Journal of Computer and System Sciences 108 (2020) 49–63

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

On low for speed oracles

Laurent Bienvenu [a,*], Rod Downey [b]

[a] LaBRI, CNRS & Université de Bordeaux, France
[b] School of Mathematics and Statistics, Victoria University of Wellington, New Zealand

ARTICLE INFO

Article history:
Received 18 June 2018
Received in revised form 11 July 2019
Accepted 15 August 2019
Available online 17 September 2019

Keywords:
Oracle computations
Lowness for speed

ABSTRACT

Relativizing computations of Turing machines to an oracle is a central concept in the theory of computation, both in complexity theory and in computability theory(!). Inspired by lowness notions from computability theory, Allender introduced the concept of "low for speed" oracles. An oracle A is low for speed if relativizing to A has essentially no effect on computational complexity, meaning that if a decidable language can be decided in time $f(n)$ with access to oracle A, then it can be decided in time $poly(f(n))$ without any oracle. The existence of non-computable such A's was later proven by Bayer and Slaman, who even constructed a computably enumerable one, and exhibited a number of properties of these oracles. In this paper, we pursue this line of research, answering the questions left by Bayer and Slaman and give further evidence that the class of low for speed oracles is a very rich one.

1. Introduction

The subject of this paper is oracle computation, more specifically the effect of oracles on the speed of computation. There are many notable results about oracles in classical complexity, beginning with the Baker-Gill-Solovay result [4] which asserts that there are oracles A such that $\mathsf{P}^A = \mathsf{NP}^A$, but that there are also 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. Eric Allender and his co-authors [1,2] showed that oracle access to the sets of random strings could give insight into basic complexity questions. For example, in [2], Allender et al. showed that $\mathsf{PSPACE} \subseteq \cap_U \mathsf{P}^{R_{K_U}} \cap \mathsf{COMP}$ where R_{K_U} denotes the strings whose prefix-free Kolmogorov complexity (relative to universal machine U) is at least their length, and COMP denotes the collection of computable sets. Later the "$\cap\mathsf{COMP}$" was removed by Cai et al. [7]. Thus we conclude that reductions to very complex sets like the random strings somehow give insight into very simple things like computable sets.

Inspired by lowness notions in computability theory, Allender asked whether there were non-trivial sets which were "low for speed" in that, as oracles, they did not accelerate running times of computations by more than a polynomial amount. Of course, as stated this makes little sense since using any X as oracle, we can decide membership in X in linear time, while without 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 oracle and polynomial p such that M' decides L in time $p \circ f$. (Here computation time of

* Corresponding author.
E-mail addresses: laurent.bienvenu@labri.fr (L. Bienvenu), rod.downey@vuw.ac.nz (R. Downey).

https://doi.org/10.1016/j.jcss.2019.08.007
0022-0000/

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 $L \in \mathsf{P}^X$, then $L \in \mathsf{P}$.*

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

Theorem 1.2 *(Bayer-Slaman [3]). There are non-computable, computably enumerable, sets X which are low for speed.*

Once their existence is established, it is natural to wonder what kind of sets might be low for speed. A precise characterization seems currently out of reach, but it is interesting to see how lowness for speed interacts with other computability-theoretic properties. One needs however to keep in mind that lowness for speed is *not* closed under Turing equivalence: as we saw above that in the $\mathbf{0}$ degree (computable sets) some members are low for speed and others that are not (on the other hand it is easy to see that if A is polynomial-time reducible to B and B is low for speed, then A is also low for speed).

In his PhD thesis, Bayer showed that if X is computably enumerable and of promptly simple Turing degree, then X is *not* low for speed, but also proved that this did not characterize the computable enumerable oracles that are low for speed. Bayer also studied the size of the set of low for speed oracles, where 'size' is understood in terms of Baire category. Surprisingly, whether the set of low for speed oracles is meager or co-meager depends on the answer of the famous $\mathsf{P} = ?\mathsf{NP}$ question.

In this paper, we continue Bayer and Slaman's investigation on the set of low for speed oracles. In the next section, we give an easier proof of the existence of non-computable low for speed oracles which does not require the full Bayer-Slaman machinery (but the oracle we construct is not computably enumerable). In Section 3, we focus on the computably enumerable low for speed oracles, and prove that – quite surprisingly – they cannot be low in the computability-theoretic sense, but can however be low_2. Finally, we pursue Bayer and Slaman's idea to study how large the set of low for speed oracles is, in terms of measure and category. In particular, we solve a question they left open by showing that the set of low for speed oracles has measure 0 and obtain some interesting connections with algorithmic randomness. Finally, though lowness for speed is not closed under Turing equivalence it is nonetheless natural to ask which Turing degrees contain a low for speed member, which is what Section 5 is about.

