The Journal of Symbolic Logic
Volume 87, Number 4, December 2022

A MINIMAL SET LOW FOR SPEED

ROD DOWNEY AND MATTHEW HARRISON-TRAINOR

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 et al. [3] 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 [1] and with Friedman and Gasarch [2] showed that oracle access to sets of random strings can give insight into basic complexity questions. In [2] Allender et al. 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 "∩COMP" was removed by Cai et al. [6]. 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 [7] characterized languages L where oracles are of no help in Gold-style learning theory: $EX^L = EX$ if and only if 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

Received November 18, 2020.
2020 *Mathematics Subject Classification*. 03D28, 68Q15.
Key words and phrases. minimal degree, low for speed.

0022-4812/22/8704-0016
DOI:10.1017/jsl.2021.108

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 [4]). *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 [5], 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*?

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

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 proof is a complicated construction in which the interaction between different requirements is quite involved. The main tension is between the minimality requirements, which might want to wait a long time looking for an e-split while building an e-splitting tree, and the lowness requirements, which need to make decisions very quickly in order to have only a polynomial-time delay. The basic strategies for meeting the requirements are not too difficult, but there are some very intricate complications that arise when trying to resolve this, some of which only show up when four requirements interact.

The proof is similar to a forcing construction, in the sense that we meet the minimality requirements one-by-one by placing the set A on more and more refined computable trees. Like a forcing construction, these trees are not uniformly computable; the choice of trees depends on the outcomes of the minimality requirements. (Once could probably explicitly define a forcing poset, but the trees are so complicated that this would just add more complexity to the construction.) The reader might wonder whether the proof could be simplified by instead using a full approximation construction. We think that such a construction would not be simpler; a full approximation construction would need the same devices used in our proof (as they deal with complexity inherent in the combinatorics of the problem), but would also have extra complexity purely for keeping track of the approximations.

§2. The requirements and basic strategies. We will construct a set A meeting the following requirements:

$$\begin{aligned} \mathcal{M}_e: &\quad \text{If } \Phi_e^A \text{ is total then it is either computable or computes } A. \\ \mathcal{L}_{\langle e,i\rangle}: &\quad \text{If } \Psi_e^A = R_i \text{ is total and computable in time } t(n), \text{ then it is} \\ &\quad \text{computable in time } p(t(n)) \text{ for some polynomial } p. \\ \mathcal{P}_e: &\quad A \neq W_e. \end{aligned}$$

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.

2.1. Meeting one $\mathcal{P}$ requirement. To meet the requirement $\mathcal{P}_e$, we just need to choose an initial segment of A which ensures that $A \neq W_e$. There is no dynamic action to take.

2.2. Meeting one $\mathcal{M}$ requirement. 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 to allow a node to have more than two children; 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 be perfect and have no dead ends, and in fact every node will have at least two children. Moreover, the trees will be what one might call *strongly computable*: for each node

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

on the tree, one can compute a complete finite list of all of its children. As usual, $[T]$ denotes the collection of paths through T. For a Turing 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. Note that this definition is slightly weaker than the usual definition of e-splitting (which requires all children of a node to e-split with each other), but because the trees are strongly computable it is still the case that for an e-splitting tree T and $B \in [T]$, if Φ_e^B is total then $\Phi_e^B \geq_T B$.

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 computable tree T with either:

(1) T is e-splitting, and so for any path $B \in [T]$, if Φ_e^B is total then $\Phi_e^B \geq_T B$; or
(2) 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}. 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 non-computable and 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.

Our construction will be much more complicated, as while building the splitting trees we will need to take into account the lowness requirements. Nevertheless, the basic idea is the same: either we can find "enough" e-splits, and put A on an e-splitting tree, or above some node we can find a subtree with no e-splits which forces that A is either partial or computable. Sometimes, in addition to options (1) and (2) above, we will sometimes choose to put A on a tree that forces Φ_e^A to be partial:

(3) there is an x such that for all paths $B \in [T]$, $\Phi_e^B(x)\uparrow$.

2.3. Meeting one $\mathcal{L}$ requirement. We use something similar to the Slaman–Beyer strategy from [4] to meet the lowness requirements. 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

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

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

where each $a_i \in \{0, 1\}$. So, for example, $100111100000000 \in T_{-1}$. In this tree, there are 2^n nodes of height 2^n on the nth level of the tree.

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 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. Consider two cases:

(1) If x is such that $\Xi(x)\downarrow \neq R(x)\downarrow$, then there is some $\sigma \in T_{-1}$ witnessing that $\Psi^\sigma(x) = \Xi(x)$. In this case the requirement $\mathcal{L}$ asks that A extend σ, so that $\Psi^A(x) \neq R(x)$ and $\mathcal{L}$ is satisfied. This is the finitary outcome.

(2) If there is no x such that $\Xi(x)\downarrow \neq R(x)\downarrow$, then if Ξ and R both total, then $\Xi = R$. So either $\mathcal{L}$ is vacuously satisfied, or $\Xi = \Psi^A = R$. The is the infinitary outcome.

In the second case we need to make sure that Ξ is only polynomially slower than Ψ^A. We can do this by appropriately arranging 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 will define $\Xi(x) = \Psi^\sigma(x)$ if it is not already defined, so that $\Xi(x)\downarrow$ in time which is only polynomially slower than $t(x)$. It is important here that T_{-1} has only polynomially many nodes at height n and that we can test membership in T_{-1} in polynomial time. (If T_{-1} was, for example, the full binary tree, then there would be 2^n different finite oracles σ of length n for which we must simulate computations $\Psi^\sigma(x)$; it might be that one of these computations converges in time $\leq n$, but that it takes time exponential in n before we get to simulating this computation and defining $\Xi(x)$.)

Think of the simulations Ξ 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 get a diagonalization $\Phi^A \neq R$ or to force that either the simulation Ξ is actually correct.

In the rest of the paper, we will sometimes say that $\mathcal{L}$ *simulates the computation* $\Psi^\sigma(x)$, that $\mathcal{L}$ *simulates* σ, or that $\mathcal{L}$ *simulates computations above* σ. These are informal notions which will be helpful to explain the construction; the simulation process is defined more formally in Section 8.3. When we say that $\mathcal{L}$ *simulates the computation* $\Psi^\sigma(x)$, what we mean is that $\mathcal{L}$ computes $\Psi^\sigma(x)$ and, if it converges and $\Xi(x)$ is so far undefined, sets $\Xi(x) = \Psi^\sigma(x)$. When we say that $\mathcal{L}$ *simulates* σ, what we mean is that $\mathcal{L}$ simulates the computations $\Psi^\sigma(x)$ for $x \leq |\sigma|$, setting $\Xi(x) = \Psi^\sigma(x)$ if possible. Finally, when we say that $\mathcal{L}$*simulates computations above* σ, we mean that $\mathcal{L}$ simulates computations $\Psi^\tau(x)$ for various τ extending σ, and sets $\Xi(x) = \Psi^\tau(x)$ whenever it finds such a computation for which $\Xi(x)$ has not yet been defined. We will also sometimes say that $\mathcal{L}$ *stops simulating computations above* σ.

CONVENTION 2.1. *When computing* $\Phi^\sigma(x)$, *the computation may run for only* $|\sigma|$ *many steps and we require that* $x < |\sigma|$.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

2.4. Meeting one $\mathcal{M}$ requirement and one $\mathcal{L}$ requirement. The interactions between the requirements get more complicated. Consider now two requirements $\mathcal{M} = \mathcal{M}_e$ and $\mathcal{L}$, with $\mathcal{M}$ is of higher priority than $\mathcal{L}$. Write $\Phi = \Phi_e$. $\mathcal{L}$ will work with Ψ, R, and Ξ.

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 partial or 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_1, τ_2 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 $T \subseteq T_{-1}$. (Of course in general $\mathcal{M}$ must build an e-splitting subtree of the tree produced by some other minimality requirement, which will introduce additional complexity; we will deal with this later.)

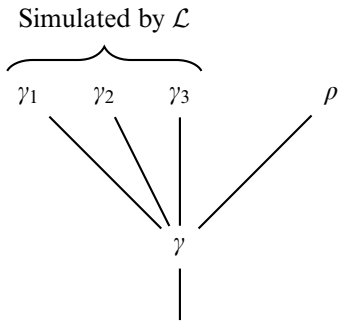
While $\mathcal{M}$ is building the tree, $\mathcal{L}$ will be simulating various computations Ψ^{σ}. The tree T might be built very slowly, while $\mathcal{L}$ has to simulate computations relatively quickly in order to define Ξ with only a polynomial delay. So when a node is removed from T, $\mathcal{L}$ will stop simulating computations above it; but $\mathcal{L}$ will have to simulate computations Ψ^{σ} for nodes σ 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 $\mathcal{L}$ 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 $\gamma_1, \gamma_2 \succeq \gamma$ and wants to set γ_1 and γ_2 to be the successors of γ on T, with both γ_1 and γ_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 $\Psi^A = R$ (say, 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, and that $\mathcal{L}$ is simulating computations above γ (i.e., γ is an *observation node* for $\mathcal{L}$, to be defined). The tree is supposed to be an e-splitting tree, so we need to look for e-splits above γ. While looking for e-splits, we also need to simulate computations by Ψ. It might be that, perhaps before finding any e-splits at all, we simulate a computation $\Psi_t^{\rho}(y)\downarrow$ and set $\Xi(y)$ to be equal to this simulated computation. For the sake of $\mathcal{L}$ we must keep ρ on the tree. (Soon we will extend the strategy to worry about what happens if we have found multiple such computations, but for now assume that there is just one.) Eventually we will find some e-splits, but this might be much later; in particular, we 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 $x_{i,j}$ such that $\Phi_e^{\gamma_i}(x_{i,j})\downarrow \neq \Phi_e^{\gamma_j}(x_{i,j})\downarrow$.

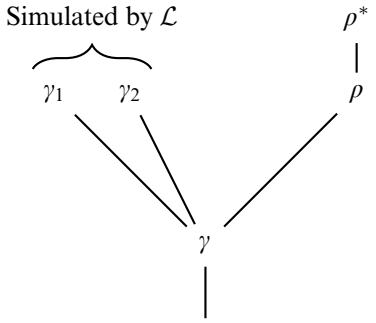
To begin, we stop simulating any computations extending ρ. This means that we are now free to extend the tree however we like above ρ without worrying about

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

how this affects the $\mathcal{L}$-simulations. We also stop simulating any other computations not compatible with γ_1, γ_2, or γ_3. We keep simulating above γ_1, γ_2, and γ_3 with the expectation that, in the infinitary outcome where $\Xi = R$, A will extend one of these.



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 $x_{i,j}$ witnessing the e-splitting of γ_1, γ_2, and γ_3, e.g., if $\Phi^{\gamma_i}(x_{i,j}) \neq \Phi^{\gamma_j}(x_{i,j})$, and $\Phi^{\rho^*}(x_{i,j})\downarrow$, then ρ^* must e-split with either γ_i or γ_j. 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^* and stop simulating computations above γ_3.



So of the extensions of γ, some are still simulated by $\mathcal{L}$, and others are not. Recall that $\Psi^{\rho}(y)$ converged, and we defined $\Xi(y) = \Psi^{\rho}(y)$. If $R(y) \neq \Psi^{\rho}(y)$ then $\mathcal{L}$ will insist that A extend ρ^* to diagonalize; this is the finitary outcome of $\mathcal{L}$. Since in this case $\mathcal{L}$ is satisfied with the finitary outcome by having $\Psi^A \neq R$, computations extending ρ^* do not have to be simulated. We call ρ^* a *diagonalization node* (*for* $\mathcal{L}$); it is kept on the tree so that we can, if needed, meet the requirement $\mathcal{L}$ with the finitary outcome. On the other hand, if $\mathcal{L}$ fails to diagonalize and has the infinitary outcome, then A will need to extend the nodes γ_1 and γ_2 that are simulated. (If A extended ρ^*, then it might be that $\Psi^A(z)\downarrow$ using an oracle extending ρ^*, but $\Xi(z)$ was not defined because $\mathcal{L}$ did not simulate this computation.) We say that γ_1 and γ_2 are *observation nodes* (for $\mathcal{L}$); by this we mean that computations above them are still simulated by $\mathcal{L}$. We need two observation children γ_1 and γ_2 rather than just one in order to meet the requirements $\mathcal{P}$. The terms *diagonalization node* and *observation*

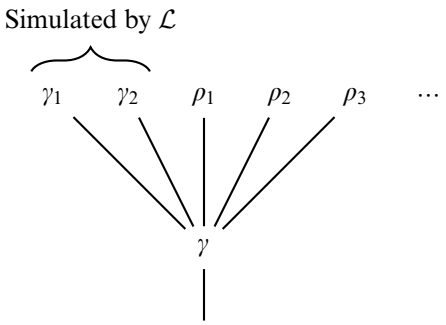
https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

node will not be used formally in the construction, but they will make explaining the construction easier.

We also note that we assumed that $\mathcal{L}$ was still simulating computations above γ at the start of the construction. What we mean is that γ itself was an observation node for $\mathcal{L}$, as was its parent, and so on, back to some node which $\mathcal{L}$ knows, via its guesses at the higher priority requirements, is an initial segment of A.

There are still some gaps in the above strategy that we must fix. What if, while looking for ρ^*, we simulate a computation $\Psi^{\gamma_3}(z)\downarrow$, and set $\Xi(z) = \Psi^{\gamma_3}(z)$, 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-split.

