be an acceptable guess. The node $\mathrm{root}(\eta)$ is the non-uniform information η has about an initial segment of the set A we are building. (The guesses which might actually be the true outcomes will also satisfy other conditions, for example the nodes in the guess will actually be on the trees built by higher priority requirements.)

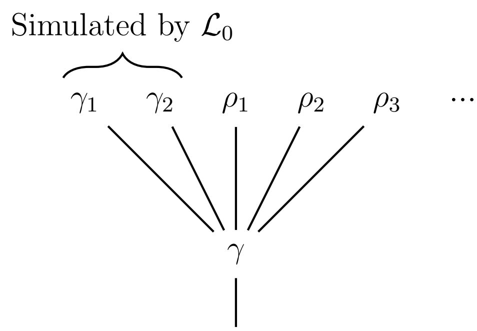
Each requirement $\mathcal{R}$ will have an *instance* $\mathcal{R}^\eta$ operating under each possible guess η at the outcomes of the lower priority requirements. Each of these instances of a particular requirement will be operating independently, but the actions of all of the instances of all the requirements will be uniformly computable. So for example each instance $\mathcal{M}_e^\eta$ of a minimality requirement will be trying to build an e-splitting tree, using η to guess at whether or not $\mathcal{M}_{e-1}$ successfully built an e-splitting tree, how $\mathcal{L}_{e-1}$ was satisfied, and the node chosen by $\mathcal{P}_{e-1}$ as an initial segment of A. The instances of different requirements will not be completely independent; for example, a requirement $\mathcal{M}_e^\eta$ must take into account all of the lower priority requirements $\mathcal{L}_d^\nu$ for $d \geq e$, $\nu \succ \eta$.

One can think of the argument as a forcing argument except that there are some (effective) interactions between the conditions. In a standard forcing construction to build a minimal degree, for each requirement $\mathcal{M}_e$, after forcing the outcomes of the lower priority requirements, we decide non-uniformly whether we can find an e-splitting tree T_e, or whether there is a node σ which has no e-splitting tree extending it. The tree T_e is in some sense built after deciding on the outcomes of the previous requirements. What we will do is attempt to build, for each guess η of $\mathcal{M}_e$ at the outcomes of the lower priority requirements, an e-splitting tree T_e^η; and then we will, at the end of the construction, choose one instance $\mathcal{M}_e^\eta$ of $\mathcal{M}_e$ to use depending on the outcomes of the lower priority requirements, and then we use the tree T_e^η built by that instance. All of these trees T_e^η were already built before we started determining the outcomes of the requirements.

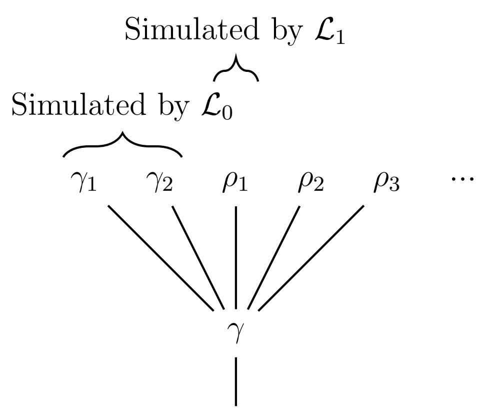
There is only one instance $\mathcal{M}_0^\varnothing$ of $\mathcal{M}_0$, since there are no higher priority requirements. After the construction, we will ask $\mathcal{M}_0^\varnothing$ what its outcome was. We then have an instance of $\mathcal{L}_0$ which guessed this outcome for $\mathcal{M}_0$, and we ask this instance what its outcome was. This gives us an instance of $\mathcal{P}_0$ that guessed correctly, and so on. So at the end, we use only one instance of each requirement, and follow whatever that instance did.

We now need to consider in more detail the interactions between the requirements. We saw in the previous section that an $\mathcal{M}$ requirement must take into account lower priority $\mathcal{L}$ requirements. In the full construction, we will have not only many different lower priority $\mathcal{L}$ requirements, but also many different instances of each one that the $\mathcal{M}$ requirement must take into account.

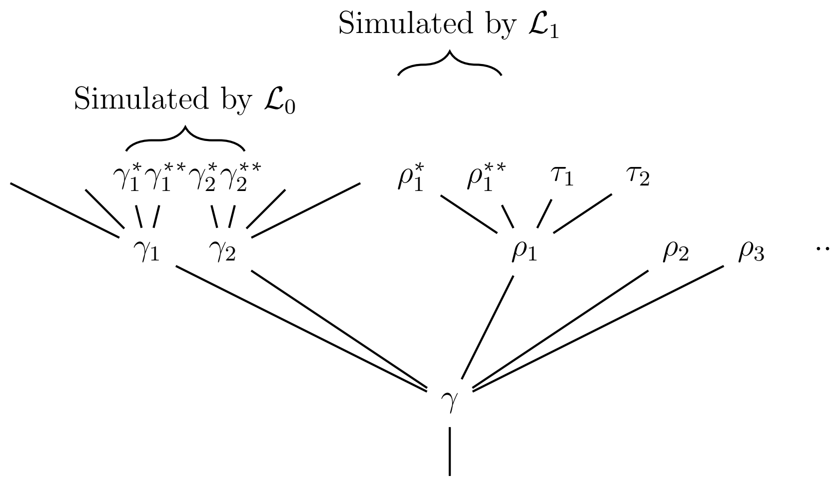
Consider three requirements, $\mathcal{M}$ of highest priority, $\mathcal{L}_0$ of middle priority, and $\mathcal{L}_1$ of lowest priority. Suppose that both $\mathcal{L}_0$ and $\mathcal{L}_1$ correctly guess that $\mathcal{M}$ has the infinitary outcome, building a splitting tree. As described before, when we extend γ, we get a picture as follows (ignoring $\mathcal{L}_1$ for now):



Now $\mathcal{L}_1$ guesses at the outcome of $\mathcal{L}_0$, and if for example $\mathcal{L}_0$ has the finitary outcome ρ_1, and $\mathcal{L}_1$ guesses this, then $\mathcal{L}_1$ must simulate computations extending ρ_1.



Now in the next step we found extensions ρ_1^*, ρ_1^{**} of ρ_1 that split with the other extensions. Before, we could simply extend ρ_1 to ρ_1^* and ρ_1^{**}. Now, while we are looking for the extensions, $\mathcal{L}_1$ might simulate other computations, say $\tau_1, \tau_2, \ldots$, extending ρ_1. We cannot remove these from the tree. So as before, $\mathcal{L}_1$ stops simulating them:



Now we have arrived to the point where computations above τ_1 and τ_2 are no longer being simulated by any $\mathcal{L}$ requirements, and when we extend γ_1^*, γ_1^{**}, γ_2^*, and γ_2^{**} we can find extensions of τ_1 and τ_2 which e-split with these extensions (as well as with extensions of ρ_2^*, ρ_3^*, etc.).

It was important here that we were only dealing with finitely many lowness requirements at a time, because eventually we arrived at a point where parts of the tree were no longer being simulated by any lowness requirement. In the full construction, there will be some important bookkeeping to manage which lowness requirements are being considered at any particular time, so that we sufficiently delay the introduction of new lowness requirements. (This will be accomplished by giving each element of the tree a *scope* in the next section.)

This is not the only case where one $\mathcal{L}$ requirement needs to simulate computations through nodes not simulated by another computation. We will introduce a relation $\mathcal{L}_{d_1}^{\nu_1} \rightsquigarrow \mathcal{L}_{d_2}^{\nu_2}$ which means that $\mathcal{L}_{d_1}^{\nu_1}$ must simulate computations above nodes that are kept on the tree to be the finitary outcome of $\mathcal{L}_{d_2}^{\nu_2}$, but which are not simulated by $\mathcal{L}_{d_2}^{\nu_2}$. We suggest reading $\mathcal{L}_{d_1}^{\nu_1} \rightsquigarrow \mathcal{L}_{d_2}^{\nu_2}$ as "$\mathcal{L}_{d_1}^{\nu_1}$ *watches* $\mathcal{L}_{d_2}^{\nu_2}$". Given the previous example, if $d_1 < d_2$ and $\nu_1 \prec \nu_2$ then we will have $\mathcal{L}_{d_1}^{\nu_1} \rightsquigarrow \mathcal{L}_{d_2}^{\nu_2}$, but there will be other, more complicated, cases where $\mathcal{L}_{d_1}^{\nu_1} \rightsquigarrow \mathcal{L}_{d_2}^{\nu_2}$.

Now consider the case of two $\mathcal{M}$ requirements $\mathcal{M}_0$ and $\mathcal{M}_1$ which are of higher priority than two $\mathcal{L}$ requirements $\mathcal{L}_0$ and $\mathcal{L}_1$. Suppose that $\mathcal{M}_0$ successfully builds a 0-splitting tree T_0, and suppose that $\mathcal{L}_0$ and $\mathcal{L}_1$ both correctly guesses this. Then $\mathcal{M}_1$ is trying to build a 1-splitting subtree of T_0. Suppose that:

(1) $\mathcal{L}_0$ guesses that $\mathcal{M}_1$ builds a 1-splitting subtree of T_0. If $\mathcal{M}_1$ succeeds at building a 1-splitting subtree T_1 of T_0, $\mathcal{L}_0$ must be able to simulate computations on T_1. On the other hand, when $\mathcal{M}_0$ built T_0, there were some nodes of T_0 which $\mathcal{L}_0$ did not simulate. So to make sure that $\mathcal{L}_0$ simulates computations through T_1, $\mathcal{M}_1$ should look for 1-splits through nodes of T_0 that are simulated by $\mathcal{L}_0$.

(2) $\mathcal{L}_1$ guesses that $\mathcal{M}_1$ fails to build a 1-splitting tree. If this guess is correct, then there is some node σ above which $\mathcal{M}_1$ fails to find a 1-split. $\mathcal{M}_1$ defines a subtree T_1' of T_0 containing no 1-splits. The tree T_1' is not being defined dynamically—it is just the subtree of T_0 above σ where $\mathcal{M}_1$ looked for (and failed to find) a 1-split.

Recall that $\mathcal{M}_0$ acted specifically to keep simulations computed by $\mathcal{L}_1$ on the tree T_0. The tree T_1' also needs to keep these computations. Putting everything together, this means that we should be looking for 1-splits through the nodes which were kept on T_0 for the sake of $\mathcal{L}_1$; these are nodes which are not simulated by $\mathcal{L}_1$, but which might be used for the finitary outcomes of $\mathcal{L}_1$.