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 for which we have found computations $\Psi^{\rho_i}(y_i)\downarrow$ and set $\Xi(y_i) = \Psi^{\rho_i}(y_i)$. Like ρ above, we must keep $\rho_1, \ldots, \rho_n$ on the tree.[1] We stop simulating computations above $\rho_1, \ldots, \rho_n$.



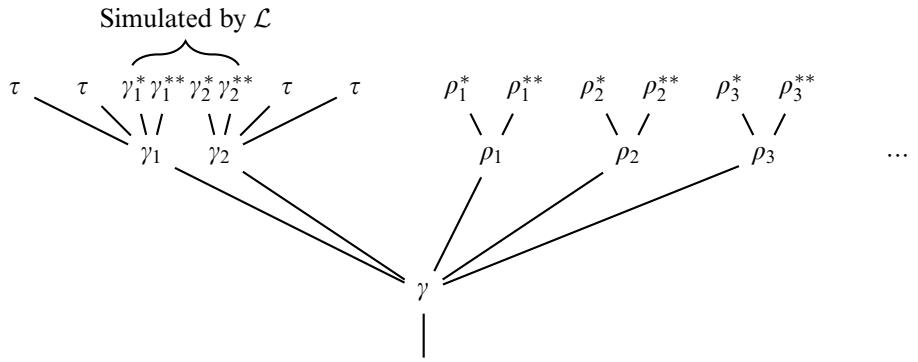
The nodes γ_1 and γ_2 are observation nodes for $\mathcal{L}$ and the ρ_i are diagonalization nodes for $\mathcal{L}$.

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, $\mathcal{L}$ must simulate more computations at nodes τ above γ_1 and γ_2 and for some of them we might see that $\Psi^\tau(y)\downarrow$ for some y and set $\Xi(y) = \Psi^\tau(y)$. Just as we had to ensure that $\rho_1, \rho_2, \ldots$ remained on the tree, we also have to ensure that these τ remain

[1] Suppose that we find γ_1, γ_2 at stage s of the construction. It might even be that $\rho_1, \ldots, \rho_n$ are all of the other nodes on T_{-1} that we have considered by stage s. Our construction will be able to handle this case, so there is no reason not to just assume that we must keep all of these nodes on the tree. This saves the bookkeeping of remembering which nodes ρ have *actually* been the use of a computation $\Psi^\rho(x)\downarrow$ for which we defined $\Xi(x)$; we just keep on the tree anything that *might* have had a computation converge.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

on the tree. On the other hand, we no longer simulate computations above the ρ_i, and so we can just look for e-splits above each ρ_i without having to worry about $\mathcal{L}$.



Here γ_1^*, γ_1^{**}, γ_2^*, and γ_2^{**} are observation nodes for $\mathcal{L}$, since we still simulate computations above them. The τ are diagonalization nodes for $\mathcal{L}$. The ρ_i^* and ρ_i^{**} are neither observation nodes nor diagonalization nodes for $\mathcal{L}$ (though of course they are extensions of diagonalization nodes).

Now at the next step of extending the tree we need to extend all of these nodes to extensions that e-split with each other; but in doing so we will introduce further extensions (above γ_1^*, γ_1^{**}, γ_2^*, and γ_2^{**}) 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* will e-split with each other.

§3. Outcomes and the general structure of the argument. Now that we have seen the basic strategies we return to talking about the priority ordering and the outcomes of the requirements. 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 $\mathcal{P}_e$. For

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

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

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

Just as in the standard forcing construction of a minimal degree, we will meet the $\mathcal{M}$ requirements one-by-one by asking, non-effectively for each e, whether there are enough e-splits on the tree T_{e-1} built by $\mathcal{M}_{e-1}$. If there are, then we build an e-splitting tree (by a complicated procedure taking into account the lower priority $\mathcal{L}$ requirements), and if not, then we can find a tree forcing that Ψ_e is partial or computable. Let T_e be the e-splitting tree in the former case, or the tree forcing that Ψ_e is partial or computable in the latter case. In the former case, $\mathcal{M}_e$ has the infinitary outcome, and in the latter case the finitary outcome. Having constructed $T_{-1}, \dots, T_e$, we non-effectively determine the outcomes of $\mathcal{L}_e$ and $\mathcal{P}_e$. (It is important here that $T_{-1}, \dots, T_e$ were specially constructed taking into account $\mathcal{L}_e$.) Then, having determined these outcomes, we turn to $\mathcal{M}_{e+1}$. The outcome of each requirement determines some initial segment of A, giving A as the limit of these extensions.

When $\mathcal{M}_e$ is constructing T_e and taking into account the lower priority $\mathcal{L}$ requirements, say some particular one $\mathcal{L}$, $\mathcal{M}_e$ does not know the outcomes of the intermediate requirements of lower priority than $\mathcal{M}_e$ but of higher priority than $\mathcal{L}$. $\mathcal{L}$, on the other hand, does know these outcomes. So $\mathcal{M}_e$ will have to consider, for each guess η by $\mathcal{L}$ at the higher priority outcomes, an *instance* $\mathcal{L}^\eta$ of $\mathcal{L}$ which makes this guess. Of course, once we come to actually meeting $\mathcal{L}$, there will only be one instance taking action, namely the one corresponding to the true outcomes of the higher priority requirements, but $\mathcal{M}_e$ does not know which this is and so has to consider all possible instances.[2]

The formal construction will be somewhat "static" in the sense that while $\mathcal{M}_e$ is constructing T_e, it is not actually running, in a dynamic way, the simulation procedures defining Ξ for the low priority $\mathcal{L}$ requirements. (See footnote 1.) Nevertheless it will be helpful to think of a dynamic construction for now, to see what kinds of interactions there are between the requirements; then in Section 5 we will develop a labeling system capturing these interactions in a static way. The e-splitting part of the construction will of course always be dynamic.

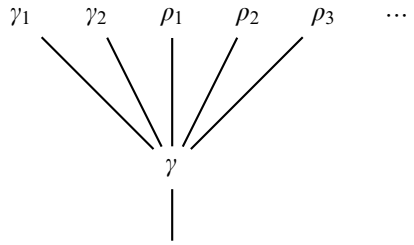
[2] In seeing the complexity of the construction, the reader might wonder whether things can be simplified by using a full approximation argument. The answer, we believe, is that a full approximation argument would only add more complexity to the construction, in that we would have to dynamically guess at the outcomes of the minimality requirements, while not simplifying the combinatorics involved in meeting the lowness requirements. Fix a minimality requirement $\mathcal{M}$. Because a full approximation argument can never be sure $\mathcal{M}$ has the finitary outcome (as we might always find more splits), the instances of lowness requirements guessing that $\mathcal{M}$ has the infinitary outcome must keep simulating computations even when we think that $\mathcal{M}$ has the finitary outcome. Thus all of the complicated interactions between lowness requirements soon to be described would still have to be considered.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

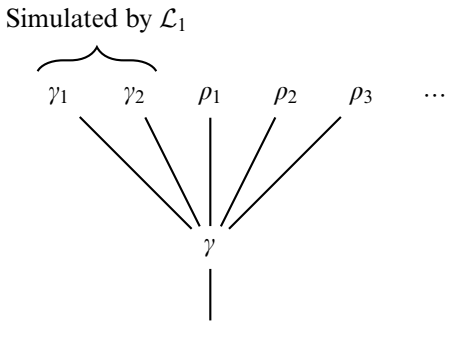
§4. Interactions between three or more requirements. We now need to consider in more detail the interactions between the requirements. We saw in the previous section that an $\mathcal{M}$ requirement must take into account lower priority $\mathcal{L}$ requirements. In the full construction, we will have 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.

4.1. One $\mathcal{M}$ requirement, two $\mathcal{L}$ requirements. Consider three requirements: $\mathcal{M} = \mathcal{M}_e$ of highest priority, $\mathcal{L}_0$ of middle priority, and $\mathcal{L}_1$ of lowest priority. Write Ψ_0, R_0, and Ξ_0 and Ψ_1, R_1, and Ξ_1 for the functionals and sets corresponding to $\mathcal{L}_0$ and $\mathcal{L}_1$ respectively. Suppose that both $\mathcal{L}_0$ and $\mathcal{L}_1$ correctly guess that $\mathcal{M}$ has the infinitary outcome, building a splitting tree. Suppose also that $\mathcal{L}_1$ correctly guesses that $\mathcal{L}_0$ has the infinitary outcome, which means that if $\Psi_0^A = R_0$ is total, then we must have $\Xi_0 = \Psi_0^A$; this means that $\mathcal{L}_0$ must simulate every initial segment of A, i.e., they must all be observation nodes for $\mathcal{L}_0$.

Suppose that the requirement $\mathcal{M}$ is trying to extend a node γ. We use the same strategy as before to extend γ, keeping certain nodes $\rho_1, \rho_2, \ldots$ on the tree. These ρ_i might be kept on the tree either for the requirement $\mathcal{L}_0$ (because we defined $\Xi_0(y_i) = \Psi_0^{\rho_i}(y_i)$) or for the requirement $\mathcal{L}_1$ (because we defined $\Xi_1(z_i) = \Psi_1^{\rho_i}(z_i)$). Indeed, for each ρ_i, it might be that both requirements want to keep ρ_i on the tree.



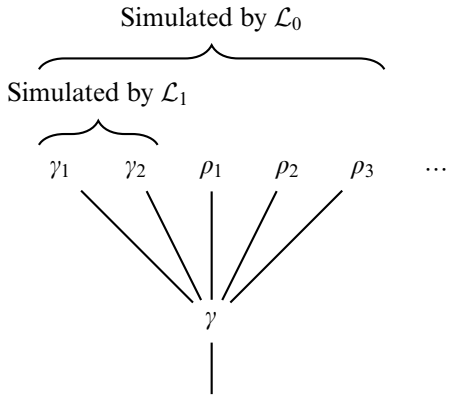
The requirement $\mathcal{L}_1$ has guessed that the outcome of $\mathcal{L}_0$ is the infinitary outcome, which means that $\mathcal{L}_0$ does not need to force A to extend one of $\rho_1, \rho_2, \ldots$. This means that $\mathcal{L}_1$ can use the same strategy as in the previous section with only one lowness requirement, and stop simulating computations above the ρ_i.



Now depending on the outcome of $\mathcal{L}_1$ it might force A to extend any one of these nodes. (For example, if $\mathcal{L}_1$ has the finitary outcome ρ_i, then A must extend ρ_i in order to obtain a diagonalization $\Psi_1^A \neq R_1$; and if $\mathcal{L}_1$ has the infinitary outcome, then A will extend one of γ_1 or γ_2.) $\mathcal{L}_0$ must simulate initial segments of A, but it

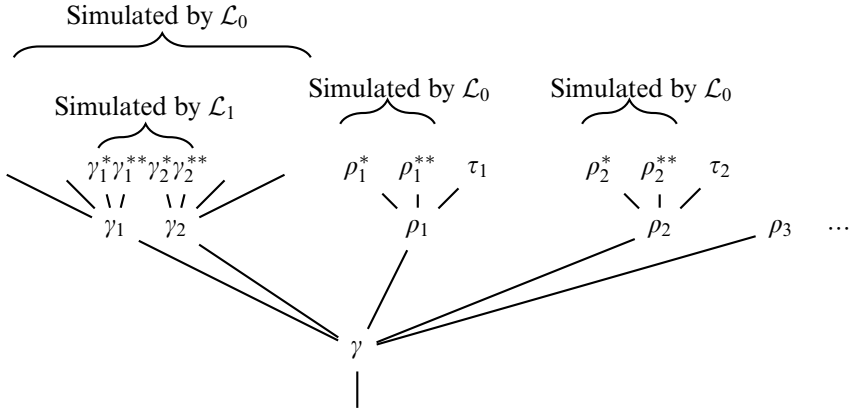
https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

does not know the outcome of $\mathcal{L}_1$ which is a lower priority requirement. So $\mathcal{L}_0$ must simulate computations through all of these nodes.



Here γ_1 and γ_2 are the observation nodes for $\mathcal{L}_1$, while the ρ_i are diagonalization nodes for $\mathcal{L}_1$. All of γ_1, γ_2, and the ρ_i are observation nodes for $\mathcal{L}_0$. All of the observed and diagonalization nodes for $\mathcal{L}_1$ are observation nodes for $\mathcal{L}_0$.

Now in the next step we found extensions γ_1^* and γ_1^{**} of γ_1, γ_2^* and γ_2^{**} of γ_2, and ρ_1^* and ρ_1^{**} of ρ_1 that e-split with each other. Before, we could simply extend ρ_1 to ρ_1^* and ρ_1^{**}. Now, while we are looking for the extensions, $\mathcal{L}_0$ might simulate other computations, say τ_1 extending ρ_1, τ_2 extending ρ_2, and so on. If $\mathcal{L}_0$ sees that $\Psi_0^{\tau_i}(z_i)\downarrow$ and sets $\Xi_0(z_i) = \Psi_0^{\tau_i}(z_i)$, then τ_i must be kept on the tree. (Of course, in general, each ρ_i might have many such nodes extending it.) Above each ρ_i, we are essentially in the case from the previous section of having only one $\mathcal{L}$ requirement; so as before, $\mathcal{L}_0$ stops simulating above the τ_i but continues simulating above the ρ_i^* and ρ_i^{**}. Above γ_1 and γ_2, we do the same thing that we did above γ.