So we see from (1) that $\mathcal{M}_1$ should look for 1-splits through nodes that are simulated by $\mathcal{L}_0$, and from (2) that $\mathcal{M}_1$ should look for 1-splits through the nodes which are kept on T_0 for the sake of $\mathcal{L}_1$, but which are not simulated by $\mathcal{L}_1$. This suggests that $\mathcal{L}_0$ should simulate the nodes which are kept on T_0 for the sake of $\mathcal{L}_1$, so that $\mathcal{L}_0 \rightsquigarrow \mathcal{L}_1$.

Given a string of outcomes ν, define $\Delta(\nu) \in \{f, \infty\}^{<\omega}$ to be the string

$$\langle \nu(\mathcal{M}_0), \nu(\mathcal{M}_1), \nu(\mathcal{M}_2), \dots \rangle$$

except that we replace any entry which is not ∞ with f. We can put an ordering $\lesssim$ on these using the lexicographic order with $\infty < f$. Suppose that ν_1 and ν_2 are guesses by $\mathcal{L}_{d_1}$ and $\mathcal{L}_{d_2}$ respectively at the outcomes of the higher priority requirements. Define $\mathcal{L}_{d_1}^{\nu_1} \rightsquigarrow \mathcal{L}_{d_2}^{\nu_2}$ if and only if $\Delta(\nu_1) \prec \Delta(\nu_2)$ in the ordering just defined.

4 Construction

4.1 Procedure for constructing splitting trees

Given the tree T_{e-1} constructed by an instance of $\mathcal{M}_{e-1}$, we will describe the (attempted) construction by $\mathcal{M}_e$ of an e-splitting subtree T. This construction will be successful if there are enough e-splittings in T_{e-1}. We write $T[n]$ for the tree up to and including the nth level.

Let ξ be the guess by the particular instance of $\mathcal{M}_e$ at the outcomes of lower priority requirements. This guess ξ determines an instance of $\mathcal{M}_{e-1}$ compatible with the given instance of $\mathcal{M}_e$, and ξ also includes a guess at the outcome of $\mathcal{M}_{e-1}$. The tree T_{e-1} inside of which we build T depends on the outcome of $\mathcal{M}_{e-1}$, i.e., if ξ guesses that $\mathcal{M}_{e-1}$ builds an $(e-1)$-splitting tree then T_{e-1} is this tree, and if ξ guesses that $\mathcal{M}_{e-1}$ fails to do so, then T_{e-1} is the tree with no $(e-1)$-splits witnessing this failure. (This will all be made more precise in the next section; for now we simply describe the procedure of building T.) If we are successful in building T then $\mathcal{M}_e$ will have the infinitary outcome. We will also leave for later the description of the subtree of T_{e-1} that we use for the finitary outcome. In this section, we just define the procedure `Procedure(`e`,`ρ`,`T_{e-1}`)` for building an e-splitting tree T with root ρ in T_{e-1}. (The procedure for building T will not actually use the guess ξ, other than to determine what the tree T_{e-1} is, but it will be helpful to refer to ξ in the discussion.)

When building T, $\mathcal{M}_e$ must take into account instances of lowness requirements $\mathcal{L}_d^\eta$ with $d \geq e$ and η extending $\xi\hat{\ }\infty$. (Since T is being built under the assumption that $\mathcal{M}_e$ has the infinitary outcome, it only needs to respect lowness requirements that guess that this is the case.) When considering $\mathcal{L}_d^\eta$ while building T, we will only need to know the guesses by $\mathcal{L}_d^\eta$ at the outcomes of the $\mathcal{M}$ requirements $\mathcal{M}_{e+1}, \dots, \mathcal{M}_d$ of lower priority than $\mathcal{M}_e$ but higher priority than $\mathcal{L}_d$ (because the only lowness requirements we need to consider have the same guesses at the requirements $\mathcal{M}_1, \dots, \mathcal{M}_e$), and moreover we will only care about whether the guess is the infinitary outcome or a finitary outcome.

The tree T will be a *labeled tree*, as follows. Figure 1 below may help the reader to understand the structure of the tree. Each node of the tree T will be labeled to show which lowness requirements are simulating it, and which lowness requirements are using it for the finitary outcome (of diagonalizing against a computable set R). Each node σ is given a *scope* $\operatorname{scope}(\sigma)$ and a *label* $\ell(\sigma)$. The scope is a natural number ≥ 0, and the label is an element of

$$\texttt{Labels} = \{f, \infty\}^{<\omega} \cup \{\top\}.$$

The scope represents the number of lowness requirements that are being considered at this level of the tree, i.e., if the scope of a node is n, then we are considering lowness requirements $\mathcal{L}_e, \dots, \mathcal{L}_{e+n}$. If a node σ has scope n, then the label of σ will be an element of

$$\texttt{Labels}_n = \{f, \infty\}^{\leq n} \cup \{\top\}.$$

Note that the label might have length less than n, and might even be the empty string; an element of $\texttt{Labels}_n$ of length m corresponds to a guess by $\mathcal{L}_{e+m}$. We order the labels lexicographically with $\infty < f$, and with $\top$ as the greatest element. E.g., in $\texttt{Labels}_2$, we have

$$\top > ff > f\infty > f > \infty f > \infty\infty > \infty > \varnothing.$$

We use $\preceq$ for this ordering. We often think of this ordering as being an ordering $\preceq_n$ on $\texttt{Labels}_n$, and write $\operatorname{pred}_n(\eta)$ for the predecessor of η in $\texttt{Labels}_n$. Though $\texttt{Labels}$ is well-founded, it does not have order type ω, and so we need to restrict to $\texttt{Labels}_n$ to make sense of the predecessor operator.

We think of elements of $\{f, \infty\}^n$ as guesses by $\mathcal{L}_{e+n}$ at the outcomes of $\mathcal{M}_{e+1}, \dots, \mathcal{M}_{e+n}$, and $\top$ is just an element greater than all of the guesses. Given an instance $\mathcal{L}^{\eta}_{e+n}$ of a lowness requirement of lower priority than $\mathcal{M}_e$, we write $\Delta_{>e}(\eta)$ for

$$\langle \Delta(\eta)(e+1), \Delta(\eta)(e+2), \dots, \Delta(\eta)(e+n) \rangle \in \{f, \infty\}^n.$$

Recall that $\Delta(\eta)$ just replaces the entries of η by f or ∞; $\Delta_{>e}(\eta)$ is the string which consists of the guesses of η at the outcomes (finitary f or infinitary ∞) of $\mathcal{M}_{e+1}, \dots, \mathcal{M}_d$. Given two requirements $\mathcal{L}^{\nu_1}_{d_1}$ and $\mathcal{L}^{\nu_2}_{d_2}$ with ν_1, ν_2 extending $\xi^\frown\infty$, $\mathcal{L}^{\nu_1}_{d_1} \rightsquigarrow \mathcal{L}^{\nu_2}_{d_2}$ if and only if $\Delta_{>e}(\nu_1) \prec \Delta_{>e}(\nu_2)$ lexicographically.

The label $\ell(\sigma^*)$ means that σ^* was kept on the tree in order to preserve simulated computations by instances of lowness requirements $\mathcal{L}^{\eta}_{e+|\ell(\sigma^*)|}$ with $\Delta_{>e}(\eta) = \ell(\sigma^*)$. So if $\mathcal{L}^{\eta}_d$ is an instance of a lowness requirement, and σ^* is the child of σ on T, with $d \leq e + \operatorname{scope}(\sigma^*)$, then:

- if $\Delta_{>e}(\eta) \prec \ell(\sigma^*)$, then if $\mathcal{L}^{\eta}_d$ simulates computations through σ, it also simulates computations through σ^*; and
- if $\Delta_{>e}(\eta) = \ell(\sigma^*)$, then $\mathcal{L}^{\eta}_d$ does not simulate computations through σ^*, but σ^* might be used for the finitary outcome of $\mathcal{L}^{\eta}_d$.

A node σ^* with $\ell(\sigma^*) = \top$ is simulated by every lowness requirement that simulates its parent. Note that we always say that σ^* is simulated if its parent σ is simulated, rather than just saying that σ^* is simulated. This is because when we apply $\Delta_{>e}$ to the guesses η of different instances of $\mathcal{L}_d^\eta$, we lump together many instances with different values for the finitary outcomes. Some of these instances may not simulate computations through σ^* because one of the higher priority requirements forces A to extend a node incompatible with σ, while other instances may be forced to simulate computations through σ^* because of these higher priority requirements. For a particular instance of a lowness requirement, there is some initial segment of A determined by the higher priority requirements; think of the labels ℓ as applying above this initial segment.

Certain levels of the tree T_e will be called *expansionary levels*. From the nth expansionary level of the tree on, we will begin to consider requirements $\mathcal{L}_{e+1}, \ldots, \mathcal{L}_{e+n}$, using guesses from at the outcomes of $\mathcal{M}_{e+1}, \ldots, \mathcal{M}_{e+n}$. (Recall that the scope of a node represents the lowness requirements that it considers.) The nodes at the nth expansionary level or higher, but below the $n+1$st expansionary level, will be said to be in the *nth strip*. We write $e_1, e_2, e_3, \ldots$ for the expansionary levels. The expansionary levels are defined statically by $e_1 = 0$ and

$$e_{i+1} = e_i + 2^{i+5}.$$

One might expect that if σ is in the nth strip, then $\operatorname{scope}(\sigma)$ will be n. This will not quite be the case; an expansionary level is where we start considering more requirements, but this might not happen immediately for particular nodes. Instead, if σ is in the nth strip, we will have $\operatorname{scope}(\sigma) = n - 1$ or $\operatorname{scope}(\sigma) = n$. The scope of a child will always be at least the scope of its parent. We say that σ^*, a child of σ, is an *expansionary node* if $\operatorname{scope}(\sigma^*) > \operatorname{scope}(\sigma)$. We say that an expansionary node σ^* is an *nth expansionary node* if $\operatorname{scope}(\sigma^*) = n$. Along any path in the tree, there is one expansionary node for each n; it is clear from the fact that the lengths of labels are non-decreasing along paths that there is at most one, and we will show in Lemma 4.3 that there is at least one. Moreover, the nth expansionary node along a path will occur in the nth strip.

Suppose that σ^* is a child of σ, with $\operatorname{scope}(\sigma) = n$. If σ^* is an $(n+1)$st expansionary node, then we will have $\operatorname{scope}(\sigma^*) = n + 1$ and $\ell(\sigma^*) = \top$. Otherwise, $\operatorname{scope}(\sigma^*) = \operatorname{scope}(\sigma) = n$ and we will have either $\ell(\sigma^*) = \ell(\sigma)$ or $\ell(\sigma^*) \prec \ell(\sigma)$. When the label stays the same (or when σ^* is an expansionary node), we say that σ^* is a *main child* of σ. Each node σ will have exactly two main children, which will e-split with each other. Otherwise, if $\ell(\sigma^*) \prec \ell(\sigma)$, then we say that σ^* is a *secondary child* of σ.

Recall from the end of the previous section that when we look for splits, we do so through only a subtree of T_{e-1}. T_{e-1} itself will be a labeled tree, with scopes $\operatorname{scope}_{e-1}: T_{e-1} \to \omega$ and labels $\ell_{e-1}: T_{e-1} \to \texttt{Labels}$. Given τ on T_{e-1}, let $T_{e-1}\{\leadsto_\tau^{\succ f}\}$ be the subtree of T_{e-1} above τ (so that τ is the root node of $T_{e-1}\{\leadsto_\tau^{\succ f}\}$) such that given σ on $T_{e-1}\{\leadsto_\tau^{\succ f}\}$, the children of σ in $T_{e-1}\{\leadsto_\tau^{\succ f}\}$ are the children σ^* of σ on T_{e-1} with $\ell_{e-1}(\sigma^*) \succ f$. When we look for a splitting extension of τ, we look through $T_{e-1}\{\leadsto_\tau^{\succ f}\}$.

In general, for any tree S, node $\tau \in S$, and relation $R(\sigma^*)$ (or even a relation $R(\sigma^*, \sigma)$ between a node σ^* and its parent σ), we can define the subtree $S\{\leadsto_\tau^R\}$ as the tree with root node τ, and such that whenever $\sigma \in S\{\leadsto_\tau^R\}$, the children σ^* of σ on $S\{\leadsto_\tau^R\}$ are exactly the children σ^* of σ on S such that $R(\sigma^*)$ holds (or $R(\sigma^*, \sigma)$). We will use this notation for the

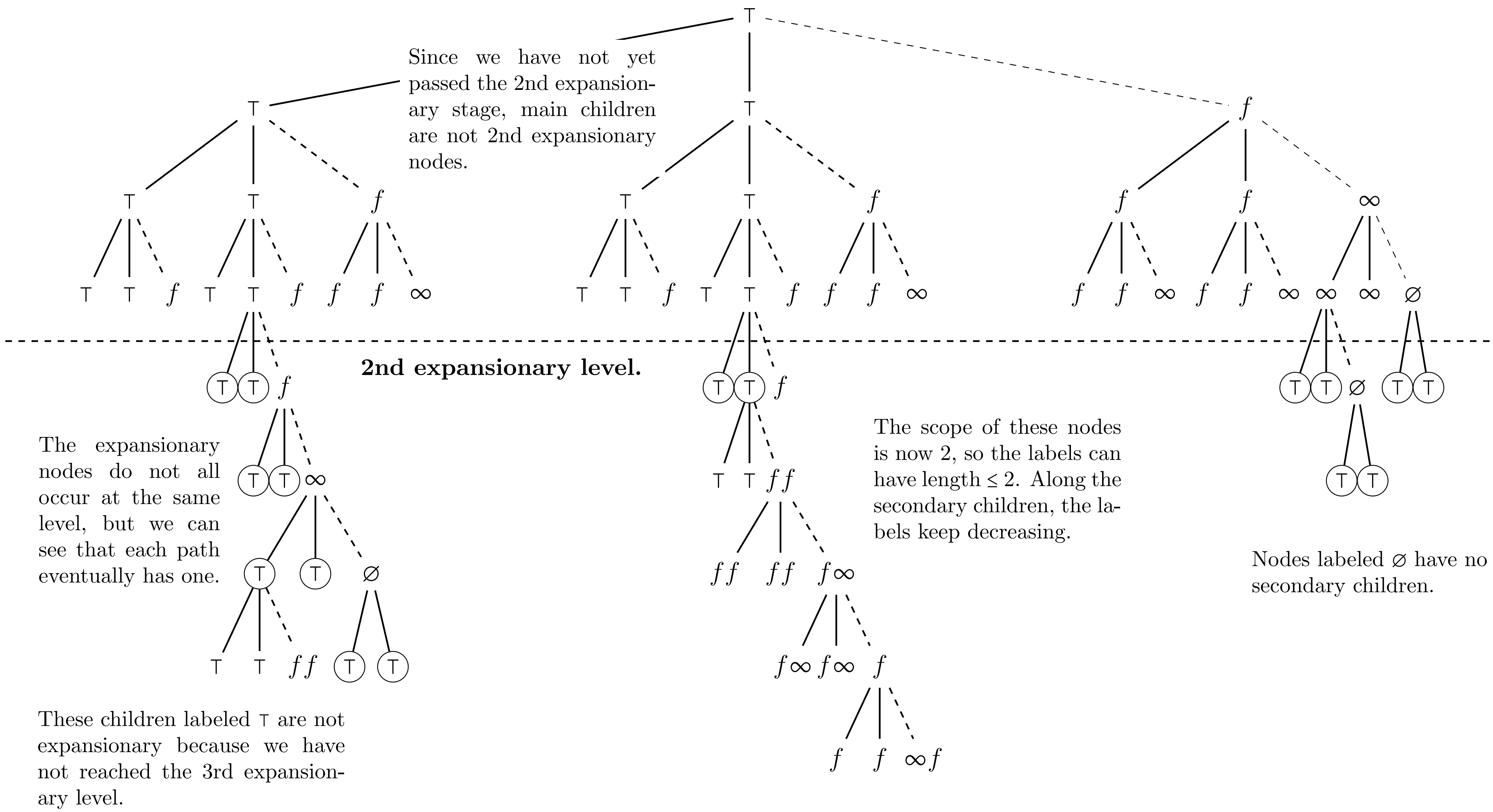


Figure 1: An example of what the labeled tree might look like. We draw the main children with a solid line and the secondary children with a dashed line. Expansionary nodes are shown by a circle. To fit the tree onto a single page, we have made some simplifications: (a) we have omitted some nodes from the diagram; (b) we have assumed that each node has only one secondary child; and (c) we have assumed that the 2nd expansionary level e_2 is much smaller than the value of 128 that we set in the construction. We show the 2nd expansionary level with the long horizontal dashed line.

trees $S\{\rightsquigarrow_\tau^{>\eta}\}$ and $S\{\rightsquigarrow_\tau^{\geq\eta}\}$ for $\eta \in \texttt{Labels}$, and $S\{\rightsquigarrow_\tau^{main}\}$ where $main(\sigma^*, \sigma)$ is the relation of being a main child. So for example $T_{e-1}\{\rightsquigarrow_\sigma^{main}\}[s]$ is the tree consisting of main children of main children of... main children of σ.

The input tree T_{e-1} will have similar properties to those described above. We say that a finitely branching tree T_{e-1} with labels ℓ_{e-1} and scopes scope_{e-1} is *admissible* if:

(1) Each $\sigma \in T_{e-1}$ has two main children σ^* and σ^{**} with $\ell_{e-1}(\sigma^*) \geq \ell_{e-1}(\sigma)$ and $\ell_{e-1}(\sigma^{**}) \geq \ell_{e-1}(\sigma)$.

(2) If σ^* is a child of σ, then $\text{scope}_{e-1}(\sigma^*) \geq \text{scope}_{e-1}(\sigma)$.

(3) For each n, each path through T_{e-1} contains a node σ with $\ell_{e-1}(\sigma) = \top$ and $\text{scope}_{e-1}(\sigma) \geq n$.

We are now ready to describe the procedure for constructing splitting trees.

$\texttt{Procedure}(e,\rho,T_{e-1})$:
Input: A value $e \geq 0$, an admissible labeled tree T_{e-1} with labels $\ell_{e-1}(\cdot)$ and scopes $\text{scope}_{e-1}(\cdot)$, and a node ρ on T_{e-1} with $\ell_{e-1}(\rho) = \top$.
Output: A possibly partial labeled e-splitting tree T, built stage-by-stage.

Construction. To begin, the root node of T is ρ with $\text{scope}(\rho) = 1$ and $\ell(\rho) = \top$. This is the 0th level of the tree, $T[0]$. At each stage of the construction, if we have so far built T up to the nth level $T[n]$, we try to add an additional $n+1$st level.

At stage s, suppose that we have defined the tree up to and including level n, and the last expansionary level $\leq n$ was e_t. Look for a length l such that for each leaf σ of $T[n]$, there is an extension σ' of σ with $\sigma' \in T_{e-1}\{\rightsquigarrow_\sigma^{main}\}$ with $\ell_{e-1}(\sigma') = \top$, and there are extensions σ^* and σ^{**} of σ on $T_{e-1}\{\rightsquigarrow_{\sigma'}^{>f}\}$, such that σ^* and σ^{**} are of length l, and such that all of these extensions pairwise e-split, i.e., for each pair of leaves σ, τ of $T[n]$, these extensions σ^*, σ^{**}, τ^*, and τ^{**} all e-split with each other. (At stage s, we look among the first s-many extensions of these leaves, and we run computations looking for e-splits up to stage s. If we do not find such extensions, move on to stage $s+1$.)

If we do find such extensions, we will define $T[n+1]$ as follows. To begin, we must wait for $T_{e-1}[s]$ to be defined. In the meantime, we designate each σ as *waiting with main children σ^* and σ^{**}*. (This designation is purely for the use of the simulations for lowness requirements, and has no effect on the resulting tree T.) While waiting, we still count through stages of the construction, so that after we resume the next stage of the construction will not be stage $s+1$ but some other stage $t > s$ depending on how long we wait. Once $T_{e-1}[s]$ has been defined, for each leaf σ of $T[n]$, the children of σ in $T[n+1]$ will be:

- σ^*, with:
  - if no predecessor of σ on T is t-expansionary, set $\text{scope}(\sigma^*) = \text{scope}(\sigma) + 1$ and $\ell(\sigma^*) = \top$, or
  - $\text{scope}(\sigma^*) = \text{scope}(\sigma)$ and $\ell(\sigma^*) = \ell(\sigma)$ otherwise;
- σ^{**}, with:

- $\mathrm{scope}(\sigma^{**}) = \mathrm{scope}(\sigma) + 1$ and $\ell(\sigma^{**}) = \top$ if no predecessor of σ on T is t-expansionary, or
- $\mathrm{scope}(\sigma^{**}) = \mathrm{scope}(\sigma)$ and $\ell(\sigma^{**}) = \ell(\sigma)$ otherwise;

- If $\ell(\sigma) \succ \varnothing$, each other extension $\sigma^\dagger$ of σ on $T_{e-1}\{\wr_{\sigma}^{\succ\varnothing}\}[s]$ which is incompatible with σ^* and σ^{**} will be a child of σ on T. Put $\mathrm{scope}(\sigma^\dagger) = \mathrm{scope}(\sigma)$. Define $\ell(\sigma^\dagger)$ as follows. Let $n = \mathrm{scope}(\sigma)$. Let $\eta \in \mathtt{Labels}_n$ be greatest such that $\sigma^\dagger \in T_{e-1}\{\wr_{\sigma}^{\succeq\eta}\}$. Then:
  - If η is $\top$ or begins with f, then let $\ell(\sigma^\dagger) = \mathrm{pred}_n(\ell(\sigma))$.
  - If η begins with ∞, say $\eta = \infty\eta^*$, then $\ell(\sigma^\dagger)$ will be the minimum, in $\mathtt{Labels}_n$, of $\mathrm{pred}_n(\ell(\sigma))$ and η^*.