Here γ_1^*, γ_1^{**}, γ_2^*, and γ_2^{**} are observation nodes for $\mathcal{L}_1$, while the other (unnamed) children of γ_1 and γ_2 are diagonalization nodes for $\mathcal{L}_1$. The ρ_i^*, ρ_i^{**}, and τ_i are

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

neither observation nodes nor diagonalization nodes for $\mathcal{L}_1$ but they extend the diagonalization nodes ρ_i. All of the children of γ_1 and γ_2, together with the ρ_i^* and ρ_i^{**}, are observation nodes for $\mathcal{L}_0$. The τ_i are diagonalization nodes for $\mathcal{L}_0$. Again, all of the observed and diagonalization nodes for $\mathcal{L}_1$ are observation nodes for $\mathcal{L}_0$.

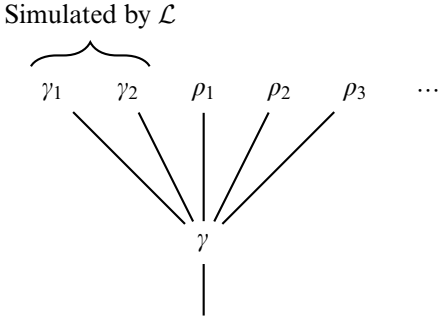
Note that, for example, τ_1 may not e-split with γ_1^*. This is because we were not able to choose ρ_1 or τ_1; they were both forced onto the tree by the lowness requirements. But we have still made some progress: computations above τ_1 are not being simulated, and so when we extend τ_1, we can find extensions that e-split with the e-splitting extensions of the other nodes.

Note that the number of levels we wait to find e-splits is now two, since there are two lowness requirements, whereas in Section 2.4 we only had to wait one level. In general, the more lowness requirements we are considering, the longer we have to wait before we can find e-splits along all paths. So it is important that we only deal with finitely many lowness requirements at a time, so that we eventually arrive at a point where parts of the tree are 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*; see Section 5.)

Note also that every node simulated by $\mathcal{L}_1$ is also simulated by the lower priority requirement $\mathcal{L}_0$, but that there are nodes simulated by $\mathcal{L}_0$ which are not simulated by $\mathcal{L}_1$. Moreover, every diagonalization node for $\mathcal{L}_1$ is an observation node for $\mathcal{L}_0$ and so the nodes above diagonalization nodes for $\mathcal{L}_1$ are simulated by $\mathcal{L}_0$. This is the first example of a relationship between the nodes simulated by one requirement and the nodes simulated by another requirement. There will be further such relationships as well.

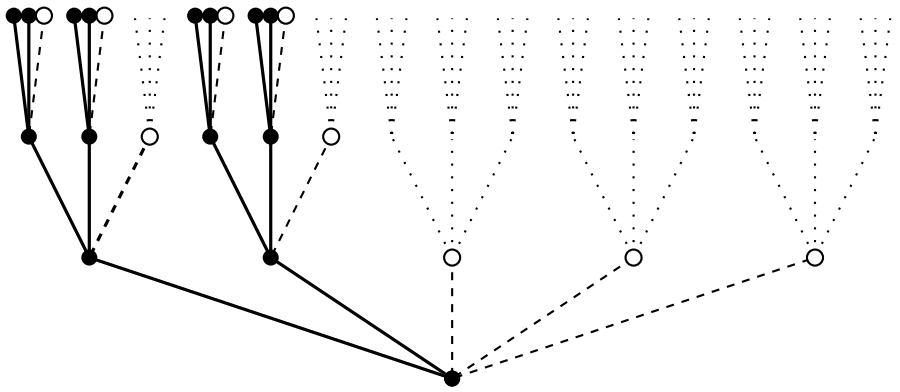
4.2. Two $\mathcal{M}$ requirements and one $\mathcal{L}$ requirement. Consider three requirements: $\mathcal{M}_0$ of highest priority, $\mathcal{M}_1$ of middle priority, and $\mathcal{L}$ of lowest priority. Suppose that $\mathcal{M}_0$ is trying to build a 0-splitting tree T_0, and that $\mathcal{M}_1$ and $\mathcal{L}$ correctly guess that it is successful in doing so and has the infinitary outcome. Suppose further that $\mathcal{M}_1$ is trying to build a 1-splitting tree T_1, and $\mathcal{L}$ is working with Ξ, Ψ, and R.

When $\mathcal{M}_0$ is building T_0 we follow the same strategy as before. When extending a node γ which is simulated by $\mathcal{L}$, we have two children which are observation nodes, and a number of other children which are diagonalization nodes.



https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

So we can think of the tree T_0 as a finitely branching tree which has, embedded in it, a subtree of nodes which are observation nodes for $\mathcal{L}$; each such node $\mathcal{L}$ has at least two children which are themselves observation nodes for $\mathcal{L}$. We can picture the tree T_0 as looking something like this. We show the observation nodes with filled in circles, and the diagonalization nodes with open circles.



The children along bold lines are observations nodes which are children of observation nodes and so on back to the root node.[3]

4.2.1. $\mathcal{M}_1$ has the infinitary outcome. Suppose that $\mathcal{M}_1$ succeeds in building a 1-splitting subtree T_1 of T_0, and that $\mathcal{L}$ correctly guesses this.

Since $\mathcal{L}$ is of lower priority than $\mathcal{M}_1$, $\mathcal{M}_1$ does not know the outcome of $\mathcal{L}$. So when $\mathcal{L}$ simulates a computation $\Psi^{\rho}(x)\downarrow$ and sets $\Xi(x) = \Psi^{\rho}(x)$, $\mathcal{M}_1$ must keep ρ on the tree T_1. That is, $\mathcal{M}_1$ must keep the diagonalization nodes for $\mathcal{L}$ on T_1. $\mathcal{L}$ will have the finitary outcome and have A extend one of these ρ if, by doing so, it can ensure that $\Psi^{A} \neq R$.

Now suppose that $\mathcal{L}$ has the infinitary outcome, though of course $\mathcal{M}_1$ does not know this. Then:

(1) Because $\mathcal{L}$ has the infinitary outcome, $\mathcal{L}$ must simulate all of the initial segments of A, i.e., the initial segments of A must be observation nodes on T_0.
(2) On the other hand, to meet $\mathcal{M}_1$, A must be on the 1-splitting tree T_1.

Every path through T_1 contains, as initial segments, many nodes which are part of a 1-split. In order to reconcile (1) and (2) it must be that when $\mathcal{M}_1$ is looking for 1-splits above an observation node for $\mathcal{L}$, it cannot look for just any 1-split; it should look through extensions that are also observation nodes for $\mathcal{L}$. In the picture of T_0 above with the bold lines and dashed lines, this means that when $\mathcal{M}_1$ is looking for a 1-split above a bold node, it should look through the bold subtree. (Of course in building T_1 the requirement $\mathcal{M}_1$ must also implement the rest of the strategy given above, keeping certain nodes ρ on the tree, etc.)

[3]As we saw in Section 4.1, when considering multiple $\mathcal{L}$ requirements, an observation node for a requirement may have more than two children which are observation nodes for that requirement.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

4.2.2. $\mathcal{M}_1$ has the finitary outcome. Suppose that $\mathcal{M}_1$ fails to build a 1-splitting subtree of T_0 and that $\mathcal{L}$ correctly guesses this. This means that there is some node γ on T_0 such that $\mathcal{M}_1$ cannot find a 1-split extending γ, or it cannot find a convergence of Ψ on some input above γ. In the standard construction of a minimal degree, we would then define a tree T_1^* as the full subtree of T_0 below γ; T_1^* forces that Φ_1^A is partial or computable for every $A \in [T_1^*]$. However, we just saw that $\mathcal{M}_1$ might not look for 1-splits through all of the extensions of γ on T_0, but just through some subtree of T_0 above γ. Then we must define T_1^* not to be the full subtree of T_0 above γ, but rather to be the subtree of T_0 above γ where we $\mathcal{M}_1$ looked for (and failed to find) a 1-split.

We meet $\mathcal{M}_1$ by having A be a path through T_1^*. Since $\mathcal{L}$ correctly guessed that $\mathcal{M}_1$ has the finitary outcome, γ, we must ensure that $\mathcal{L}$ is satisfied. When we defined T_0, there were certain nodes ρ (the diagonalization nodes for $\mathcal{L}$) for which $\mathcal{L}$ found computations $\Psi_1^\rho(x)\downarrow$ and set $\Xi_1(x) = \Psi_1^\rho(x)\downarrow$. $\mathcal{M}_0$ was required to ensure that these nodes ρ stayed on the tree T_0. Similarly, these nodes ρ must stay on the tree T_1^*; if they were not on T_1^*, then we would lose the opportunity to achieve diagonalizations $\Psi^A \neq R$. However, the tree T_1^* is not being defined dynamically, so we cannot take some dynamic action to keep these nodes on the tree. Instead, we must ensure that these nodes are among those we look for 1-splits through, so that they end up on T_1^*.

4.3. Two $\mathcal{M}$ requirements and two $\mathcal{L}$ requirements. Now consider four requirements: $\mathcal{M}_0$ of highest priority, $\mathcal{M}_1$ of middle priority, and $\mathcal{L}_0$ and $\mathcal{L}_1$ of lowest priority. As above, suppose that $\mathcal{M}_0$ has the infinitary outcome, successfully building a 0-splitting tree T_0, and that all of the other requirements correctly guess this. Suppose that $\mathcal{L}_0$ guesses that $\mathcal{M}_1$ has the infinitary outcome, successfully building a 1-splitting subtree T_1 of T_0, and suppose that $\mathcal{L}_1$ guesses that $\mathcal{M}_1$ has the finitary outcome, failing to build a 1-splitting tree. ($\mathcal{L}_0$ and $\mathcal{L}_1$ might be of any priority ordering relative to each other, or might even be instances of the same lowness requirement which have different guesses about $\mathcal{M}_1$.) This is the combination of the situations described in Sections 4.2.1 and 4.2.2. The observations there create a dependence between $\mathcal{L}_0$ and $\mathcal{L}_1$:

(1) When $\mathcal{M}_1$ looks for a 1-split extending an observation node for $\mathcal{L}_0$, it should look through extensions which are also observation nodes (Section 4.2.1).
(2) The diagonalization nodes for $\mathcal{L}_1$ must be among those through which $\mathcal{M}_1$ looks for 1-splits (Section 4.2.2).

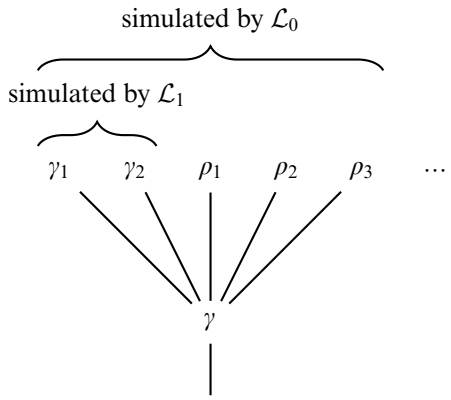
Taken together, this means that $\mathcal{L}_0$ must simulate computations at the diagonalization nodes for $\mathcal{L}_1$, i.e., the diagonalization nodes for $\mathcal{L}_1$ should be observation nodes for $\mathcal{L}_0$. This might seem somewhat odd, because $\mathcal{L}_0$ and $\mathcal{L}_1$ are incompatible in the sense that they have different guesses at the outcome of the higher priority requirement $\mathcal{M}_1$, and because $\mathcal{L}_0$ and $\mathcal{L}_1$ might be of any priority ordering relative to each other. Nevertheless, they are entangled; this is a significant source of combinatorial complexity in the full construction.

Now let us return to the construction of T_0 by $\mathcal{M}_0$ in the presence of these two requirements $\mathcal{L}_0$ and $\mathcal{L}_1$. The construction goes in exactly the same way as in Section 4.1, though some of the reasoning is slightly different. In Section 4.1 the requirement $\mathcal{L}_0$ was of higher priority than $\mathcal{L}_1$, and $\mathcal{L}_1$ guessed that $\mathcal{L}_0$ has the infinitary outcome.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

Now $\mathcal{L}_0$ might be of lower priority than $\mathcal{L}_1$ (or $\mathcal{L}_0$ and $\mathcal{L}_1$ might even be different instances of the same requirement); but there is some minimality requirement $\mathcal{M}_1$ such that $\mathcal{L}_0$ guesses that $\mathcal{M}_1$ has the infinitary outcome and $\mathcal{L}_1$ guesses that it has the finitary outcome. As $\mathcal{M}_0$ has higher priority than $\mathcal{M}_1$, $\mathcal{M}_0$ does not know the outcome of $\mathcal{M}_1$, and hence does not know which of $\mathcal{L}_0$ or $\mathcal{L}_1$ has guessed correctly. Thus it has to take both of them into account.

As in Section 4.1 suppose that $\mathcal{M}_0$ is extending a node γ which is simulated by both $\mathcal{L}_0$ and $\mathcal{L}_1$, and suppose that $\mathcal{M}_0$ must keep nodes $\rho_1, \rho_2, \ldots$ on T_0 because we have set $\Xi_1(x_i) = \Psi_1^{\rho_i}(x_i)$. As in Section 4.1 we will make the extension as follows:



This is exactly the same as in Section 4.1, but now the argument that $\mathcal{L}_0$ must continue to simulate computations above $\rho_1, \rho_2, \ldots$ is different from before. It is now because the ρ_i are diagonalization nodes for $\mathcal{L}_1$ and, as argued above, as a consequence of (1) and (2) these must be observation nodes for $\mathcal{L}_0$.