  Note that $\mathrm{pred}_n(\ell(\sigma))$ exists because $\ell(\sigma) \succ \varnothing$.

The children σ^* and σ^{**} are the main children of σ, and the $\sigma^\dagger$, if they exist, are secondary children. This ends the construction at stage s.

End construction.

We say that the procedure is *successful* if it never gets stuck, and construct the nth level of the tree T for every n. The next lemma is the formal statement that if T_{e-1} has enough e-splits, then the procedure is successful.

Lemma 4.1. *Fix e, an admissible labeled tree T_{e-1}, and $\rho \in T_{e-1}$ with $\ell_{e-1}(\rho) = \top$. Suppose that for all $\sigma \in T_{e-1}\{\wr_{\rho}^{\succ\varnothing}\}$ with $\ell_{e-1}(\sigma) = \top$,*

- *for all n, there is $\tau \in T_{e-1}\{\wr_{\sigma}^{\succ f}\}$ such that $\Phi_e^{\tau}(n)\downarrow$, and*
- *there are n and $\tau_1, \tau_2 \in T_{e-1}\{\wr_{\sigma}^{\succ f}\}$ such that*

$$\Phi_e^{\tau_1}(n) \neq \Phi_e^{\tau_2}(n).$$

Then $\mathtt{Procedure}(e,\rho,T_{e-1})$ *is successful.*

As part of proving this lemma, we will use the following remark, which follows easily from the construction:

Remark 4.2. Every node on T is a node on $T_{e-1}\{\wr_{\rho}^{\succ\varnothing}\}$.

Proof of Lemma 4.1. If we have built T up to level n, and $T[n]$ has leaves $\sigma_1, \dots, \sigma_k$, then as T_{e-1} is admissible, for each i there are σ_i' on $T_{e-1}\{\wr_{\sigma_i}^{main}\}$ with $\ell(\sigma_i') = \top$. By the remark, each $\sigma_i' \in T_{e-1}\{\wr_{\rho}^{\succ\varnothing}\}$. Then using the assumption of the lemma and standard arguments there are $\sigma_i^*, \sigma_i^{**}$ on $T_{e-1}\{\wr_{\sigma_i'}^{\succ f}\}$ such that all of the σ_i^* and σ_i^{**} pairwise e-split. For sufficiently large stages s, we will find these extensions. $\square$

The remaining lemmas of this section give properties of the tree constructed by the procedure. The next few lemmas show that the tree T has expansionary levels and is e-splitting. As a result, we will see that T is admissible.

Lemma 4.3. *Suppose that* `Procedure(`e`,`ρ`,`T_{e-1}`)` *successfully constructs* T*. For each* $\sigma^* \in T[e_{n+1}]$*, there is a predecessor* σ *of* σ^* *which is* n*-expansionary.*

Proof. Let $\sigma_0 \in T[e_n]$ be the predecessor of σ^* at the nth expansionary level, and let $\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_k = \sigma^*$ be the sequence of predecessors of σ^* between σ_0 and σ^*. If any σ_{i+1} were a main child of σ_i, then either σ_{i+1} would be n-expansionary, or σ_i or one of its predecessors would be n-expansionary as desired. If none of these are expansionary, then we must have

$$\top \geq \ell(\sigma_0) \succ \ell(\sigma_1) \succ \ell(\sigma_2) \succ \cdots \succ \ell(\sigma_k) = \ell(\sigma^*)$$

with all of these in $\texttt{Labels}_{n-1}$. Since $e_{n+1} > e_n + 2^{n+1} > |\texttt{Labels}_{n-1}|$, this cannot be the case, and so some predecessor of σ^* must be expansionary. □

The following lemma is easy to see by inspecting the construction.

Lemma 4.4. *Suppose that* `Procedure(`e`,`ρ`,`T_{e-1}`)` *successfully constructs* T*. Given distinct leaves* σ *and* σ *of* $T[n]$*, and* $\sigma^*, \tau^* \in T[n+1]$ *which are children of* σ *and* τ *respectively, either:*

- σ^* *and* τ^* *are main children of* σ *and* τ *respectively, and* σ^* *and* τ^* e*-split,*
- σ^* *is a secondary child of* σ*, and* $\text{scope}(\sigma^*) = \text{scope}(\sigma)$ *and* $\ell(\sigma^*) \prec \ell(\sigma)$*, or*
- τ^* *is a secondary child of* τ*, and* $\text{scope}(\tau^*) = \text{scope}(\tau)$ *and* $\ell(\tau^*) \prec \ell(\tau)$*.*

Lemma 4.5. *Suppose that* `Procedure(`e`,`ρ`,`T_{e-1}`)` *successfully constructs* T*. Given distinct* σ *and* τ *in* T *at the* n*th expansionary level of the tree, and* σ^*, τ^* *which are extensions of* σ *and* τ *respectively at the* $n+1$*st expansionary level of the tree,* σ^* *and* τ^* *are* e*-splitting.*

Proof. Let $\sigma_0 = \sigma, \sigma_1, \sigma_2, \ldots, \sigma_k = \sigma^*$ be the sequence of predecessors of σ^* between σ and σ^*, and similarly for $\tau_0 = \tau, \tau_1, \tau_2, \ldots, \tau_k = \tau^*$. Since σ_0 and τ_0 are at the level e_n, and σ^* and τ^* are at the level e_{n+1}, we have $k \geq 2^{n+5}$. If, for any i, both σ_{i+1} and τ_{i+1} are main children of σ_i and τ_i, then by Lemma 4.4, σ_{i+1} and τ_{i+1} are e-splitting. We argue that this must happen for some $i < k$.

For each i, either (a) σ_{i+1} is n-expansionary and $\ell(\sigma_{i+1}) = \top$, or $\text{scope}(\sigma_{i+1}) = \text{scope}(\sigma_i)$ and either (b) $\ell(\sigma_{i+1}) \prec \ell(\sigma_i)$, or (c) $\ell(\sigma_{i+1}) = \ell(\sigma_i)$. There is at most one i for which (a) is the case. Thus there are at most $|\texttt{Labels}_n| + |\texttt{Labels}_{n+1}| \leq 2^{n+3}$ values of i for which (b) is the case. The same is true for the τ_i. So, as $k \geq 2^{n+5}$, there must be some i for which neither (a) nor (b) is the case for either the σ_i or the τ_i. For this i, we have both σ_{i+1} and τ_{i+1} are main children of σ_i and τ_i respectively, and so σ_{i+1} and τ_{i+1} are e-splitting. Thus σ^* and τ^* are e-splitting. □

Lemma 4.6. *Suppose that* `Procedure(`e`,`ρ`,`T_{e-1}`)` *successfully constructs* T*.* T *is an* e*-splitting tree: any two paths in* T *are* e*-splitting.*

Proof. Choose σ_1 and σ_2 initial segments of the two paths, long enough that they are distinct, which are at the nth expansionary level $T[e_n]$. Let τ_1, τ_2 be the longer initial segments of the paths at the $n+1$st expansionary level $T[e_{n+1}]$. Then by the previous lemma, τ_1 and τ_2 are e-splitting, and so the two paths are e-splitting. □

The next lemmas relate the labels of T to the labels on T_{e-1}. If a node is labeled on T_{e-1} so that it is not simulated by some lowness requirement, then it should also be labeled on T to not be simulated by that lowness requirement. (The converse is not necessary; T might determine that some node should not be simulated even if that was not determined by T_{e-1}.)

Lemma 4.7. *Suppose that* `Procedure(`e`,`ρ`,`T_{e-1}`)` *successfully constructs T. Given $\sigma \in T$ and $\sigma^* \in T\{\Lsh^{\succ\eta}_{\sigma}\}$, with $\ell_{e-1}(\sigma) = \top$, we have $\sigma^* \in T_{e-1}\{\Lsh^{\succ\infty\eta}_{\sigma}\}$. In particular, if $\ell_e(\sigma^*) \succ \eta$, then $\ell_{e-1}(\sigma^*) \succ \infty\eta$.*

Proof. It suffices to prove the lemma when σ^* is a child of σ on T, and $\ell(\sigma^*) \succ \eta$. We have two cases, depending on whether σ^* is a main child or secondary child of σ on T.

- If σ^* is a main child of σ on T, then $\sigma^* \in T_{e-1}\{\Lsh^{\succ f}_{\sigma'}\}$ and $\sigma' \in T_{e-1}\{\Lsh^{main}_{\sigma}\}$, and $\ell_{e-1}(\sigma') = \top$. Since $\ell_{e-1}(\sigma) = \top$, for every τ on T_{e-1} between σ and σ' we have $\ell_{e-1}(\tau) = \top$. Thus $\ell_{e-1}(\sigma') = \top$.

  Now for every τ on T_{e-1} between σ' and σ^*, we have $\ell_{e-1}(\tau) \succ f \succ \infty\eta$. So $\ell_{e-1}(\sigma^*) \succ \infty\eta$.

- If σ^* is a secondary child of σ, let $n = \text{scope}(\sigma)$ and let $\nu \in \texttt{Labels}_n$ be least such that $\sigma^* \in T_{e-1}\{\Lsh^{\succeq\nu}_{\sigma}\}$. If ν is $\top$ or begins with f, then $\nu \succ \infty\eta$ and so $\sigma^* \in T_{e-1}\{\Lsh^{\succ\infty\eta}_{\sigma}\}$. Otherwise, if ν begins with ∞, say $\nu = \infty\nu^*$, then $\ell(\sigma^*) \succ \eta$ is the minimum, in $\texttt{Labels}_n$, of $\text{pred}_n(\ell(\sigma))$ and ν^*. Thus $\nu^* \succ \eta$, which means that $\infty\nu^* \succ \infty\eta$, and so $\sigma^* \in T_{e-1}\{\Lsh^{\succ\infty\eta}_{\sigma}\}$.

This proves the lemma. □

Similarly, we can prove the same lemma but replacing $\succ$ with $\succeq$. We have:

Lemma 4.8. *Suppose that* `Procedure(`e`,`ρ`,`T_{e-1}`)` *successfully constructs T. Given $\sigma \in T$ and $\sigma^* \in T\{\Lsh^{\succeq\eta}_{\sigma}\}$, we have $\sigma^* \in T_{e-1}\{\Lsh^{\succeq\infty\eta}_{\sigma}\}$. In particular, if $\ell_e(\sigma^*) \succeq \eta$, then $\ell_{e-1}(\sigma^*) \succeq \infty\eta$.*

Finally, putting together results from all of these lemmas, we have:

Lemma 4.9. *Suppose that* `Procedure(`e`,`ρ`,`T_{e-1}`)` *successfully constructs T. Then T is an admissible tree.*

4.2 Minimality requirements

In the previous subsection, we described a procedure for constructing an e-splitting tree. In this section, we will show how the procedure is applied.

We begin with $\mathbb{T}_{-1}$ defined as in Section 2. We put $\ell_{-1}(\sigma) = \top$ and $\text{scope}_{-1}(\sigma) = |\sigma|$ for each $\sigma \in \mathbb{T}_{-1}$. This $\mathbb{T}_{-1}$ is admissible. For each instance $\mathcal{M}_e^{\xi}$ we will define trees $\mathbb{T}_e^{\xi\hat{}\infty}$ and $\mathbb{T}_e^{\xi\hat{}\sigma}$, where $\mathbb{T}_e^{\xi\hat{}\infty}$ is the outcome of the attempt to construct an e-splitting tree, and the $\mathbb{T}_e^{\xi\hat{}\sigma}$ are subtrees which would witness the failure of the construction of the e-splitting tree. Define

- $\mathbb{T}_e^{\xi\hat{}\infty}$ is the labeled tree T produced by `Procedure(`e`,`σ`,`$\mathbb{T}_{e-1}^{\xi\upharpoonright_{3e-3}}$`)` where $\sigma = \xi(\mathcal{P}_{e-1})$ (or σ is the root of $\mathbb{T}_{-1}$ if $e = 0$). The tree T built by this procedure has labels ℓ_e and scopes scope_e defined in its construction.

- $\mathbb{T}_e^{\xi\hat{}\sigma}$ is the tree $\mathbb{T}_{e-1}^{\xi\upharpoonright 3e-3}\{\wr^{\succ f}_{\sigma}\}$. The labels ℓ_e of the tree $\mathbb{T}_e^{\xi\hat{}\sigma}$ are defined by setting:
  - $\text{scope}_e(\sigma) = 1$ and $\ell_e(\sigma) = \top$;
  - for $\tau \neq \sigma$, $\text{scope}_e(\tau) = \text{scope}_{e-1}(\tau) - 1$ and $\ell_e(\tau) = \top$ if $\ell_{e-1}(\tau) = \top$, or $\ell_e(\tau) = \eta$ if $\ell_{e-1}(\tau) = f\eta$.

There is a uniform and computable construction of all of these trees simultaneously. Whenever in `Procedure(`e`,`σ`,`$\mathbb{T}_{e-1}^{\xi\upharpoonright 3e-3}$`)` we need to determine the next level of the tree $\mathbb{T}_{e-1}^{\xi\upharpoonright 3e-3}$, the procedure waits until this next level is defined. If $\mathbb{T}_{e-1}^{\xi\upharpoonright 3e-3}$ is a total tree then at some point enough of it will be define for the procedure to continue, and it if is not a total tree, then the procedure will get stuck and $\mathbb{T}_e^{\xi\hat{}\infty}$ will also be partial. Similarly, to define a level of $\mathbb{T}_e^{\xi\hat{}\sigma} = \mathbb{T}_{e-1}^{\xi\upharpoonright 3e-3}\{\wr^{\succ f}_{\sigma}\}$, we need to first build some portion of $\mathbb{T}_{e-1}^{\xi\upharpoonright 3e-3}$.

To be a bit more precise, there are two parts of `Procedure`. First, there is the part where we look for extensions σ^* and σ^{**} of each leaf σ of $T[n]$; and second, after waiting for $T_{e-1}[s]$ to be defined, we define the next level $T[n+1]$ of T. If in the first part of `Procedure`, the tree T_{e-1} has not been sufficiently defined, we just end the current stage of the procedure and restart at the next stage. In the second part of `Procedure`, we just wait for $T_{e-1}[s]$ to be defined, and then continue from where we were; but while we wait, we still count through the stages, so that when we return it is not at stage $s+1$ but at some greater stage depending on how long we waited for T_{e-1}. This will be important to make sure that the simulations used by the lowness requirements are not too slow.

After we define the true path, we will use Lemma 4.1 to show that along the true path the trees are all fully defined and admissible. When we are not on the true path, e.g. if the outcome of $\mathcal{M}_e^\eta$ is finitary (which means that `Procedure` is unsuccessful because it cannot find enough e-splits), the tree $\mathbb{T}_e^{\eta\hat{}\infty}$ will be partial, and then any tree defined using this tree will also be partial. Of course, these will all be off the true path.

4.3 Construction of A and the true path

We simultaneously define A and a true path of outcomes π by finite extension. The construction of the trees in the previous section was uniformly computable, but A and the true path π will non-computable. (This of this part of the construction as analogous to choosing a generic in a forcing construction.)

Begin with $A_{-1} = \varnothing$ and $\pi_{-1} = \varnothing$. Suppose that we have so far defined $A_s \prec A$ and $\pi_s \prec \pi$, with $|\pi_s| = s+1$. To define A_{s+1} and π_{s+1} we first ask the next requirement what it's outcome is, and then define the extensions appropriately. We use $\mathbb{T}_e$ for the tree defined along the true outcome, i.e. $\mathbb{T}_e := \mathbb{T}_e^{\pi_{3e}}$. Begin with $\pi_{-1} = \varnothing$.

$s+1 = 3e$**:** Consider $\mathcal{M}_e^{\pi_s}$. By Lemma 4.1, either $\mathbb{T}_e^{\pi_s\hat{}\infty}$ is an e-splitting tree, or there is $\sigma \in \mathbb{T}_{e-1}\{\wr^{\succ\varnothing}_{\pi_s(\mathcal{P}_{e-1})}\}$ with $\ell_{e-1}(\sigma) = \top$ such that either:

(1) there is n such that for all $\tau \in \mathbb{T}\{\wr^{\succ f}_{\sigma}\}_{e-1}$, $\Phi_e^\tau(n)\uparrow$, or

(2) for all $\tau_1, \tau_2 \in \mathbb{T}\{\wr^{\succ f}_{\sigma}\}_{e-1}$ and n,

$$\Phi_e^{\tau_1}(n)\downarrow \wedge \Phi_e^{\tau_2}(n)\downarrow \quad\longrightarrow\quad \Phi_e^{\pi_1}(n) = \Phi_e^{\pi_2}(n).$$

In the former case, let $\pi_{s+1} = \pi_s{}^\frown\infty$ and $A_{s+1} = A_s$.

In the latter two case, let $\pi_{s+1} = \pi_s{}^\frown\sigma$ and $A_{s+1} = \sigma \succeq A_s$.

$s+1 = 3e+1$**:** Consider $\mathcal{L}_e^{\pi_s}$ with $e = \langle e_1, e_2 \rangle$. If there is $\sigma \in \mathbb{T}_e$ and n such that $\Psi_{e_1}^{\sigma}(n) \neq R_{e_2}(n)$, then let $A_{s+1} = \sigma$ and $\pi_{s+1} = \pi_s{}^\frown\sigma$. We may choose σ to have $\ell_e(\sigma) = \top$, as (by Lemma 4.10) $\mathbb{T}_e$ is admissible. Otherwise, if there is no such σ, let $A_{s+1} = A_s$ and $\pi_{s+1} = \pi_s{}^\frown\infty$.

$s+1 = 3e+2$**:** Consider $\mathcal{P}_e^{\pi_s}$. If $A_s = \sigma \in \mathbb{T}_e$, let τ_1 and τ_2 be the two main children of σ on $\mathbb{T}_e$. Choose $A_{s+1} \succ \sigma$ to be whichever of τ_1, τ_2 is not an initial segment of the eth c.e. set W_e.

We define $A = \bigcup_s A_s$ and the true sequence of outcomes $\pi = \bigcup_s \pi_s$. We denote by $\pi_{\mathcal{R}}$ the true outcome up to and including the requirement $\mathcal{R}$; for example, $\pi_{\mathcal{M}_e} = \pi_{3e}$; and similarly for $A_{\mathcal{R}}$.

In the following lemma, we prove that along the true path, the trees that we construct are total and admissible.

Lemma 4.10. *For each e, $\mathbb{T}_e$ is an admissible labeled tree.*

Proof. We argue by induction on e. $\mathbb{T}_{-1}$ is an admissible tree. Given $\mathbb{T}_e$ total and admissible, if $\mathcal{M}_{e+1}$ has the infinitary outcome then $\mathbb{T}_{e+1}$ is defined from $\mathbb{T}_e$ using the `Procedure`, which is successful, and hence by Lemma 4.9 is admissible.

So suppose that $\mathcal{M}_{e+1}$ has the finitary outcome, and $\mathbb{T}_{e+1}$ is the tree $\mathbb{T}_e\{\upsquigarrow_{\sigma}^{\succ f}\}$ for some $\sigma \in \mathbb{T}_{e-1}\{\upsquigarrow_{\pi_s(\mathcal{P}_{e-1})}^{\succ\varnothing}\}$ with $\ell_{e-1}(\sigma) = \top$. The labels ℓ_e of the tree $\mathbb{T}_e^{\xi^\frown\sigma}$ are defined by setting:

- $\mathrm{scope}_e(\sigma) = 1$ and $\ell_e(\sigma) = \top$;
- for $\tau \neq \sigma$, $\mathrm{scope}_e(\tau) = \mathrm{scope}_{e-1}(\tau) - 1$ and $\ell_e(\tau) = \top$ if $\ell_{e-1}(\tau) = \top$, or $\ell_e(\tau) = \eta$ if $\ell_{e-1}(\tau) = f\eta$.

We must argue that $\mathbb{T}\{\upsquigarrow_{\sigma}^{\succ f}\}$ is admissible:

(1) Each $\sigma \in \mathbb{T}_{e+1}$ has two main children σ^* and σ^{**}, namely the same two main children of σ in $\mathbb{T}_e$. We have $\ell_{e+1}(\sigma^*) = \ell_e(\sigma^*) - 1 \geq \ell_e(\sigma) - 1 = \ell_{e+1}(\sigma)$ and similarly for σ^{**}.