4.4. The relation of watching. We have already seen two situations in which a requirement $\mathcal{L}_1$ must ensure that the diagonalization nodes for another requirement $\mathcal{L}_0$ are all observation nodes for $\mathcal{L}_1$, thereby ensuring that $\mathcal{L}_1$ simulates computations above these diagonalization nodes for $\mathcal{L}_0$.

We will introduce a relation $\mathcal{L}_{d_1}^{v_1} \rightsquigarrow \mathcal{L}_{d_2}^{v_2}$ to represent this. We suggest reading $\mathcal{L}_{d_1}^{v_1} \rightsquigarrow \mathcal{L}_{d_2}^{v_2}$ as "$\mathcal{L}_{d_1}^{v_1}$ *watches* $\mathcal{L}_{d_2}^{v_2}$." The two situations we have seen are:

(1) If $d_1 < d_2$ and $v_1 \prec v_2$ then we should have $\mathcal{L}_{d_1}^{v_1} \rightsquigarrow \mathcal{L}_{d_2}^{v_2}$ (Section 4.1).

(2) If $\mathcal{L}_{d_1}^{v_1}$ and $\mathcal{L}_{d_2}^{v_2}$ first disagree on the outcome of a minimality requirement $\mathcal{M}$, $\mathcal{L}_{d_1}^{v_1}$ guesses that $\mathcal{M}$ will have the infinitary outcome, and $\mathcal{L}_{d_2}^{v_2}$ guesses that it will have the finitary outcome, then we should have $\mathcal{L}_{d_1}^{v_1} \rightsquigarrow \mathcal{L}_{d_2}^{v_2}$ (Section 4.3).

The reader will immediately notice that this resembles a lexicographic ordering. We will give a formal definition of $\rightsquigarrow$ which captures (1) and (2).

DEFINITION 4.1. Given a string of outcomes v, define $\Delta(v) \in \{\mathrm{f}, \infty\}^{<\omega}$ to be the string

$$\langle v(\mathcal{M}_0), v(\mathcal{M}_1), v(\mathcal{M}_2), \ldots \rangle,$$

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

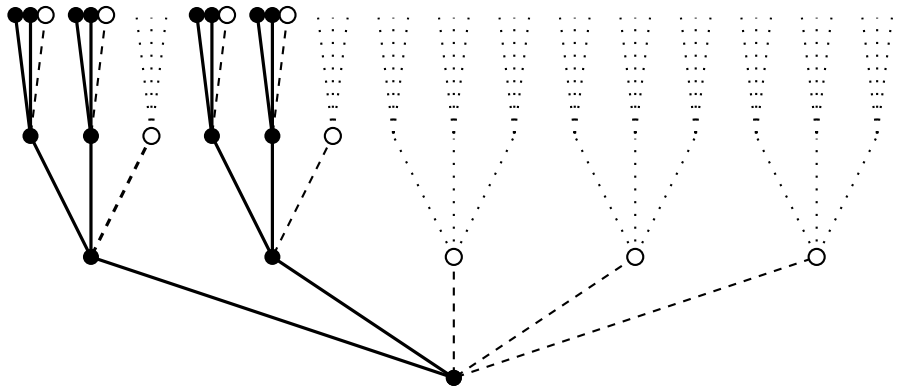
except that we replace any entry which is not ∞ with f. Put an ordering $\precsim$ on these using the lexicographic order with $\infty < \mathrm{f}$.

DEFINITION 4.2. Suppose that v_1 and v_2 are guesses by $\mathcal{L}_{d_1}$ and $\mathcal{L}_{d_2}$ respectively at the outcomes of the higher priority requirements. Define $\mathcal{L}_{d_1}^{v_1} \rightsquigarrow \mathcal{L}_{d_2}^{v_2}$ if and only if $\Delta(v_1) \prec \Delta(v_2)$ in the ordering just defined.

$\mathcal{L}_{d_1}^{v_1} \rightsquigarrow \mathcal{L}_{d_2}^{v_2}$ means that, given a node σ which is an observation node for both requirements, any child of σ which is an observation or diagonalization node for $\mathcal{L}_{d_2}^{v_2}$ should be an observation node for $\mathcal{L}_{d_1}^{v_1}$.

4.5. Two $\mathcal{M}$ requirements and one $\mathcal{L}$ requirement, sandwiched. In Section 4.2 we considered the case of four requirements: $\mathcal{M}_0$ of highest priority, $\mathcal{M}_1$ of middle priority, and $\mathcal{L}$ of lowest priority. Consider now three requirements: $\mathcal{M}_e$ of highest priority, $\mathcal{L}_e$ of middle priority, and $\mathcal{M}_{e+1}$ of lowest priority. Suppose that $\mathcal{M}_e$ successfully builds an e-splitting tree T_e, and that $\mathcal{L}$ correctly guess this.

Consider the diagram from Section 4.2 of the tree T_e, with the observation nodes for $\mathcal{L}$ given in bold. We show the observation nodes with filled in circles, and the diagonalization nodes with open circles.



Suppose moreover that $\mathcal{L}_e$ has the infinitary outcome. This means that $\mathcal{L}_e$ was not able to obtain a diagonalization by having A extend one of the diagonalization nodes. So as previously described, $\mathcal{L}_e$ requires that A be a path through the observation nodes for $\mathcal{L}_e$. The way that we will enforce this is to ensure that when $\mathcal{M}_{e+1}$ builds an $e+1$-splitting tree T_1, or a tree T_{e+1}^* with no $e+1$-splits, it builds it as a subtree of the observation nodes for $\mathcal{L}_e$. (Recall also from Section 4.4 that $\mathcal{L}_e$ is minimal under the watching relation among all instances of lowness requirements of lower priority than $\mathcal{M}_e$; this means that all of the diagonalization nodes for these requirements are observation nodes for $\mathcal{L}_e$, and so these diagonalization nodes can all be preserved on the tree built by $\mathcal{M}_{e+1}$.)

§5. The labeling system. In the full construction there will be many things to keep track of, such as which nodes are observation nodes or diagonalization nodes for which $\mathcal{L}$ requirements, how many $\mathcal{L}$ requirements we are considering, and so on. In

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

this section we will describe a labeling system that will allow us to keep track of all of this. *These labels have nothing to do with the labels in a full approximation argument*.

One simplification we can make is to note that the relation $\mathcal{L}_{d_1}^{\nu_1} \rightsquigarrow \mathcal{L}_{d_2}^{\nu_2}$ depends entirely on $\Delta(\eta_1)$ and $\Delta(\eta_2)$, that is, only on the guesses at the $\mathcal{M}$ outcomes, and only at whether the outcome is finitary or infinitary. Requirements $\mathcal{L}_{d_1}^{\nu_1}$ and $\mathcal{L}_{d_2}^{\nu_2}$ with $\Delta(\eta_1) = \Delta(\eta_2)$ can then be treated the same in terms of the labeling. Of course, there are some differences; for example, if $\mathcal{L}_{d_1}^{\nu_1}$ knows from its guesses at the higher priority requirements that A must extend a node σ, then it will not simulate any computation incompatible with σ. So the labels on their own do not completely determine which nodes are simulated by $\mathcal{L}_{d_1}^{\nu_1}$, or which nodes are observation nodes or diagonalization nodes, but they are sufficient together with the initial segment of A guessed by ν_1. Think of this initial segment of A as being the first observation node of $\mathcal{L}_{d_1}^{\nu_1}$, so that there is a subtree of observation nodes with it as a root.

Consider the e-splitting tree T constructed by $\mathcal{M}_e$. Let ξ be the outcomes of lower priority requirements. 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 $\xi^\frown\infty$. (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.

Definition 5.1. Given an instance $\mathcal{L}_{e+n}^\eta$ of a lowness requirement of lower priority than $\mathcal{M}_e$, we write $\Delta_{>e}(\eta)$ for

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

Recall that $\Delta(\eta)$ just replaces the entries of η by $\mathtt{f}$ or ∞; $\Delta_{>e}(\eta)$ is the string which consists of the guesses by η at the outcomes (finitary $\mathtt{f}$ or infinitary ∞) of $\mathcal{M}_{e+1}, \dots, \mathcal{M}_d$.

The tree T will be a *labeled tree*. These labels have nothing to do with the labels used in a full approximation construction, which represent a guess at whether we can find a splits above a node. Instead, these signify whether a node is an observation node, a diagonalization node, or neither for the various $\mathcal{L}$ requirements. Figure 1 below may help the reader to understand the structure of the tree. In Section 6 we will also revisit some of the examples from Sections 2 and 4 and show how the labels work there.

Recall from Section 4.1 that at any point in the construction we will consider only finitely many $\mathcal{L}$ requirements. Each node σ is given a *scope* $\mathrm{scope}(\sigma)$, which is a natural number that 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 $\mathcal{L}_e, \dots, \mathcal{L}_{e+n}$. Each node σ will also be given *label* $\ell(\sigma)$. The label is an element of

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

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

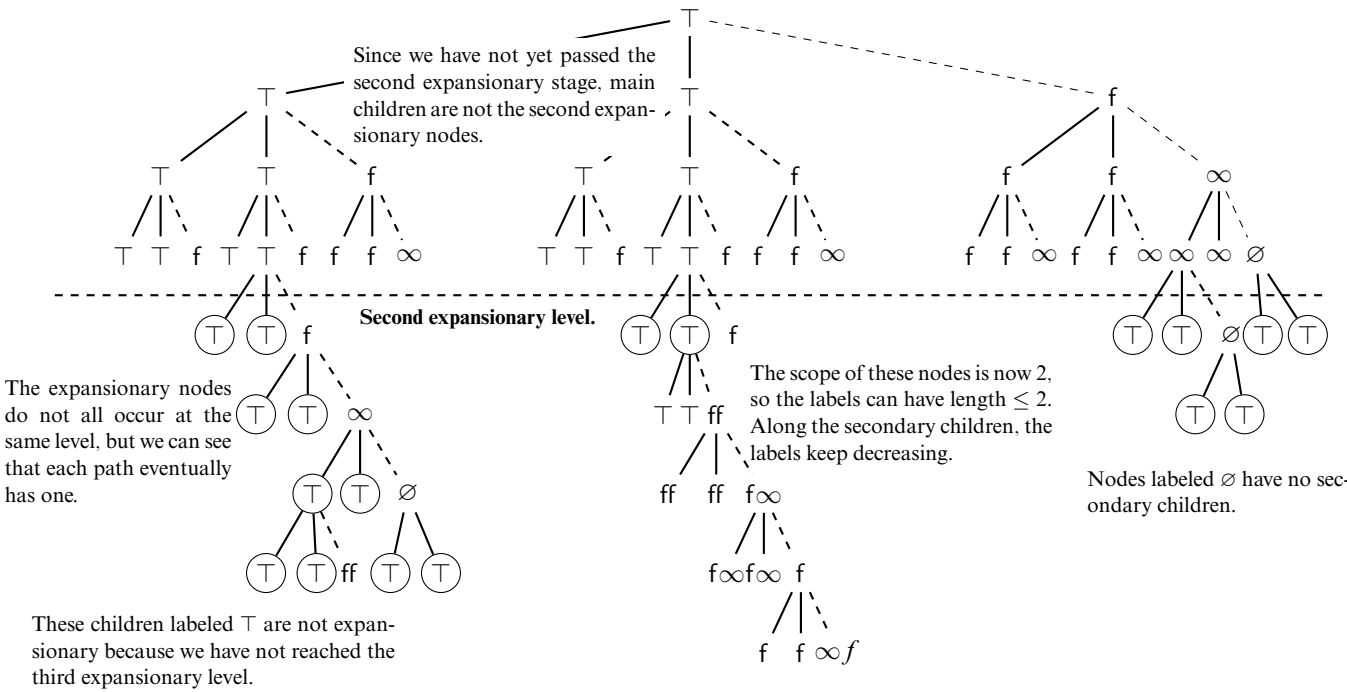


FIGURE 1. An example of what the labeled tree might look like. We draw the main children with a solid line and the secondary children with a dashed line. Expansionary nodes are shown by a circle. To fit the tree onto a single page, we have made some simplifications: (a) we have omitted some nodes from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

If a node σ has scope n, then the label of σ will be an element of

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

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 $\mathcal{L}_{e+m}$ at the outcomes of $\mathcal{M}_{e+1}, \dots, \mathcal{M}_{e+m}$. We order the labels lexicographically with $\infty < \mathsf{f}$, and with $\top$ as the greatest element. For example, in $\texttt{Labels}_2$, we have

$$\top \succ \mathsf{ff} \succ \mathsf{f}\infty \succ \mathsf{f} \succ \infty\mathsf{f} \succ \infty\infty \succ \infty \succ \varnothing.$$

We use $\preceq$ for this ordering, which is of course the same ordering that defined the watching relation $\rightsquigarrow$. We often think of this ordering as being an ordering $\preceq_n$ on $\texttt{Labels}_n$, and write $\text{pred}_n(\eta)$ 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. The label $\mathsf{f}\infty$, for example, represents the instances of $\mathcal{L}_{e+2}$ which guesses that $\mathcal{M}_{e+1}$ has the finitary outcome and $\mathcal{M}_{e+2}$ has the infinitary outcome; the label $\varnothing$ represents $\mathcal{L}_e$, which has no $\mathcal{M}$ requirements of lower priority than $\mathcal{M}_e$ to guess at. In $\texttt{Labels}_2$, we have $\text{pred}_2(\mathsf{f}) = \infty\mathsf{f}$, whereas in $\texttt{Labels}_1$ we have $\text{pred}_1(\mathsf{f}) = \infty$.