(2) If σ^* is a child of σ on $\mathbb{T}_{e+1}$, then σ^* is a child of σ on $\mathbb{T}_e$ and $\mathrm{scope}_{e+1}(\sigma^*) = \mathrm{scope}_e(\sigma) - 1 \geq \mathrm{scope}_e(\sigma) - 1 = \mathrm{scope}_{e+1}(\sigma)$.

(3) For each n, each path through $\mathbb{T}_{e+1}$ is also a path through $\mathbb{T}_e$, and hence contains a node σ with $\ell_{e+1}(\sigma) = \ell_e(\sigma) = \top$ and $\mathrm{scope}_{e+1}(\sigma) = \mathrm{scope}_e(\sigma) - 1 \geq n$. □

5 Verification

In this section, we check that the A constructed above is non-computable, of minimal degree, and low for speed.

5.1 Non-computable

We chose the initial segment A_{3e+2} of A such that it differs from the eth c.e. set. Thus A is not computable.

5.2 Minimal degree

Recall that we write $\mathbb{T}_e$ for $\mathbb{T}_e^{\pi_{\mathcal{M}_e}}$, the tree produced by the true outcome of $\mathcal{M}_e$, and we sometimes write $\mathcal{M}_e$ for the instance of $\mathcal{M}_e$ acting along the true sequence of outcomes. We show that A is of minimal degree by showing that it lies on the trees $\mathbb{T}_e$ which are either e-splitting or force Φ_e^A to be partial or computable.

Lemma 5.1. *For all e, $A \in [\mathbb{T}_e]$.*

Proof. Note that for each e, A_{3e-1} is the outcome of $\mathcal{P}_{e-1}$ and so it is the root node of $\mathbb{T}_e$. Then we choose $A_{3e} \preceq A_{3e+1} \preceq A_{3e+2}$ in $\mathbb{T}_e$. Since for each $e' \geq e$, $A_{e'} \in \mathbb{T}_{e'} \subseteq \mathbb{T}_e$, the lemma follows. □

Lemma 5.2. *A is a minimal degree.*

Proof. A is non-computable. We must show that A is minimal. Suppose that Φ_e^A is total. If the outcome of $\mathcal{M}_e^{\pi_{3e}}$ is ∞, then A lies on the e-splitting tree $\mathbb{T}_e = T$ produced by `Procedure(`e`,`$A_{\mathcal{P}_{e-1}}$`,`$\mathbb{T}_{e-1}$`)` and hence $\Phi_e^A \geq_T A$. If the outcome of $\mathcal{M}_e^{\pi_{3e}}$ is σ, then A lies on $\mathbb{T}_e = \mathbb{T}_{e-1}\{\rightsquigarrow_{\sigma}^{>f}\}$ and (since Φ_e^A is total) for all $\tau_1, \tau_2 \in \mathbb{T}_{e-1}\{\rightsquigarrow_{\sigma}^{>f}\}$ and for all n,

$$\Phi_e^{\tau_1}(n)\downarrow \;\wedge\; \Phi_e^{\tau_2}(n)\downarrow \quad\longrightarrow\quad \Phi_e^{\tau_1}(n) = \Phi_e^{\tau_2}(n).$$

Thus Φ_e^A is computable. □

5.3 Low-for-speed

Our final task is to show that A is low-for-speed. Because we now have to deal with running times, we need to be a bit more precise about the construction of the trees $\mathbb{T}$ in Section 4.2. Fixing a particular set of parameters for `Procedure(`e`,`ρ`,`T_{e-1}`)`, one can check that the sth stage takes time polynomial in s and e. (If the leaves of T have been designated *waiting*, then we charge the time required to wait for $T_{e-1}[s]$ to be defined to stage $s+1, s+2, \ldots$.) In checking this, it is important to note that because all of these trees are subtrees of $\mathbb{T}_{-1}$, there are only polynomially in s many elements of each tree of length (as a binary string) at most s. Thus by dovetailing all of the procedures, we can ensure that the sth stage of each instance of `Procedure` takes time polynomial in s; the particular polynomial will depend on the parameters for `Procedure`.

As we build all of the trees $\mathbb{T}$, we keep track of them in an easy-to-query way (such as using pointers) so, e.g., querying whether an element is in a tree can be done in time polynomial in the length (as a binary string) of that element. Again, we use the fact that all of these elements are in $\mathbb{T}_{-1}$.

Now we will define the simulation procedure. Fix a lowness requirement $\mathcal{L}_e$. Define $\eta = \Delta(\pi_{\mathcal{L}}) \in \{f, \infty\}^{e+1}$; η is the sequence of guesses, f or ∞, at the outcomes of $\mathcal{M}_0, \ldots, \mathcal{M}_e$. Write $\eta_{>i}$ for the final segment $\langle \eta(i+1), \ldots, \eta(e) \rangle$ of η, the guesses at $\mathcal{M}_{i+1}, \ldots, \mathcal{M}_e$.

Write scope_i and ℓ_i for the scope and labeling function on $\mathbb{T}_i$. Let $\rho_1, \ldots, \rho_k$ be incomparable nodes on $\mathbb{T}_e$ such that (a) every path on $\mathbb{T}_e$ passes through some ρ_i, (b) each ρ_i has $\ell_e(\rho_i) = \top$, and (c) for each i and each $e' \leq e$, $\text{scope}_{e'}(\rho_i) \geq e - e'$. We can find such ρ_i because $\mathbb{T}_e$ is admissible. (Think of the ρ_i as an open cover of $\mathbb{T}_e$ by nodes whose scope, in every $\mathbb{T}_{e'}$ ($e' \leq e$), includes $\mathcal{L}_e$.) For each i, we define a simulation $\Xi_{e,i}$ which works for extensions of ρ_i. (It will be non-uniform to know which $\Xi_{e,i}$ to use to simulate A, as we will need to know which ρ_i is extended by A.) Fix i, for which we define the simulation $\Xi = \Xi_{e,i}$:

Simulation $\Xi = \Xi_{e,i}$*:* Begin at stage 0 with $\Xi(x) \uparrow$ for all x. At stage s of the simulation, for each $\sigma \in \mathbb{T}_{-1}$ with $|\sigma| < s$ and $\sigma \geq \rho_i$, check whether, for each $e' \leq e$, if $\mathbb{T}_{e'}[n]$ is the greatest level of the tree $\mathbb{T}_{e'}$ defined by stage s of the construction of the trees $\mathbb{T}$, then either:

- σ is on $\mathbb{T}_{e'}[n]$ and $\sigma \in \mathbb{T}_{e'}\{\Lsh_{\rho_i}^{\succ \eta_{>e'}}\}$, or
- σ extends a leaf σ' of $\mathbb{T}_{e'}[n]$, with $\sigma' \in \mathbb{T}_{e'}\{\Lsh_{\rho_i}^{\succ \eta_{>e'}}\}$, and:
  - ($*$) if $\mathbb{T}_{e'}$ is defined using `Procedure` ($\mathcal{M}_{e'}$ has the infinitary outcome),
  $$\text{pred}_{\text{scope}_{e'}(\sigma')}(\ell_{e'}(\sigma')) = \eta_{>e'},$$
  and σ' has at stage s been designated *waiting with main children* σ^* *and* σ^{**}, then σ extends or is extended by either σ^* or σ^{**}.

If for some σ this is true for all $e' \leq e$, then for any $k < s$ with $\Psi_s^\sigma(k) \downarrow$, set $\Xi(k) = \Psi_s^\sigma(k)$ if it is not already defined.

The idea behind the condition ($*$) is that if σ' has been designated *waiting*, this is a warning that the secondary children of σ will not be simulated by any lowness requirement with guess $\leq \eta_{>e'}$. So, if $\mathcal{L}_e$ is such a lowness requirement, and if σ is along a secondary child of σ', then we should not simulate σ.

When we say stage s of the construction of the tree $\mathbb{T}_{e'}$, we mean stage s in `Procedure` if that is how $\mathbb{T}_{e'}$ is defined, or if $\mathcal{M}_{e'}$ has a finitary outcome then that part of $\mathbb{T}_{e'}$ which can be defined from stage s in the construction of $\mathbb{T}_{e'-1}$. Recall that we can run these constructions up to stage s in time polynomial in s. Thus:

Remark 5.3. Stage s of the simulation can be computed in time polynomial in s. (The polynomial may depend on e.) This is because there are polynomially many in s nodes $\sigma \in \mathbb{T}_{-1}$ with $|\sigma| < s$.

The next series of lemmas are proved in the context above of a fixed e, with $\rho_1, \ldots, \rho_k$. If $e = \langle e_1, e_2 \rangle$, we write Ψ for Ψ_e. Fix j such that A extends ρ_j.

If $\mathcal{L}_e$ has outcome ∞, then we need Ξ_e to include the initial segments of A in the computations that it simulates. First we prove that the initial segments of A have the right labels to be simulated.

Lemma 5.4. *If* $\pi(\mathcal{L}_e) = \infty$*, for each* $e' \leq e$*,* $A \in [\mathbb{T}_{e'}\{\Lsh_{\rho_j}^{\succ \eta_{>e'}}\}]$ *for some* i*.*

Proof. Let $\sigma = A_{\mathcal{M}_e}$ be the root of $\mathbb{T}_e$. Since $\pi(\mathcal{L}_e) = \infty$, $A_{\mathcal{L}_e} = A_{\mathcal{M}_e}$. Let $\sigma^* = \xi(\mathcal{P}_e) = A_{\mathcal{P}_e}$. Then, by construction, σ^* a main child of σ. A is a path through $\mathbb{T}_{e+1}$ extending σ^*, and $\mathbb{T}_{e+1}$ is one of the following two trees, depending on the outcome of $\mathcal{M}_{e+1}$:

(1) the labeled tree T produced by `Procedure(`$e+1$`,`σ^*`,`$\mathbb{T}_e$`)`, which is a subtree of $\mathbb{T}_e\{\uparrow^{>\varnothing}_{\sigma^*}\}$; or

(2) the tree $\mathbb{T}_e\{\uparrow^{>f}_{\tau}\}$ for some $\tau \in \mathbb{T}_e\{\uparrow^{>\varnothing}_{\sigma^*}\}$ with $\ell_e(\tau) = \top$.

In either case, $A \in [\mathbb{T}_e\{\uparrow^{>\varnothing}_{\rho_j}\}]$, and $\varnothing = \eta_{>e}$. (Note that ρ_j extends σ and is extended by A.)