We think of elements of $\{\mathsf{f}, \infty\}^n$ as guesses by $\mathcal{L}_{e+n}$ at the outcomes of $\mathcal{M}_{e+1}, \dots, \mathcal{M}_{e+n}$. Think of a label ℓ of length n as being associated with all of the lowness requirements $\mathcal{L}^\eta_{e+n}$ with η extending $\xi^\frown\infty$ and with $\Delta_{>e}(\eta) = \ell$. 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 labels $\ell(\sigma^*)$ are the formal replacement for the more informal notions of diagonalization nodes and observation nodes. Suppose that σ^* is a child of σ on T. Giving σ^* the label $\ell(\sigma^*)$ means that σ^* is a diagonalization node on T for requirements $\mathcal{L}^\eta_{e+|\ell(\sigma^*)|}$ with $\Delta_{>e}(\eta) = \ell(\sigma^*)$ and for which σ was an observation node. Combining this with the relation $\rightsquigarrow$, we get that if $\mathcal{L}^\eta_d$ is an instance of a lowness requirement, and σ^* is the child of σ on T, with $d \leq e + \text{scope}(\sigma^*)$, then:

- if $\ell(\sigma^*) \succ \Delta_{>e}(\eta)$, then if σ is an observation node for $\mathcal{L}^\eta_d$ then σ^* is an observation node for $\mathcal{L}^\eta_d$ and
- if $\ell(\sigma^*) = \Delta_{>e}(\eta)$, then if σ is an observation node for $\mathcal{L}^\eta_d$ then σ^* is a diagonalization node for $\mathcal{L}^\eta_d$.

If $\ell(\sigma^*) = \top$ then σ^* is an observation node for every requirement $\mathcal{L}^\eta_d$ for which the parent node σ was an observation node. These are nodes such as γ_1 and γ_2 from the examples of Sections 2.4 and 4.1. If $\ell(\sigma^*) = \varnothing$ then σ^* is not an observation node for any requirement (but, if σ was an observation node for $\mathcal{L}_e$, then σ^* is a diagonalization node for $\mathcal{L}_e$). These are nodes such as the ρ from Section 2.4 and the τ from Section 4.1; the ρ in Section 4.1 are observation nodes for $\mathcal{L}_0$ and hence would not be labeled $\varnothing$.

The following notation will be helpful for talking about observation nodes among other things.

DEFINITION 5.2. For any labeled 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 $S\{\updownarrow^R_\tau\}$ as the tree with root node τ, and such that whenever $\sigma \in S\{\updownarrow^R_\tau\}$, the children

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

σ^* of σ on $S\{\Upsilon_\tau^R\}$ 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 trees $S\{\Upsilon_\tau^{\succ\eta}\}$ and $S\{\Upsilon_\tau^{\succeq\eta}\}$ for $\eta \in \texttt{Labels}$, and for $S\{\Upsilon_\tau^{main}\}$ (to be defined). So:

- if σ is an observation node for $\mathcal{L}_d^\eta$ then any element of $T\{\Upsilon_\sigma^{\succ\Delta_{>e}(\eta)}\}$ is also an observation node for $\mathcal{L}_d^\eta$ and
- if σ is an observation node for $\mathcal{L}_d^\eta$, $\sigma' \in T\{\Upsilon_\sigma^{\succ\Delta_{>e}(\eta)}\}$, and σ^* is a child of σ' with $\ell(\sigma^*) = \Delta_{>e}(\eta)$, then σ^* is a diagonalization node for $\mathcal{L}_d^\eta$.

We need a system by which we consider more and more requirements $\mathcal{L}$ as we move up the tree. We must be careful about how we introduce new requirements, because introducing a new requirement could be disruptive to our strategy. Certain levels of the tree T_e will be called *expansionary levels*. From the nth expansionary level of the tree on, we will consider requirements $\mathcal{L}_{e+1}, \ldots, \mathcal{L}_{e+n}$, using guesses from at the outcomes of $\mathcal{M}_{e+1}, \ldots, \mathcal{M}_{e+n}$. 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. Conveniently we can precompute the expansionary levels; they 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 $\text{scope}(\sigma)$ 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 $\text{scope}(\sigma) = n - 1$ or $\text{scope}(\sigma) = n$. 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 $\text{scope}(\sigma^*) > \text{scope}(\sigma)$. We say that an expansionary node σ^* is an *nth expansionary node* if $\text{scope}(\sigma^*) = n$. The scopes will be non-decreasing along paths on the tree. Along any path in the tree, there is one expansionary node for each n; it is clear from the fact that the scopes are non-decreasing that there is at most one, and we will show in Lemma 7.3 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 $\text{scope}(\sigma) = n$. If σ^* is an $(n+1)$st expansionary node, then we will have $\text{scope}(\sigma^*) = n + 1$ and $\ell(\sigma^*) = \top$. Otherwise, $\text{scope}(\sigma^*) = \text{scope}(\sigma) = n$ and we will have either $\ell(\sigma^*) = \ell(\sigma)$ or $\ell(\sigma^*) \prec \ell(\sigma)$.

DEFINITION 5.3. If $\ell(\sigma^*) = \ell(\sigma)$ (or if σ^* is an expansionary node, and $\ell(\sigma^*) = \top$) then we say that σ^* is a *main child* of σ. Otherwise, if $\ell(\sigma^*) \prec \ell(\sigma)$, then we say that σ^* is a *secondary child* of σ.

Each node will have two main children and any finite number (including zero) of secondary children. In Section 6 we will return to the examples of Sections 2 and 4 and explain which children are main children and which are secondary children. One should think of the main children as those where the construction "went smoothly": they are the nodes where we found e-splittings. They are not diagonalization nodes for any lowness requirement, and are observation nodes for any lowness requirement for which the parent was an observation node. In contrast, the secondary children

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

are those we were forced to keep on the tree by lowness requirements; they are all diagonalization nodes for some requirement.

Using the notation of Definition 5.2, we will often consider $S\{\restriction_\tau^{main}\}$ where $main(\sigma^*, \sigma)$ is the relation of being a main child. So for example $T\{\restriction_\sigma^{main}\}$ is the tree consisting of main children of main children of... main children of σ.

Consider the construction by $\mathcal{M}_e$ of an e-splitting subtree of T_{e-1}. Recall from Section 4.3 that when we look for e-splits, we do so through only a subtree of T_{e-1}. Namely, if we are looking for an e-split above $\tau \in T_{e-1}$, and if τ is an observation node for a requirement $\mathcal{L}_d^\eta$ which guesses that $\mathcal{M}_e$ has the infinitary outcome, then we should look for that e-split among extensions of τ which are also observation nodes for $\mathcal{L}_{e+d}^\eta$. Let us translate this into the language of labels.

T_{e-1} itself will be a labeled tree, with scopes $\text{scope}_{e-1}\colon T_{e-1} \to \omega$ and labels $\ell_{e-1}\colon T_{e-1} \to \texttt{Labels}$. The nodes on T_{e-1} which are observation nodes for *all* requirements $\mathcal{L}_d^\eta$ which guess that $\mathcal{M}_e$ has the infinitary outcome are the nodes with label $\ell_{e-1}(\sigma) = \mathsf{f}\cdots$ or $\ell_{e-1}(\sigma) = \top$; equivalently, we may write $\ell_{e-1}(\sigma) \succeq \mathsf{f}$. Let $T_{e-1}\{\restriction_\tau^{\succeq \mathsf{f}}\}$ be the subtree of T_{e-1} above τ (so that τ is the root node of $T_{e-1}\{\restriction_\tau^{\succeq \mathsf{f}}\}$) such that given σ on $T_{e-1}\{\restriction_\tau^{\succeq \mathsf{f}}\}$, the children of σ in $T_{e-1}\{\restriction_\tau^{\succeq \mathsf{f}}\}$ are the children σ^* of σ on T_{e-1} with $\ell_{e-1}(\sigma^*) \succ \mathsf{f}$. When we look for a splitting extension of τ, we look through $T_{e-1}\{\restriction_\tau^{\succeq \mathsf{f}}\}$.

The input tree T_{e-1} will have similar properties to those described above.

Definition 5.4. We say that a tree T with labels ℓ and scope is *admissible* if:

(1) Each $\sigma \in T$ has two *main children* σ^* and σ^{**} with $\ell(\sigma^*) \succeq \ell(\sigma)$ and $\ell(\sigma^{**}) \succeq \ell(\sigma)$.
(2) If σ^* is a child of σ, then $\text{scope}(\sigma^*) \geq \text{scope}(\sigma)$.
(3) For each n, each path through T contains a node σ with $\ell(\sigma) = \top$ and $\text{scope}(\sigma) \geq n$.

We must also begin with an admissible tree. As in Section 2.3 let T_{-1} be the tree consisting of all nodes in $2^{<\omega}$ of the form

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

where each $a_i \in \{0, 1\}$. We put $\ell_{-1}(\sigma) = \top$ and $\text{scope}_{-1}(\sigma) = |\sigma|$ for each $\sigma \in T_{-1}$. This T_{-1} is admissible.

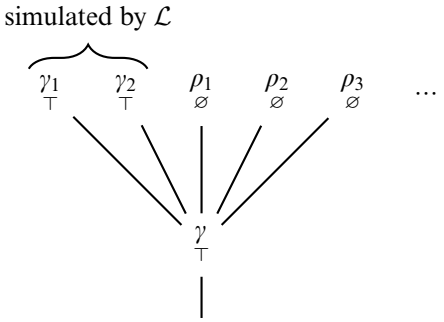
§6. Three or four requirements revisited. Now that we have described the labeling system formally, let us return to the situations described in Sections 2 and 4 to see what happens with the labels.

6.1. One $\mathcal{M}$ requirement and one $\mathcal{L}$ requirement, redux. We begin by revisiting Section 2.4. We have two requirements $\mathcal{M} = \mathcal{M}_e$ and $\mathcal{L} = \mathcal{L}_e$, so that $\mathcal{M}$ is of higher priority than $\mathcal{L}$. Let T_{e-1} be the tree built by $\mathcal{M}_{e-1}$, and assume that we have enough e-splits on T_{e-1}. $\mathcal{M}$ is building an e-splitting tree T and $\mathcal{L}$ is guessing that it is successful.

Suppose that we have determined that γ is on T and we are trying to extend it. Suppose also that $\text{scope}_e(\gamma) = 0$ so that we are only concerned with the single requirement $\mathcal{L}$. We are then assigning labels from $\texttt{Labels}_1 = \{\top \succ \varnothing\}$. Suppose

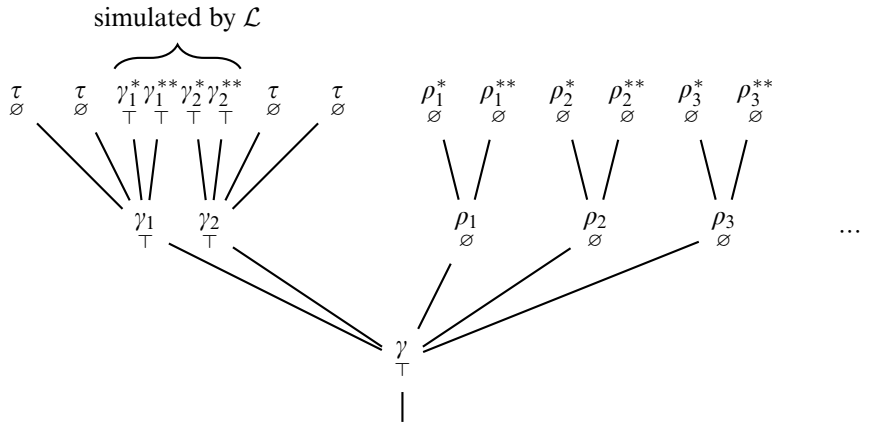
https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

that γ has the label $\top$, and is an observation node for $\mathcal{L}$. We look for a pair of nodes γ_1, γ_2 on $T_{e-1}\{\Xi_\gamma^{\succeq \mathrm{f}}\}$ that e-split. Since we assumed that these exist, at some point we will find them at stage s. Now let $\rho_1, \dots, \rho_n$ be the other nodes on $T_{e-1}[s]$. $\mathcal{L}$ stops simulating computations above $\rho_1, \dots, \rho_n$.



The nodes γ_1 and γ_2 are the main children of γ and are also labeled $\top$, while the ρ_i are secondary children and are labeled $\varnothing$.

Now at the next step we look for γ_1^* and γ_1^{**} on $T_{e-1}\{\Xi_{\gamma_2}^{\succeq \mathrm{f}}\}$, γ_2^* and γ_2^{**} on $T_{e-1}\{\Xi_{\gamma_2}^{\succeq \mathrm{f}}\}$, and ρ_i^* and ρ_i^{**} on $T_{e-1}\{\Xi_{\rho_i}^{\succeq \mathrm{f}}\}$ such that all of these pairwise e-split. The nodes γ_1 and γ_2 also get children τ.



Here γ_1^*, γ_1^{**}, γ_2^*, and γ_2^{**} are labeled $\top$, and the τ, ρ_i^*, and ρ_i^{**} are labeled $\varnothing$. The main children of γ_1 are γ_1^* and γ_1^{**}; the main children of γ_2 are γ_2^* and γ_2^{**}; and the main children of ρ_i are ρ_i^* and ρ_i^{**}. The τ are secondary children.

6.2. One $\mathcal{M}$ requirement and two $\mathcal{L}$ requirements, redux. We now revisit Section 4.1. Consider three requirements: $\mathcal{M}_e$ of highest priority, $\mathcal{L}_e$ of middle priority, and $\mathcal{L}_{e+1}$ of lowest priority. Because $\mathcal{L}_{e+1}$ is guessing at the outcome of $\mathcal{M}_{e+1}$, and $\mathcal{M}_e$ does not know this outcome, $\mathcal{M}_e$ has to consider both instances $\mathcal{L}_{e+1}^{\mathrm{f}}$ and $\mathcal{L}_{e+1}^{\infty}$

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

which guess that $\mathcal{M}_{e+1}$ has the finitary or infinitary outcome respectively. (We omit from the superscript of the instance the guesses at the outcomes of requirements $\mathcal{M}_0, \ldots, \mathcal{M}_e$.)

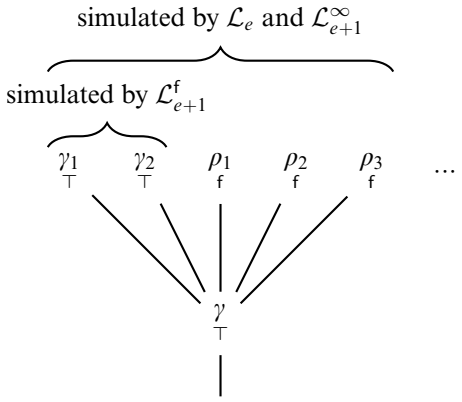
Suppose that the requirement $\mathcal{M}_e$ is trying to extend a node γ and that $\text{scope}(\gamma) = 1$ so that we are only concerned with $\mathcal{L}_e$, $\mathcal{L}^{\mathsf{f}}_{e+1}$, and $\mathcal{L}^{\infty}_{e+1}$. We have

$$\texttt{Labels}_2 = \{\top \succ \mathsf{f} \succ \infty \succ \varnothing\}.$$

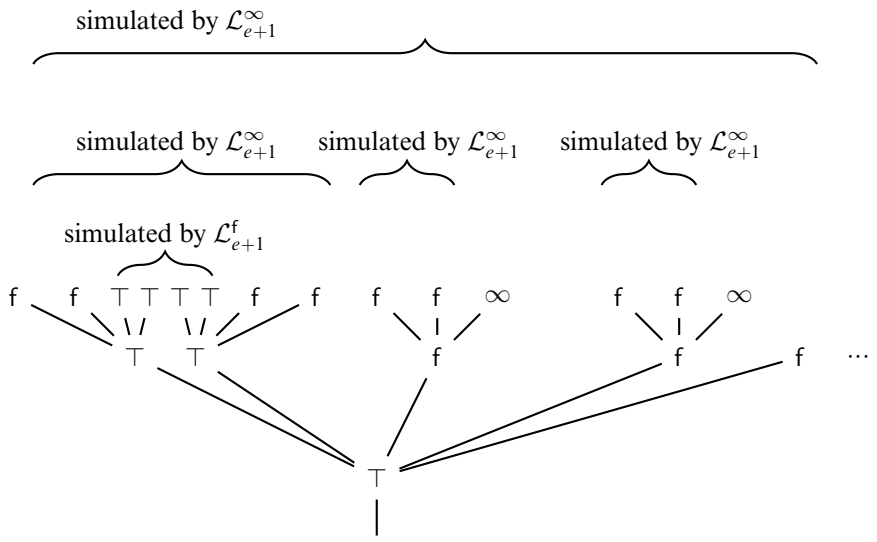
The labels $\varnothing$, f, and ∞ correspond respectively to $\mathcal{L}_e$, $\mathcal{L}^{\mathsf{f}}_{e+1}$ and $\mathcal{L}^{\infty}_{e+1}$. So we have

$$\mathcal{L}_e \rightsquigarrow \mathcal{L}^{\infty}_{e+1} \rightsquigarrow \mathcal{L}^{\mathsf{f}}_{e+1}.$$

At the first level, we have:



At the next level, we have:



https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

It is only at the following stage that we get some nodes which are labeled $\varnothing$ and which are not observation nodes for $\mathcal{L}_{e+1}^{\infty}$.

6.3. Two $\mathcal{M}$ requirements and one $\mathcal{L}$ requirement, sandwiched, redux. Finally, we revisit Section 4.5. We consider three requirements: $\mathcal{M}_e$ of highest priority, $\mathcal{L}_e$ of middle priority, and $\mathcal{M}_{e+1}$ of lowest priority. Suppose that $\mathcal{M}_e$ successfully builds an e-splitting tree T_e, and that $\mathcal{L}_e$ correctly guesses this. If $\mathcal{L}_e$ has the infinitary outcome, then we need A to live within the observation nodes for $\mathcal{L}_e$. Recall that the way that we ensure that this happens is that $\mathcal{M}_{e+1}$ will work only within the observation nodes for $\mathcal{L}_e$ when it builds its 1-splitting tree.

Let η be the guesses by $\mathcal{L}_e$ at the higher priority requirements. Since $\mathcal{L}_e$ is the highest priority requirement after $\mathcal{M}_e$, we have $\Delta_{>e}(\eta) = \varnothing$. Thus the nodes labeled $\varnothing$ are those that might be diagonalization nodes for $\mathcal{L}_e$, and those with labels $\succ \varnothing$ are observation nodes for $\mathcal{L}_e$. Thus if ρ is the node at which we want to start building T_{e+1}, then we want to build T_{e+1} as a subtree of $T_e\{\uparrow_{\rho}^{\succ \varnothing}\}$.

6.4. A new issue. There is one more issue which we have glossed over previously (and so some of the previous examples are somewhat misleading without keeping this issue in mind). Consider the requirements $\mathcal{M}_{e-1}$, with corresponding tree T_{e-1}, and $\mathcal{M}_e$, which is building an e-splitting subtree of T_{e-1}. Suppose that $\mathcal{M}_e$ has the infinitary outcome. Let $\mathcal{L}^v$ be an instance of a requirement of lower priority than $\mathcal{M}_e$. The issue is that the labels on T_{e-1} might affect the labels on T_e: essentially, if we determine that a node on T_{e-1} is *not* an observation node for $\mathcal{L}^v$, then this means that $\mathcal{L}^v$ stops simulating computations above that node, and so no node extending that node should be an observation node on T_e.

Let $\eta = \Delta_{>e}(v)$. Assume that $\mathcal{L}$ guesses correctly that $\mathcal{M}_e$ has the infinitary outcome, so that $\Delta_{>e-1}(v) = \infty\eta$.

Suppose that we have determined on T_e that σ^* should be a child of σ, with σ an observation node for $\mathcal{L}$ on both T_{e-1} and T_e. On T_{e-1}, we might have a sequence of children $\sigma = \sigma_0, \sigma_1, \ldots, \sigma_n = \sigma^*$. If *any* of these is not an observation node for $\mathcal{L}^v$ on T_{e-1}—that is, if $\ell_{e-1}(\sigma_i) \not\succ \infty\eta$—then σ^* should not be an observation node for $\mathcal{L}^v$ on T_e—that is, we should have $\ell_e(\sigma^*) \not\succ \eta$.

§7. Construction.

7.1. Procedure for constructing splitting trees. We are now ready to describe the procedure for constructing splitting trees. Given the tree T_{e-1} constructed by $\mathcal{M}_{e-1}$, 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}. The construction will happen above some root node ρ, which is determined by T_{e-1} together with the outcomes of the requirements $\mathcal{L}_{e-1}$ and $\mathcal{P}_{e-1}$. For example, if $\mathcal{L}_{e-1}$ has the finitary outcome, then ρ will extend this.

The construction will be given by a procedure $\mathsf{Procedure}(e, \rho, T_{e-1})$ for building an e-splitting tree T with root ρ in T_{e-1}. We write $T[n]$ for the tree up to and including the nth level.

$\mathsf{Procedure}(e, \rho, T_{e-1})$:

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

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 zeroth 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 $\sigma' \in T_{e-1}\{\updownarrow_{\sigma}^{main}\}$ with $\ell_{e-1}(\sigma') = \top$, and there are extensions σ^* and σ^{**} of σ on $T_{e-1}\{\updownarrow_{\sigma'}^{\succ f}\}$, 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^{**}*.[4] 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, including σ itself, is a tth expansionary node, set $\text{scope}(\sigma^*) = \text{scope}(\sigma) + 1$ and $\ell(\sigma^*) = \top$ (so that σ^* is a tth expansionary node).
  - Otherwise, set $\text{scope}(\sigma^*) = \text{scope}(\sigma)$ and $\ell(\sigma^*) = \ell(\sigma)$.
- σ^{**}, with:
  - If no predecessor of σ on T, including σ itself, is a tth expansionary node, set $\text{scope}(\sigma^{**}) = \text{scope}(\sigma) + 1$ and $\ell(\sigma^{**}) = \top$ (so that σ^{**} is a tth expansionary node).
  - Otherwise, set $\text{scope}(\sigma^{**}) = \text{scope}(\sigma)$ and $\ell(\sigma^{**}) = \ell(\sigma)$.
- If $\ell(\sigma) \succ \varnothing$, each other maximal extension $\sigma^\dagger$ of σ on $T_{e-1}\{\updownarrow_{\sigma}^{\succ \varnothing}\}[s]$ which is incompatible with σ^* and σ^{**} will be a child of σ on T.[5] Put $\text{scope}(\sigma^\dagger) = \text{scope}(\sigma)$. Define $\ell(\sigma^\dagger)$ as follows. Let $n = \text{scope}(\sigma)$. Let $\eta \in \texttt{Labels}_n$ be greatest such that $\sigma^\dagger \in T_{e-1}\{\updownarrow_{\sigma}^{\succeq \eta}\}$. Then:
  - If η is $\top$ or begins with f, then let $\ell(\sigma^\dagger) = \text{pred}_n(\ell(\sigma))$.
  - If η begins with ∞, say $\eta = \infty\eta^*$, then $\ell(\sigma^\dagger)$ will be the minimum, in $\texttt{Labels}_n$, of $\text{pred}_n(\ell(\sigma))$ and η^*.[6]

[4] This designation is purely for the use of the simulations for lowness requirements, and has no effect on the resulting tree T. See $(*)$ of the definition of Ξ in Section 8.3.

[5] We take only nodes from $T_{e-1}\{\updownarrow_{\sigma}^{\succ \varnothing}\}$ in order to meet the infinitary outcome of the lowness requirement $\mathcal{L}_{e-1}$. If $\mathcal{L}_{e-1}$ has the finitary outcome there will be no harm in taking nodes only from $T_{e-1}\{\updownarrow_{\sigma}^{\succ \varnothing}\}$. See Sections 4.5 and 6.3 and Remark 7.2. This fact will be used in Lemma 8.4.

[6] See Section 6.4 for why we define the labels in this way with reference to η. The main place this is used is in Lemma 7.7, which is then used in Lemma 8.4 to verify that the simulations Ξ work.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

Note that $\text{pred}_n(\ell(\sigma))$ exists because $\ell(\sigma) \succ \varnothing$. (Recall that pred_n is the predecessor relation on labels of length at most n.)

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 constructs 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 7.1. *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 $\sigma \in T_{e-1}\{\substack{\succ\varnothing\\ \rho}\}$ with $\ell_{e-1}(\sigma) = \top$,*

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

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

Then Procedure(e, ρ, T_{e-1}) *is successful.*

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

REMARK 7.2. Every node on T is a node on $T_{e-1}\{\substack{\succ\varnothing\\ \rho}\}$.

PROOF OF LEMMA 7.1. 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 $T_{e-1}\{\substack{main\\ \sigma_i}\}$ with $\ell(\sigma_i') = \top$. By the remark, each $\sigma_i' \in T_{e-1}\{\substack{\succ\varnothing\\ \rho}\}$. Then using the assumption of the lemma and standard arguments there are $\sigma_i^*, \sigma_i^{**}$ on $T_{e-1}\{\substack{\succeq \mathsf{f}\\ \sigma_i'}\}$ such that all of the σ_i^* and σ_i^{**} pairwise e-split. For sufficiently large stages s, we will find these extensions. ⊣

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 7.3. *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 $\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_k = \sigma^*$ 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

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

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.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

LEMMA 7.4. *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 7.5. *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 $\sigma_0 = \sigma, \sigma_1, \sigma_2, \dots, \sigma_k = \sigma^*$ be the sequence of predecessors of σ^* between σ and σ^*, and similarly for $\tau_0 = \tau, \tau_1, \tau_2, \dots, \tau_k = \tau^*$. 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 7.4, 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 7.6. *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 an observation node for some lowness requirement $\mathcal{L}$, then it should also be labeled on T to not be an observation node for $\mathcal{L}$. (The converse is not necessary; T might determine that some node is not an observation node for $\mathcal{L}$ even if that was not determined by T_{e-1}.) See Section 6.4. Translating this into labels, we get the following lemmas:

LEMMA 7.7. *Suppose that* Procedure(e, ρ, T_{e-1}) *successfully constructs* T. *Given* $\sigma \in T$, *with* $\ell_{e-1}(\sigma) = \top$, *and* $\sigma^* \in T\{\substack{\succ\eta \\ \sigma}\}$ *we have* $\sigma^* \in T_{e-1}\{\substack{\succ\infty\eta \\ \sigma}\}$. *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 $\sigma^* \in T_{e-1}\{\substack{\succ\mathsf{f} \\ \sigma'}\}$ and $\sigma' \in T_{e-1}\{\substack{main \\ \sigma}\}$, 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 $\ell_{e-1}(\sigma') = \top$.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

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

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

This proves the lemma. ⊣

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

LEMMA 7.8. *Suppose that* Procedure(e, ρ, T_{e-1}) *successfully constructs* T. *Given* $\sigma \in T$, *with* $\ell_{e-1}(\sigma) = \top$, *if*:

- $\sigma^* \in T\{\substack{\succeq\eta\\ \sigma}\}$, *we have* $\sigma^* \in T_{e-1}\{\substack{\succeq\infty\eta\\ \sigma}\}$. *In particular, if* $\ell_e(\sigma^*) \succeq \eta$, *then* $\ell_{e-1}(\sigma^*) \succeq \infty\eta$.
- $\sigma^* \in T\{\substack{\succeq\top\\ \sigma}\}$, *we have* $\sigma^* \in T_{e-1}\{\substack{\succeq\top\\ \sigma}\}$. *In particular, if* $\ell_e(\sigma^*) = \top$, *then* $\ell_{e-1}(\sigma^*) = \top$.