Now we argue backwards by induction. Suppose that $A \in [\mathbb{T}_{e'}\{\uparrow^{>\eta_{>e'}}_{\rho_j}\}]$. We want to argue that $A \in [\mathbb{T}_{e'-1}\{\uparrow^{>\eta_{>e'-1}}_{\rho_j}\}]$. We have two cases, depending on the outcome of $\mathcal{M}_e$:

- The outcome of $\mathcal{M}_e$ is ∞. Then $\eta_{>e'-1} = \infty\eta_{>e'}$ and $\mathbb{T}_{e'}$ is the labeled tree T produced by `Procedure(`e'`,`$A_{\mathcal{P}_{e'-1}}$`,`$\mathbb{T}_{e'-1}$`)`. By Lemma 4.7, given $\sigma \in \mathbb{T}_{e'}$ and $\sigma^* \in \mathbb{T}_{e'}\{\uparrow^{>\eta_{>e'}}_{\sigma}\}$, we have that $\sigma^* \in \mathbb{T}_{e'-1}\{\uparrow^{>\infty\eta_{>e'}}_{\sigma}\} = \mathbb{T}_{e'-1}\{\uparrow^{>\eta_{>e'-1}}_{\sigma}\}$. As $A \in [\mathbb{T}_e\{\uparrow^{>\eta_{>e'}}_{\rho_j}\}]$ we have $A \in [\mathbb{T}_{e'-1}\{\uparrow^{>\eta_{>e'-1}}_{\rho_j}\}]$.
- The outcome of $\mathcal{M}_e$ is f. Then $\eta_{>e'-1} = f\eta_{>e'}$ and $\mathbb{T}_{e'}$ is the tree $\mathbb{T}_{e'-1}\{\uparrow^{>f}_{\tau}\}$ for some $\tau \in \mathbb{T}_{e-1}\{\uparrow^{>\varnothing}_{A_{\mathcal{L}_{e'-1}}}\}$ with $\ell_{e-1}(\tau) = \top$. The labels on $\mathbb{T}_{e'}$ are defined so that if $\sigma \in \mathbb{T}_{e'}$, then $\ell_{e'-1}(\sigma) = f\ell_{e'}(\sigma)$ or $\ell_{e'-1}(\sigma) = \top$. Thus, as $A \in [\mathbb{T}_{e'}\{\uparrow^{>\eta_{>e'}}_{\rho_j}\}]$, and $\mathbb{T}_{e'} = \mathbb{T}_{e'-1}\{\uparrow^{>f}_{\tau}\}$, we get that $A \in [\mathbb{T}_{e'-1}\{\uparrow^{>f\eta_{>e'}}_{\rho_j}\}] = [\mathbb{T}_{e'-1}\{\uparrow^{>\eta_{>e'-1}}_{\rho_j}\}]$. □

Now we prove that if $\Psi^A(r)$ converges, then the simulation $\Xi(r)$ converges as well, though it is possible that it will have a different value if there was some other computation $\Psi^\sigma(r)$ which converged before $\Psi^A(r)$ did. Moreover, the simulation will not be too much delayed.

Lemma 5.5. *If $\pi(\mathcal{L}_e) = \infty$, and $\Psi^A(r)\downarrow$, then $\Xi(r)\downarrow$. Moreover, there is a polynomial p depending only on e such that if $\Psi^A_s(r)\downarrow$, then $\Xi_{p(s)}(r)\downarrow$.*

Proof. By the previous lemma for each $e' \leq e$, $A \in [\mathbb{T}_{e'}\{\uparrow^{>\eta_{>e'}}_{\rho_j}\}]$. Let $\sigma \in \mathbb{T}_e$ be an initial segment of A and s a stage such that $\Psi^\sigma_s(r)\downarrow$. We may assume that σ is sufficiently long that σ extends ρ_j.

Fix e' and let $\mathbb{T}_{e'}[n]$ be the greatest level of the tree $\mathbb{T}_{e'}$ defined by stage s. Then, as $A \in [\mathbb{T}_{e'}\{\uparrow^{>\eta_{>e'}}_{\rho_j}\}]$, $\sigma \in \mathbb{T}_{e'}\{\uparrow^{>\eta_{>e'}}_{\rho_j}\}$. We check the conditions (for e') from the definition of the simulation Ξ. Either $\sigma \in \mathbb{T}_{e'}\{\uparrow^{>\eta_{>e'}}_{\rho_j}\}[n]$, or some initial segment σ' of σ is in $\mathbb{T}_{e'}\{\uparrow^{>\eta_{>e'}}_{\rho_j}\}[n]$. In the second case, let us check that we satisfy $(*)$. We only need to check $(*)$ in the case that $\mathbb{T}_{e'}$ was defined using `Procedure`,

$$\operatorname{pred}_{\operatorname{scope}_{e'}(\sigma')}(\ell_{e'}(\sigma')) = \eta_{>e'},$$

and σ' has at stage s been designated *waiting with main children σ^* and σ^{**}*. Now, if $\sigma \in \mathbb{T}_e$ does not extend σ^* or σ^{**}, then it would extend a secondary child $\sigma^\dagger$ of σ' with

$$\ell_{e'}(\sigma^\dagger) \leq \operatorname{pred}_{\operatorname{scope}_{e'}(\sigma')}(\ell_{e'}(\sigma')) = \eta_{>e'}.$$

This contradicts the fact that $\sigma \in \mathbb{T}_{e'}\{\uparrow^{>\eta_{>e'}}_{\rho_j}\}$.

Since this is true for every $e' \leq e$, and $\Psi^\sigma_s(r)\downarrow$, the simulation defines $\Xi(r) = \Psi^\sigma_s(r)$ if $\Xi(r)$ is not already defined.

The simulation defines $\Xi(r)$ at the sth stage of the simulation. By Remark 5.3, the sth stage of the simulation can be computed in time polynomial in s. □

Lemma 5.5 covers the infinitary outcome of $\mathcal{L}_e$. For the finitary outcome, we need to see that any computation simulated by Ξ_e is witnessed by a computation on the tree, because the use of such a computation was not removed from the tree.

Lemma 5.6. *For all k, if $\Xi(r)\downarrow$ then there is $\sigma \in \mathbb{T}_e$, $\sigma \succeq \rho_j$, such that $\Psi^\sigma(r) = \Xi(r)$.*

Proof. Since $\Xi(r)\downarrow$, by definition of the simulation, there is a stage s of the simulation and $\sigma \in \mathbb{T}_{-1}$ with $|\sigma| < s$ such that, for each $e' \le e$, if $\mathbb{T}_{e'}[n]$ is the greatest level of the tree $\mathbb{T}_{e'}$ defined by stage s, then either:

(1) σ is on $\mathbb{T}_{e'}[n]$ and $\sigma \in \mathbb{T}_{e'}\{\upharpoonleft_{\rho_j}^{\succ\eta_{>e'}}\}$, or

(2) σ extends a leaf σ' of $\mathbb{T}_{e'}[n]$, with $\sigma' \in \mathbb{T}_{e'}\{\upharpoonleft_{\rho_j}^{\succ\eta_{>e'}}\}$, and:

    (∗) if $\mathbb{T}_{e'}$ is defined using `Procedure` ($\mathcal{M}_{e'}$ has the infinitary outcome),

$$\mathrm{pred}_{\mathrm{scope}_{e'}(\sigma')}(\ell_{e'}(\sigma')) = \eta_{>e'},$$

    and σ' has at stage s been designated *waiting with main children σ^* and σ^{**}*, then σ extends or is extended by either σ^* or σ^{**}.

$\Xi(r)$ was defined to be $\Psi^\sigma_s(r)$ for some such σ.

We argue by induction on $i \le e$ that there is a $\sigma \in \mathbb{T}_i\{\upharpoonleft_{\rho_j}^{\succeq\eta_{>i}}\}$, with the parent of σ in $\mathbb{T}_i\{\upharpoonleft_{\rho_j}^{\succ\eta_{>i}}\}$, with $\Xi(r) = \Psi^\sigma_s(r)$ and satisfying, for each e' with $i < e' \le e$, either (1) or (2). By the previous paragraph, this is true for $i = -1$. If we can show it for $i = e$, then the lemma is proved. All that is left is the inductive step. Suppose that it is true for i; we will show that it is true for $i+1$. We have two cases, depending on the outcome of $\mathcal{M}_{i+1}$.

Case 1. $\pi_{\mathcal{M}_{i+1}} = \infty$.

Since $\pi_{\mathcal{M}_{i+1}} = \infty$, $\mathbb{T}_{i+1}$ is the $(i+1)$-splitting tree defined by `Procedure(`$i+1$`,`$A_{\mathcal{P}_i}$`,`$\mathbb{T}_i$`)`. Fix σ from the induction hypothesis: $\sigma \in \mathbb{T}_i\{\upharpoonleft_{\rho_j}^{\succeq\eta_{>i}}\}$, the parent of σ is in $\mathbb{T}_i\{\upharpoonleft_{\rho_j}^{\succ\eta_{>i}}\}$, $\Xi(r) = \Psi^\sigma_s(r)$ and satisfies, for each e' with $i < e' \le e$, either (1) or (2). Since $\Psi^\sigma_s(r)$ converges, we have that $|\sigma| \le s$ and so $\sigma \in \mathbb{T}_i[s]$. Let n be the greatest level of $\mathbb{T}_{i+1}$ defined by stage s.

At stage s, either (1) σ is already on $\mathbb{T}_{i+1}\{\upharpoonleft_{\rho_j}^{\succ\eta_{>i+1}}\}[n]$, in which case we are done, or (2) σ extends a leaf σ' of $\mathbb{T}_{i+1}[n]$, with $\sigma' \in \mathbb{T}_{i+1}\{\upharpoonleft_{\rho_j}^{\succ\eta_{>i+1}}\}$, and:

(∗∗) if σ' has at stage s been designated *waiting with main children σ^* and σ^{**}* and

$$\mathrm{pred}_{\mathrm{scope}_{i+1}(\sigma')}(\ell_{i+1}(\sigma')) = \eta_{>i+1},$$

then σ extends or is extended by either σ^* or σ^{**}.

We argue in case (2).

We have $\ell_{i+1}(\sigma') \succ \eta_{>i+1}$. (If $\sigma' = \rho_j$ we use the fact that $\ell_{i+1}(\rho_j) = \top$.) Now at some stage we define the next level of the tree, $\mathbb{T}_{i+1}[n+1]$. When we do this, the children of σ' are:

- the main children σ^*, σ^{**} of σ';

- each other $\sigma^\dagger \in \mathbb{T}_i\{\upsquigarrow^{\succ\emptyset}_{\sigma'}\}[t]$, where t is the stage of the `Procedure` at which σ' was declared *waiting*. We might have $t < s$ if σ' was already declared *waiting* before stage s. In this case, $(**)$ will come into play.

We divide into two possibilities, and use $(**)$ to argue that these are the only two possibilities.

(P1) There is a child σ'' of σ' such that σ'' extends σ.

(P2) There is a child σ'' of σ', with $\ell_{i+1}(\sigma'') \succ \eta_{>i+1}$, such that σ strictly extends σ''.

Since $\sigma \in \mathbb{T}_{e'}$, σ must be compatible with (i.e., it extends or is extended by) one of the children of σ'. If we are not in case (P1), then σ extends one of the children of σ'. If σ extends one of the main children σ'' of σ', then we have $\ell_{i+1}(\sigma'') \succeq \ell_{i+1}(\sigma') \succ \eta_{>i+1}$ and are in case (P2). So the remaining possibility is that σ extends one of the secondary children σ'' of σ'. Now $\sigma'' \in \mathbb{T}_i[t]$, where t is the stage of the `Procedure` at which σ' was declared *waiting*. As $\sigma \in \mathbb{T}_i[s]$, because σ'' does not extend σ, it must be that $t < s$. Since σ properly extends σ'', and the parent of σ in $\mathbb{T}_i$ is in $\mathbb{T}_i\{\upsquigarrow^{\succ\eta_{>i}}_{\rho_j}\}$, $\sigma'' \in \mathbb{T}_i\{\upsquigarrow^{\succ\eta_{>i}}_{\rho_j}\}$. Then, looking at the construction of $\mathbb{T}_{i+1}$, $\ell_{i+1}(\sigma'') \succ \eta_{>i+1}$ unless $\mathrm{pred}_{\mathrm{scope}_{i+1}(\sigma')}(\ell_{i+1}(\sigma')) = \eta_{>i+1}$. (Recall that $\ell_{i+1}(\sigma') \succ \eta_{>i+1}$.) But then $(**)$ would imply that σ extends a secondary child of σ', which we assumed was no the case. Thus $\ell_{i+1}(\sigma'') \succ \eta_{>i+1}$. We have successfully argued that (P1) and (P2) are the only possibilities.

We begin by considering (P1). If σ'' is a main child of σ', then we have $\sigma'' \in \mathbb{T}_{i+1}\{\upsquigarrow^{\succ\eta_{>i+1}}_{\rho_j}\}$. If it is a secondary child of σ', then there may be many possible choices for σ''. We have $\sigma \in \mathbb{T}_i\{\upsquigarrow^{\succeq\eta_{>i}}_{\rho_j}\}$ and so we could have chosen $\sigma'' \in \mathbb{T}_i\{\upsquigarrow^{\succeq\eta_{>i}}_{\rho_j}\}$ (for example, we could choose σ'' to be a main child of the main child of... of σ in $\mathbb{T}_i$). Thus $\ell_{i+1}(\sigma'')$ is at least the minimum of $\eta_{>i+1}$ and $\mathrm{pred}_{\mathrm{scope}_{i+1}(\sigma')}(\ell_{i+1}(\sigma')) \succeq \eta_{>i+1}$. So $\sigma'' \in \mathbb{T}_{i+1}\{\upsquigarrow^{\succeq\eta_{>i+1}}_{\rho_j}\}$.

Since σ'' extends σ, $\Xi(r) = \Psi_s^{\sigma''}(r)$.

Fix e' with $i + 1 < e' \leq e$. Now as σ was not on $\mathbb{T}_{i+1}[n]$, it cannot be on that part of $\mathbb{T}_{e'}$ constructed by stage s. Thus (1) cannot be true for e' and σ, and so (2) must be true. Then (2) is also true for e' and the extension σ'' of σ.

Now consider (P2). There is a sequence of children $\sigma_0 = \sigma', \sigma_1 = \sigma'', \sigma_2, \ldots, \sigma_k$, with σ strictly extending σ_{k-1} and σ_k extending σ, σ_k is an initial segment of A, and with $k \geq 2$. We first claim that σ_2 is a main child of σ_1, σ_3 is a main child of σ_2, and so on, up until σ_{k-1} is a main child of σ_{k-2}. Indeed, when we begin to define $(n+2)$nd level of the tree $\mathbb{T}_{i+1}$, we do so at a stage greater than s, and so the secondary children of σ_1 are at a level at least s in $\mathbb{T}_i$ (while $\sigma \in \mathbb{T}_i[s]$). So any secondary child of σ_1 compatible with σ would extend σ, and hence be σ_k. A similar argument works for the children of σ_2, σ_3, ..., σ_{k-2}.

Thus $\ell_{i+1}(\sigma_{k-1}) \succeq \ell_{i+1}(\sigma_{k-2}) \succeq \cdots \succeq \ell_{i+1}(\sigma_1) \succ \eta_{>i+1}$. So σ_{k-1}, the parent of $\sigma_k = \sigma$, has $\sigma_{k-1} \in \mathbb{T}_{i+1}\{\upsquigarrow^{\succ\eta_{>i+1}}_{\rho_j}\}$.

Now if σ_k is a main child of σ_{k-1}, then we have $\sigma_k \in \mathbb{T}_{i+1}\{\upsquigarrow^{\succ\eta_{>i+1}}_{\rho_j}\}$. If it is a secondary child of σ_{k-1}, then there may be many possible choices for σ_k. We have $\sigma \in \mathbb{T}_i\{\upsquigarrow^{\succeq\eta_{>i}}_{\rho_j}\}$ and so we could have chosen $\sigma_k \in \mathbb{T}_i\{\upsquigarrow^{\succeq\eta_{>i}}_{\rho_j}\}$ (for example, we could choose σ_k to be a main child of the main child of... of σ in $\mathbb{T}_i$). Thus $\ell_{i+1}(\sigma_k)$ is at least the minimum of $\eta_{>i+1}$ and $\mathrm{pred}_{\mathrm{scope}_{i+1}(\sigma_{k-1})}(\ell_{i+1}(\sigma_{k-1})) \succeq \eta_{>i+1}$. So $\sigma_k \in \mathbb{T}_{i+1}\{\upsquigarrow^{\succeq\eta_{>i+1}}_{\rho_j}\}$.

Since σ_k extends σ, we have $\Xi(r) = \Psi_s^{\sigma_k}(r)$.

Fix e' with $i+1 < e' \leq e$. Now as σ was not on $\mathbb{T}_{i+1}[n]$, it cannot be on that part of $\mathbb{T}_{e'}$ constructed by stage s. Thus (1) cannot be true for e' and σ, and so (2) must be true. Then (2) is also true for e' and the extension σ_k of σ.

Case 2. $\pi_{\mathcal{M}_{i+1}} = \tau$.

We have that $\sigma \in \mathbb{T}_i\{\wr^{\eta_{>i}}_{\rho_j}\}$. $\mathbb{T}_{i+1}$ is the tree $\mathbb{T}_i\{\wr^{\succ f}_{\tau}\}$ for some $\tau \in \mathbb{T}_i\{\wr^{\succ \varnothing}_{\pi_{\mathcal{P}_i}}\}$. The labels ℓ_{i+1} of the tree $\mathbb{T}_{i+1}$ are defined from the labels ℓ_i of the tree $\mathbb{T}_i$ by setting $\ell_{i+1}(\sigma) = \top$ if $\ell_i = \top$, or $\ell_{i+1}(\sigma) = \eta$ if $\ell_i = f\eta$.

Note that since $\pi_{\mathcal{M}_{i+1}} = \tau$ is the finitary outcome, $\eta_{>i}$ begins with f. Thus σ is still on $\mathbb{T}_{i+1}$. Moreover, for each $\tau' \in \mathbb{T}_{i+1}$, $\ell_i(\tau') \geq \eta_{>i}$ if and only if $\ell_{i+1}(\tau') \geq \eta_{i+1}$. So σ is on $\mathbb{T}_{i+1}\{\wr^{\succ \eta_{>i+1}}_{\rho_j}\}$. □

We now show how to use these lemmas to prove that A is low-for-speed.

Lemma 5.7. *A is low-for-speed.*

Proof. Given $\langle e, i\rangle$, suppose that $\Psi_e^A = R_i$ and that $\Psi_e^A(n)$ is computable in time $t(n)$. We must show that R_i is computable in time $p(t(n))$ for some polynomial p. Note that the outcome of $\mathcal{L}_{\langle e,i\rangle}$ must be ∞, as otherwise we would have ensured that $\Psi_e^A \neq R_i$. Let j be such that A extends the ρ_j from the simulation for e. So by Lemma 5.5 $\Xi_{\langle e,i\rangle,j}$ is total and there is a polynomial p depending only on $\langle e, i\rangle$ such that if $\Psi_e^A(n)$ is computed in time s, then $\Xi_{\langle e,i\rangle,j}(n)$ is computed in time $p(s)$.

Now we argue that $\Xi_{\langle e,i\rangle,j}$ computes $R_i = \Psi_e^A$. Suppose not; then there is n such that $\Xi_{\langle e,i\rangle,j}(n) \neq R_i(n) = \Psi_e^A(n)$. Since $\Xi_{\langle e,i\rangle,j}(n)$ does in fact converge, by Lemma 5.6 there is $\sigma \in \mathbb{T}_{\langle e,i\rangle}$ extending ρ_j such that $\Psi_e^\sigma(n) = \Xi_{\langle e,i\rangle,j}(n) \neq R_i(n)$. This contradicts the fact that the outcome of $\mathcal{L}_{\langle e,i\rangle}$ is ∞, as we would have chosen τ as the outcome. □

References

[ABK06] Eric Allender, Harry Buhrman, and Michal Koucký. What can be efficiently reduced to the Kolmogorov-random strings? *Ann. Pure Appl. Logic*, 138(1-3):2–19, 2006.

[AFG13] Eric Allender, Luke Friedman, and William Gasarch. Limits on the computational power of random strings. *Inform. and Comput.*, 222:80–92, 2013.

[Bay12] Robertson Edward Bayer. *Lowness for Computational Speed*. ProQuest LLC, Ann Arbor, MI, 2012. Thesis (Ph.D.)–University of California, Berkeley.

[BD18] Laurent Bienvenu and Rodney Downey. On low for speed oracles. In *35th Symposium on Theoretical Aspects of Computer Science*, volume 96 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 15, 13. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the $\mathcal{P} =?\mathcal{NP}$ question. *SIAM J. Comput.*, 4(4):431–442, 1975.

[CDE+14] Mingzhong Cai, Rodney G. Downey, Rachel Epstein, Steffen Lempp, and Joseph S. Miller. Random strings and truth-table degrees of Turing complete c.e. sets. *Log. Methods Comput. Sci.*, 10(3):3:15, 24, 2014.

[SS91] Theodore A. Slaman and Robert Solovay. When oracles do not help. In Manfred K. Warmuth and Leslie G. Valiant, editors, *COLT*, page 379–383. Morgan Kaufmann, Oxford, 1991.