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

LEMMA 7.9. *Suppose that* Procedure(e, ρ, T_{e-1}) *successfully constructs* T. *Then* T *is an admissible tree.*

7.2. Construction of A and the true path. We simultaneously define A, the trees T_e satisfying $\mathcal{M}_e$, and a true path of outcomes π by finite extension. This construction is non-uniform, and is analogous to choosing a generic in a forcing construction.

Begin with $A_{-1} = \varnothing$ and $\pi_{-1} = \varnothing$. Recall that, as in Section 2.3, we let T_{-1} be the tree consisting of all nodes in $2^{<\omega}$ of the form

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

where each $a_i \in \{0, 1\}$. We put $\ell_{-1}(\sigma) = \top$ and $\mathrm{scope}_{-1}(\sigma) = |\sigma|$ for each $\sigma \in T_{-1}$. This T_{-1} is admissible.

Suppose that we have so far defined $A_s \prec A$, T_s, 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 its outcome is, and then define the extensions appropriately.

$s{+}1 = 3e$: Consider $\mathcal{M}_e^{\pi_s}$. If Procedure(e, A_s, T_{e-1}) successfully constructs a tree T, let T_e be this tree T, and set $\pi_{s+1} = \pi_s{}^\frown\infty$ and $A_{s+1} = A_s$. Otherwise, by Lemma 7.1, there is $\sigma \in T_{e-1}\{\substack{\succ\varnothing\\ A_s}\}$ with $\ell_{e-1}(\sigma) = \top$ and $\mathrm{scope}_{e-1}(\sigma) \geq 1$ such that either:

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

(2) for all $\tau_1, \tau_2 \in T\{\substack{\succeq\mathsf{f}\\ \sigma}\}_{e-1}$ 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 either case, let $\pi_{s+1} = \pi_s{}^\frown\sigma$ and $A_{s+1} = \sigma \succeq A_s$. Let T_e be the tree $T_{e-1}\{\substack{\succeq\mathsf{f}\\ \sigma}\}$. The labels ℓ_e and scopes of the tree T_e are defined by setting $\mathrm{scope}_e(\tau) = \mathrm{scope}_{e-1}(\tau) - 1$ and:

- $\ell_e(\tau) = \top$ if $\ell_{e-1}(\tau) = \top$ or

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

- $\ell_e(\tau) = \eta$ if $\ell_{e-1}(\tau) = \mathsf{f}\eta$.[7]

$s{+}1 = 3e{+}1$: Consider $\mathcal{L}_e^{\pi_s}$ with $e = \langle e_1, e_2\rangle$. If there is $\sigma \in T_e$ 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 7.10 below) 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 $A_s = \sigma \in T_e$, let τ_1 and τ_2 be the two main children of σ on 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 7.10. *For each e, T_e is an admissible labeled tree.*

PROOF. We argue by induction on e. T_{-1} is an admissible tree. Given T_e total and admissible, if $\mathcal{M}_{e+1}$ has the infinitary outcome then T_{e+1} is defined from T_e using the Procedure, which is successful, and hence by Lemma 7.9 is admissible.

So suppose that $\mathcal{M}_{e+1}$ has the finitary outcome, and T_{e+1} is the tree $T_e\{\hat{\ }_{\sigma}^{\succ\mathsf{f}}\}$ for some $\rho \in T_e\{\hat{\ }_{A_{\mathcal{P}_{e-1}}}^{\succ\varnothing}\}$ with $\ell_e(\rho) = \top$. We must argue that T_{e+1} is admissible:

(1) Each $\sigma \in T_{e+1}$ has two main children σ^* and σ^{**}, namely the same two main children of σ in T_e. We have $\ell_{e+1}(\sigma^*) \succeq \ell_{e+1}(\sigma)$. There are two cases to check: if $\ell_e(\sigma^*) = \mathsf{f}\ell_{e+1}(\sigma^*)$, then $\ell_e(\sigma) = \mathsf{f}\ell_{e+1}(\sigma)$ and from $\ell_e(\sigma^*) \succeq \ell_e(\sigma)$ we conclude that $\ell_{e+1}(\sigma^*) \succeq \ell_{e+1}(\sigma)$; and otherwise, $\ell_e(\sigma^*) = \top$ and so $\ell_{e+1}(\sigma^*) \succeq \ell_{e+1}(\sigma)$. Similarly for σ^{**} we have $\ell_{e+1}(\sigma^{**}) \succeq \ell_{e+1}(\sigma)$.

(2) If σ^* is a child of σ on T_{e+1}, then σ^* is a child of σ on 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 T_{e+1} is also a path through T_e, and hence contains a node σ with $\ell_{e+1}(\sigma) = \ell_e(\sigma) = \top$ and $\mathrm{scope}_{e+1}(\sigma) = \mathrm{scope}_e(\sigma) - 1 \geq n$. ⊣

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

8.1. Non-computable. We chose the initial segment $A_{\mathcal{P}_e} = A_{3e+2}$ of A such that it differs from the eth c.e. set. Thus A is not computable. (These requirements also ensure that A is extended to an infinite string.)

8.2. Minimal degree. We show that A is of minimal degree by showing that it lies on the trees T_e which are either e-splitting or force Φ_e^A to be partial or computable.

LEMMA 8.1. *For all e, $A \in [T_e]$.*

[7] The definitions of the labels here are to ensure that nodes which are determined not to be observation nodes on T_{e-1} are also not observation nodes on T_e; this is in some sense the equivalent, for the finitary outcome of $\mathcal{M}_e$, of Lemmas 7.7 and 7.8. See also Section 6.4.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

PROOF. Fix e. Then $A_{3e} = A_{\mathcal{M}_e}$ is the root of T_e. Then we choose $A_{3e} \preceq A_{3e+1} \preceq A_{3e+2}$ in T_e. Then for each $e' \geq e$, $A_{e'} = A_{\mathcal{M}_{e'}} \in T_{e'}$ which is a subtree of T_e. So $A \in [T_e]$. ⊣

LEMMA 8.2. *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 $\mathcal{M}_e$ is ∞, then A lies on the e-splitting tree $T_e = T$ produced by Procedure$(e, A_{\mathcal{P}_{e-1}}, T_{e-1})$ and hence $\Phi_e^A \geq_T A$. If the outcome of $\mathcal{M}_e$ is σ, then A lies on $T_e = T_{e-1}\{\sigma^{\succ \mathsf{f}}\}$ and (since Φ_e^A is total) for all $\tau_1, \tau_2 \in T_{e-1}\{\sigma^{\succ \mathsf{f}}\}$ 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. ⊣

8.3. Low-for-speed. Our final task is to show that A is low-for-speed. Fix a lowness requirement $\mathcal{L}_e$. Define $\eta = \Delta(\pi_{\mathcal{L}}) \in \{\mathsf{f}, \infty\}^{e+1}$; 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 $\mathcal{M}_{i+1}, \ldots, \mathcal{M}_e$.

In checking that $\mathcal{L}_e$ is satisfied, we will refer only to the trees $T_{-1}, \ldots, T_e$. Write scope_i and ℓ_i for the scope and labeling function on T_i.

By stage s of the construction of T_i we mean:

(1) for $i = -1$, the strings of length $\leq s$ in T_{-1};
(2) if the outcome of $\mathcal{M}_i$ was infinitary, stage s of the Procedure constructing T_i; and
(3) if the outcome of $\mathcal{M}_i$ was finitary, that part of T_i which is determined by stage s of the construction of T_{i-1}.

One can check that the sth stage of Procedure takes time polynomial in s. The particular polynomial will depend on the parameters for Procedure. In checking this, it is important to note that 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 stages $s+1, s+2, \ldots$. Also, because all of these trees are subtrees of T_{-1}, there are only polynomially in s many elements of each tree of length (as a binary string) at most s.

Now we will define the simulation procedure. Let $\rho_1, \ldots, \rho_k$ be incomparable nodes on T_e such that (a) every path on 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 T_e is admissible. (Think of the ρ_i as an open cover of T_e by nodes whose scope, in every $T_{e'}$ $(e' \leq e)$, includes $\mathcal{L}_e$.) For each i, we define a simulation $\Xi_{e,i}$ which works for extensions of ρ_i. As A extends some ρ_i, one of these simulations will work for 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 $\sigma \in T_{-1}$ with $|\sigma| < s$ and $\sigma \succeq \rho_i$, check whether, for each $e' \leq e$, if $T_{e'}[n]$ is the greatest level of the tree $T_{e'}$ defined by stage s of the construction of the trees T, then either:

(S1) σ is on $T_{e'}[n]$ and $\sigma \in T_{e'}\{\rho_i^{\succ \eta_{>e'}}\}$, or

(S2) σ extends a leaf σ' of $T_{e'}[n]$, with $\sigma' \in T_{e'}\{\rho_i^{\succ \eta_{>e'}}\}$, and:

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

$(*)$ if $T_{e'}$ is defined using Procedure (i.e., $\mathcal{M}_{e'}$ 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^{**}.[8]

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.

REMARK 8.3. 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 $\sigma \in T_{-1}$ with $|\sigma| < s$.

We can now make a more formal definition of an observation node and a diagonalization node for $\mathcal{L}_e^\eta$ on $T_{e'}$ $(e' \leq e)$ above ρ_i, though this terminology will only be used for explaining the proofs and not in the proofs themselves. A node σ extending ρ_i is an *observation node* for $\mathcal{L}_e^\eta$ above ρ_i on $T_{e'}$ if $\sigma \in T_{e'}\{\updownarrow_{\rho_i}^{\succ\eta_{>e'}}\}$. A node σ extending ρ_i is a *diagonalization node* for $\mathcal{L}_e^\eta$ above ρ_i on $T_{e'}$ if the parent of σ is in $T_{e'}\{\updownarrow_{\rho_i}^{\succ\eta_{>e'}}\}$ and if $\ell(\sigma) = \eta_{>e'}$.

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_1}. Fix j such that A extends ρ_j, and write $\Xi = \Xi_{e,j}$.

If $\mathcal{L}_e$ has outcome ∞, then we need the initial segments of A to be observation nodes for $\mathcal{L}_e$, so that if $\Psi^A(x)\downarrow$ then we set $\Xi(x) = \Psi^A(x)$ if it was not already defined. First we prove that the initial segments of A have the right labels, i.e., that they are observation nodes, and second that these observation nodes actually get simulated by the simulation procedure.

LEMMA 8.4. *If* $\pi(\mathcal{L}_e) = \infty$, *for each* $e' \leq e$, $A \in [T_{e'}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'}}\}]$ *for some* i.

PROOF. Let $\sigma = A_{\mathcal{M}_e}$ be the root of 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^* is a main child of σ. A is a path through T_{e+1} extending σ^*, and T_{e+1} is one of the following two trees, depending on the outcome of $\mathcal{M}_{e+1}$:[9]

(1) the labeled tree T produced by Procedure$(e+1, \sigma^*, T_e)$, which is a subtree of $T_e\{\updownarrow_{\sigma^*}^{\succ\varnothing}\}$ or

(2) the tree $T_e\{\updownarrow_{\tau}^{\succeq \mathsf{f}}\}$ for some $\tau \in T_e\{\updownarrow_{\sigma^*}^{\succ\varnothing}\}$ with $\ell_e(\tau) = \top$.

In either case, $A \in [T_e\{\updownarrow_{\rho_j}^{\succ\varnothing}\}]$, and $\varnothing = \eta_{>e}$. (Note that ρ_j extends σ and is extended by A.)

Now we argue backward by induction. Suppose that $A \in [T_{e'}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'}}\}]$. We want to argue that $A \in [T_{e'-1}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'-1}}\}]$. We have two cases, depending on the outcome of $\mathcal{M}_{e'}$:

[8]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 $\preceq \eta_{>e'}$. We just have not yet determined what these children are because we are waiting for T_{e-1}. 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.

[9]Recall from Sections 4.5 and 6.3 that when $\mathcal{L}_e$ has the infinitary outcome it is the choice of T_{e+1} that forces the initial segments of A to be observation nodes for $\mathcal{L}_e$, which is why we consider T_{e+1} here.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

- The outcome of $\mathcal{M}_{e'}$ is ∞. Then $\eta_{>e'-1} = \infty\eta_{>e'}$ and $T_{e'}$ is the labeled tree T produced by Procedure$(e', A_{\mathcal{P}_{e'-1}}, T_{e'-1})$. By Lemma 7.7, given $\sigma \in T_{e'}$ and $\sigma^* \in T_{e'}\{\updownarrow_{\sigma}^{\succ\eta_{>e'}}\}$, we have that $\sigma^* \in T_{e'-1}\{\updownarrow_{\sigma}^{\succ\infty\eta_{>e'}}\} = T_{e'-1}\{\updownarrow_{\sigma}^{\succ\eta_{>e'-1}}\}$. As $A \in [T_e\{\updownarrow_{\rho_j}^{\succ\eta_{>e'}}\}]$ we have $A \in [T_{e'-1}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'-1}}\}]$.
- The outcome of $\mathcal{M}_{e'}$ is f. Then $\eta_{>e'-1} = \mathsf{f}\eta_{>e'}$ and $T_{e'}$ is the tree $T_{e'-1}\{\updownarrow_{\tau}^{\succ\mathsf{f}}\}$ for some $\tau \in T_{e-1}\{\updownarrow_{A_{\mathcal{P}_{e'-1}}}^{\succ\varnothing}\}$ with $\ell_{e-1}(\tau) = \top$. The labels on $T_{e'}$ are defined so that if $\sigma \in T_{e'}$, then $\ell_{e'-1}(\sigma) = \mathsf{f}\ell_{e'}(\sigma)$ or $\ell_{e'-1}(\sigma) = \top$. Thus, as $A \in [T_{e'}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'}}\}]$, we get that $A \in [T_{e'-1}\{\updownarrow_{\rho_j}^{\succ\mathsf{f}\eta_{>e'}}\}] = [T_{e'-1}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'-1}}\}]$. ⊣

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. The proof is essentially showing that we do in fact simulate computations at observation nodes.

LEMMA 8.5. *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 $\Psi_s^A(r)\downarrow$, then $\Xi_{p(s)}(r)\downarrow$.*

PROOF. By the previous lemma for each $e' \leq e$, $A \in [T_{e'}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'}}\}]$. Let $\sigma \in T_e$ be an initial segment of A and s a stage such that $\Psi_s^\sigma(r)\downarrow$. We may assume that σ is sufficiently long that σ extends ρ_j.

Fix e' and let $T_{e'}[n]$ be the greatest level of the tree $T_{e'}$ defined by stage s. Then, as $A \in [T_{e'}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'}}\}]$, $\sigma \in T_{e'}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'}}\}$. We check the conditions (for e') from the definition of the simulation Ξ. Either $\sigma \in T_{e'}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'}}\}[n]$, or some initial segment σ' of σ is in $T_{e'}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'}}\}[n]$. In the second case, let us check that we satisfy $(*)$. We only need to check $(*)$ in the case that $T_{e'}$ was defined using Procedure,

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

and σ' has at stage s been designated *waiting with main children σ^* and σ^{**}*. Now, if $\sigma \in T_e$ does not extend σ^* or σ^{**} then it would have to extend one of the other children of σ, namely a secondary child $\sigma^\dagger$ of σ' with

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

This contradicts the fact that $\sigma \in T_{e'}\{\updownarrow_{\rho_j}^{\succ\eta_{>e'}}\}$.

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

The simulation defines $\Xi(r)$ at the sth stage of the simulation. By Remark 8.3, the sth stage of the simulation can be computed in time polynomial in s. ⊣

Lemma 8.5 covers the infinitary outcome of $\mathcal{L}_e$. For the finitary outcome, we need to see that any computation simulated by Ξ is witnessed by a computation on the tree, because the use of such a computation is a diagonalization nodes that was not removed from the tree.

LEMMA 8.6. *If $\Xi(r)\downarrow$ then there is $\sigma \in T_e$, $\sigma \succeq \rho_j$, such that $\Psi^\sigma(r) = \Xi(r)$.*

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

PROOF. Since $\Xi(r)\downarrow$, by definition of the simulation, there is a stage s and a $\sigma \in T_{-1}$ with $|\sigma| < s$ such that $\Psi_s^{\sigma}(r)\downarrow$ for which we set $\Xi(r) = \Psi_s^{\sigma}(r)$. At stage s, for each $e' \leq e$ this σ satisfied either (S1) or (S2). Fix this s for the rest of the proof.

We argue by induction on $-1 \leq i \leq e$ that there is σ^* a child on T_i of a node σ with:

(1) $\ell_i(\sigma^*) \succeq \eta_{>i}$;
(2) $\sigma \in T_i\{\substack{\succ \eta_{>i} \\ \rho_j}\}$;
(3) $\Xi(r) = \Psi_s^{\sigma^*}(r)$; and
(4) σ^* satisfies either (S1) or (S2) for each e' with $i < e' \leq e$.

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. $\pi_{\mathcal{M}_{i+1}} = \infty$, *the infinitary outcome.*

Since $\pi_{\mathcal{M}_{i+1}} = \infty$, T_{i+1} is the $(i+1)$-splitting tree defined by Procedure$(i+1, A_{\mathcal{P}_i}, T_i)$. Fix σ^* from the induction hypothesis. Since $\Psi_s^{\sigma^*}(r)$ converges, we have that $|\sigma^*| \leq s$ and so $\sigma^* \in T_i[s]$. Let n be the greatest level of T_{i+1} defined by stage s.

At stage s, σ^* satisfied either (S1) or (S2) for $i+1$. If it satisfies (S1) then $\sigma^* \in T_{i+1}\{\substack{\succ \eta_{>i+1} \\ \rho_j}\}$ and we are done. So assume that σ^* satisfies (S2). Then σ^* extends a leaf σ_0 of $T_{i+1}[n]$, with $\sigma_0 \in T_{i+1}\{\substack{\succ \eta_{>i+1} \\ \rho_j}\}$, and also satisfies $(*)$. Let $\sigma_0, \sigma_1, \ldots, \sigma_n$ be the maximal sequence of children of σ_0 on T_{i+1} which are strict predecessors of σ^*.

CLAIM 1. *For each k, σ_{k+1} is a main child of σ_k on T_{i+1}. Thus for each k, $\ell_{i+1}(\sigma_k) \succ \eta_{>i+1}$.*

PROOF. We argue inductively. Suppose that σ_k is a main child of σ_{k-1}, σ_{k-1} is a main child of σ_{k-2}, and so on. Thus $\ell_{i+1}(\sigma_k) \succ \eta_{>i+1}$. (If $\sigma_0 = \rho_j$ we use the fact that $\ell_{i+1}(\rho_j) = \top$.)

Suppose that σ_{k+1} is put on T_{i+1} at a stage $t \geq s$. We claim that if σ_{k+1} was a secondary child of σ_k, then σ_{k+1} would extend σ^*. It suffices to show that $\sigma^* \in T_i\{\substack{\succ \varnothing \\ \sigma_k}\}[t]$. Indeed, as remarked above, $\sigma^* \in T_i[s]$, $\sigma^* \in T_i\{\substack{\succeq \eta_{>i} \\ \rho_j}\}$, and $\eta_{>i} = \infty\eta_{>i+1} \succ \varnothing$. ⊣

CLAIM 2. *One of the children σ_{n+1} of σ_n extends σ^* and has $\ell_{i+1}(\sigma_{n+1}) \succeq \eta_{>i+1}$.*

PROOF. Suppose that the children of σ_n are put on T_{i+1} at a stage $t \geq s$. To see that one of the children of σ_n extends σ^*, it suffices to show that $\sigma^* \in T_i\{\substack{\succ \varnothing \\ \sigma_n}\}[t]$ as all such nodes are either compatible with a main child of σ_n—in which case σ^* would be extended by this main child by choice of σ_n—or are extended by a secondary child of σ_n. Indeed, as in the previous claim, $\sigma^* \in T_i[s]$, $\sigma^* \in T_i\{\substack{\succeq \eta_{>i} \\ \rho_j}\}$, and $\eta_{>i} = \infty\eta_{>i+1} \succ \varnothing$.

We have two cases. (a) Suppose that σ^* is extended by a main child σ_{n+1} of σ_n. Then $\ell_{i+1}(\sigma_{n+1}) \succeq \ell_{i+1}(\sigma_n) \succ \eta_{>i+1}$ and we are done.

(b) Suppose that σ^* is extended by a main child of σ_n. We have $\sigma^* \in T_i\{\substack{\succeq \eta_{>i} \\ \rho_j}\}$, and so there is an extension σ_{n+1} of σ^* on T_i with σ_{n+1} on the tth level of $T_i\{\substack{\succeq \eta_{>i} \\ \rho_j}\}$,

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

namely, σ_{n+1} is a main child (on T_i) of main children of ...of σ^*. Now σ_{n+1} is taken as a secondary child of σ_n. As $\ell_{i+1}(\sigma_n) \succ \eta_{>i+1}$, $\sigma_{n+1} \in T_i\{\substack{\succeq \eta_{>i} \\ \sigma_n}\}$, and $\eta_{>i} = \infty\eta_{>i+1}$, we have $\ell_{i+1}(\sigma_{n+1}) \succeq \eta_{>i+1}$. ⊣

CLAIM 3. *σ_{n+1} and its parent σ_n satisfy* (1)–(4).

PROOF. We get (1) from Claim 2. We get (2) from Claim 1 together with the fact that $\sigma_0 \in T_{i+1}\{\substack{\succ \eta_{>i+1} \\ \rho_j}\}$. (3) follows immediately from the fact that σ_{n+1} extends σ^*.

Fix e', $i+1 < e' \leq e$. Note that as σ^* did not satisfy (S1) for $i+1$ we had $\sigma^* \notin T_{i+1}[n]$. Thus $\sigma^* \notin T_{e'}[s]$ and so σ^* did not satisfy (S1) for e'. So σ^* satisfied (S2) for e'. As an extension of σ^*, σ_{n+1} also satisfies (S2) for e'. ⊣

This ends Case 1.

CASE 2. *$\pi_{\mathcal{M}_{i+1}} = \tau$, a finitary outcome.*

Let σ^* be as in the induction hypothesis for i. Let σ be the parent of σ^* on T_{i+1}; note that σ is a predecessor of the parent of σ^* on T_i. We have that $\ell_i(\sigma^*) \succeq \eta_{>i}$ and σ is on $T_i\{\substack{\succ \eta_{>i} \\ \rho_j}\}$.

Note that since $\pi_{\mathcal{M}_{i+1}} = \tau$ is the finitary outcome, $\eta_{>i}$ begins with f, and $\eta_{>i} = \mathsf{f}\eta_{>i+1}$. T_{i+1} is the tree $T_i\{\substack{\succeq \mathsf{f} \\ \tau}\}$ for some $\tau \in T_i\{\substack{\succ \varnothing \\ \pi_{\mathcal{P}_i}}\}$. The labels ℓ_{i+1} of the tree T_{i+1} are defined from the labels ℓ_i of the tree T_i by setting $\ell_{i+1}(\tau') = \top$ if $\ell_i(\tau') = \top$, or $\ell_{i+1}(\tau') = \eta^*$ if $\ell_i(\tau') = \mathsf{f}\eta^*$. Thus σ^* is still on T_{i+1}, and $\ell_{i+1}(\sigma^*) \succeq \eta_{>i+1}$. Moreover, for each $\tau' \in T_{i+1}$, $\ell_i(\tau') \succ \eta_{>i}$ if and only if $\ell_{i+1}(\tau') \succ \eta_{>i+1}$. So σ is on $T_{i+1}\{\substack{\succ \eta_{>i+1} \\ \rho_j}\}$, as desired. ⊣

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

LEMMA 8.7. *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 8.5 $\Xi_{\langle e,i\rangle,j}$ 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 8.6 there is $\sigma \in T_{\langle e,i\rangle}$ 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 σ as the outcome. ⊣

REFERENCES

[1] E. ALLENDER, H. BUHRMAN, and M. KOUCKÝ, *What can be efficiently reduced to the Kolmogorov-random strings?* ***Annals of Pure and Applied Logic***, vol. 138 (2006), nos. 1–3, pp. 2–19.

[2] E. ALLENDER, L. FRIEDMAN, and W. GASARCH, *Limits on the computational power of random strings*. ***Information and Computation***, vol. 222 (2013), pp. 80–92.

[3] T. BAKER, J. GILL, and R. SOLOVAY, *Relativizations of the $\mathcal{P} =? \mathcal{NP}$ question*. ***SIAM Journal on Computing***, vol. 4 (1975), no. 4, pp. 431–442.

[4] R. E. BAYER, ***Lowness for computational speed***, Ph.D. thesis, University of California, Berkeley, and ProQuest LLC, 2012.

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press

[5] L. Bienvenu and R. Downey, *On low for speed oracles*, ***Journal of Computer and System Sciences***, vol. 108 (2020), pp. 49–63.

[6] M. Cai, R. G. Downey, R. Epstein, S. Lempp, and J. S. Miller, *Random strings and truth-table degrees of Turing complete c.e. sets*. ***Logical Methods in Computer Science***, vol. 10 (2014), no. 3:15, 24 pp.

[7] T. A. Slaman and R. Solovay, *When oracles do not help*, ***COLT*** (M. K. Warmuth and L. G. Valiant, editors), Morgan Kaufmann, Oxford, 1991, pp. 379–383.

SCHOOL OF MATHEMATICS AND STATISTICS
VICTORIA UNIVERSITY OF WELLINGTON
WELLINGTON, NEW ZEALAND

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF MICHIGAN
ANN ARBOR, MICHIGAN
E-mail: matthhar@umich.edu

https://doi.org/10.1017/jsl.2021.108 Published online by Cambridge University Press