Throughout this paper, we will denote by $\{0,1\}^*$ the set of finite strings. In our setting, an oracle is a *language*, i.e., a subset of $\{0,1\}^*$; however, as is typical in computability theory, it is more convenient in some of the results we present below to view oracles as infinite binary sequences (whose set we denote by $\{0,1\}^\omega$), by first identifying finite strings with integers (the $(n+1)$-th string in the length-lexicographic order being identified with n) making the oracle a subset of $\mathbb{N}$ and then identifying the oracle with its characteristic sequence (the $(n+1)$-th bit is 1 if n belongs to the oracle, 0 otherwise). When building oracles X with certain computability-theoretic properties, viewed as infinite binary sequences, we will often need to refer to prefixes of X, which are themselves binary strings. To avoid confusion between *members* and *prefixes* of oracles, we will use Latin letters $x, y, z, \ldots$ to denote members of oracles, and Greek letters $\sigma, \tau, \ldots$ for prefixes of oracles. Two strings σ and τ are *incompatible* if for some $i < \min(|\sigma|, |\tau|)$, $\sigma(i) \neq \tau(i)$. We denote this by $\sigma \perp \tau$. The join $X \oplus Y$ of two infinite binary sequences X, Y is the sequence $X(0)Y(0)X(1)Y(1)\ldots$. Finally $X \upharpoonright n$ is the prefix of X of length n.

Our paper requires some knowledge of computability theory and algorithmic randomness. One can consult the book [9] for the results and concepts we allude to below. Our notation is mostly standard. We denote Cantor's pairing function by $\langle .,. \rangle$. We also fix an effective list (Φ_e) of all oracle Turing functionals (or machines: Φ_e^A is the Turing machine of index e with oracle A, which for a fixed A is a partial function from $\{0,1\}^*$ to $\{0,1\}$). For a given functional Φ_e and oracle A, $time(\Phi_e^A, x)$ denotes the running time of Φ_e on input x with oracle A (counting time according to the model of computation described above) and $time(\Phi_e^A)$ is the function $x \mapsto time(\Phi_e^A, x)$. We let (R_i) be an effective enumeration of all partial computable functions from $\{0,1\}^*$ to $\{0,1\}$. We denote the set of low for speed oracles by LFS, and the subset of LFS consisting of its non-computable elements by LFS^*.

2. Existence of non-computable low for speed sets

In this section we will present a simple proof of the existence of a non-computable low for speed oracle. Define the set S of strings by $S = \{0^{2^n} \mid n \in \mathbb{N}\}$ and – identifying S with a set of integers as discussed above – let $\mathbb{S}$ be the set of 'sparse' infinite binary sequences (viewed as sets of integers) that only contain elements from S, that is, $\mathbb{S} = \{X \in \{0,1\}^\omega \mid X \subseteq S\}$.

By extension, we say that a string σ is in $\mathbb{S}$ if it is a prefix of some element of $\mathbb{S}$. The interest of the set $\mathbb{S}$ is that there are only $O(n)$ strings in $\mathbb{S}$ of length n. Thus, given a Turing machine Φ, it is possible to simulate in time $poly(t)$ the behavior of Φ^X during t steps of computation on all $X \in \mathbb{S}$ (an idea which is already present in the Bayer-Slaman argument presented in the next section).

Theorem 2.1. *There exists a non-computable X which is low for speed.*

Proof. We want X to satisfy all requirements $\mathcal{R}_{(e,i)}$, where e, i range over integers, defined as follows

$\mathcal{R}_{(e,i)}$: either R_i is partial, or $\Phi_e^X \neq R_i$, or $\Phi_e^X = R_i$ but the computation of R_i via Φ_e^X can be simulated by a functional Ψ running in time polynomial in $time(\Phi_e^X)$.

We build our oracle X by finite extension. Let σ_0 be the empty string. At stage $s+1 = \langle e, i\rangle$, do the following.

(a) If there is an n and a $\tau \in \mathbb{S}$ extending σ_s such that $\Phi_e^\tau(n)$ and $R_i(n)$ both converge and have different values, then let σ_{s+1} be the first (say in length-lexicographic order) such string τ.
(b) If there is no such string τ, then take $\sigma_{s+1} = \sigma_s 0$.

Finally let X be the unique infinite sequence extending all σ_s. We claim that X is as wanted. Let us first prove that X must be incomputable. Suppose $X = R_i$ for a total R_i. Let e be an index such that Φ_e is the identity functional. By construction, when choosing the prefix τ of X at stage $s+1 = \langle e, i\rangle$, we must be in case (a), and thus τ is precisely chosen to ensure $X \neq R_i$, a contradiction. Let us now prove that X is low for speed. Fix a pair (e, i) let $s+1 = \langle e, i\rangle$, and let us see how σ_{s+1} was constructed. If we were in case (a) at that stage, we have ensured $\Phi_e^{\sigma_{s+1}} \perp R_i$ and thus $\Phi_e^X \perp R_i$, thereby satisfying $\mathcal{R}_{(e,i)}$. If we were in case (b), there are three subcases:

- Either R_i is partial, then the requirement $\mathcal{R}_{(e,i)}$ is satisfied.
- Or there is an n such that $\Phi_e^\tau(n) \uparrow$ for any extension τ of σ_s, in which case $\Phi_e^X(n) \uparrow$ and thus $\Phi_e^X \neq R_i$ should R_i be total.
- Or, if we are in neither of the two above cases, for every n there is an extension τ of σ_s such that $\Phi_e^\tau(n) \downarrow$, and for any such τ, we have $\Phi_e^\tau(n) = R_i(n)$. In this case, we can build a functional Ψ which computes R_i as follows. On input n, at stage t, it computes $\Phi_e^\tau(n)$ during t steps of computation for all $\tau \in \mathbb{S}$ of length t extending σ_s. If a τ is found such that $\Phi_e^\tau(n) \downarrow$, then we set $\Psi(n) = \Phi_e^\tau(n)$. As we already mentioned, there are only $O(t)$ strings of length t in $\mathbb{S}$ and it is obvious that they can be listed in polynomial time. Hence, simulating all computations $\Phi_e^\tau(n)$ during t steps can be done in time $p(t)$ for some polynomial t. This shows that for any $Y \in \mathbb{S}$ extending σ_s, if $\Phi_e^Y(n)$ returns (the value of $R_i(n)$) in time t, this is found out by the procedure Ψ at stage t, which corresponds to $\sum_{s \leq t} p(s) + O(1)$ steps of computation for Ψ, which is also polynomial in t. This being true for any $Y \in \mathbb{S}$ extending σ_s, we have in particular that $time(\Psi) = poly(time(\Phi_e^X))$. □

One should note that the case disjunction in this proof is a Σ_1/Π_1 dichotomy, and therefore one can carry out the construction below $\mathbf{0}'$, therefore establishing the existence of a $\mathbf{0}'$-computable set that is low for speed. This is weaker than the Bayer-Slaman result presented in the next section, which asserts the existence of a *c.e.* such set. However, this proof is simpler and, as we will see in the remainder of the paper, has further useful corollaries.

3. Computably enumerable low for speed sets

We now restrict ourselves to the computably enumerable (c.e.) sets, and study which of these can be low for speed. For the sake of completeness, we present the main ideas of the proof of Bayer and Slaman [3] that there are indeed c.e. sets in LFS*. We give a full proof of this result as no proof has appeared beyond Bayer's unpublished PhD Thesis.

Theorem 3.1 *(Bayer-Slaman Theorem). There exist c.e. non-computable sets that are low for speed.*

Proof sketch. The proof uses a tree-of-strategies argument. We need to satisfy

$$\mathcal{P}_e : \overline{A} \neq W_e,$$

and

$$\mathcal{L}_{e,i} : \text{If } \Phi_e^A = R_i \text{ total, then some } \Psi \text{ computes } R_i \text{ in time polynomial in } time(\Phi_e^A).$$

The $\mathcal{P}_e$-strategy is a standard Friedberg-Muchnik strategy on a tree. A node ρ devoted to this requirement picks a fresh follower x, waits for $x \in W_e[s]$ and if this happens puts x into A. The ρ-node has two outcomes $1 <_L 0$. 1 represents the fact we have actually diagonalized $\mathcal{P}_e$.

We will represent $\mathcal{L}_{e,i}$ on the tree of strategies as a node τ with outcomes $k_s <_L k_w <_L \infty <_L w$. The letters k_i represent the "kill" outcomes, meaning that we have or will successfully diagonalize $\mathcal{L}_{e,i}$, with k_s we have actually diagonalized, and k_w meaning that either we don't have to worry about it further or at some stage we will kill it, ∞ meaning that Ψ emulates Φ_e^A and w means that Φ_e^A is partial or R_i is partial. Before we discuss how this is played in the actual construction we will look at the basic module.

The basic module for $\mathcal{L}_{e,i}$ is the following. First, throughout the whole construction of A, we will promise that if we add an element x to A at stage t, then we must immediately also add all $y \in [x, t]$ (this is often referred to as a *dump construction*). This way, at any stage s, there will only be at most s strings α of length s that can potentially be a prefix of (the final) A. And thus – just like in the previous section – at stage s, it is possible to emulate all computations $\Phi_e^\alpha(x)[s]$ for all such α's and $x \leq s$ in time $poly(s)$.

During the construction, for every $x \leq s$ on which Ψ is not defined yet, it computes all $\Phi_e^\alpha(x)[s]$ for all potential prefixes α of A, and should one of them converge, defines $\Psi(x)$ to be the value of $\Phi_e^\alpha(x)$ for the α that has the fastest convergence. If no $\Phi_e^\alpha(x)[s]$ converges, $\Psi(x)$ remains undefined.

In the actual tree construction, we will have several versions of Ψ corresponding to the locations on the tree of strategies on a single level. Each of these, as we see below, will have its own idea as to what might constitute a possible future A-configuration, and hence for each such location τ it will be defining a version of $\Psi_\tau(x)$ only looking at τ-admissible A-configurations α as oracles for $\Phi_e^\alpha(x)$, as described below.

For a single $\mathcal{L}_{e,i}$ in isolation interacting with the $\mathcal{P}_f$, we will drop the τ, and only consider Ψ. For each x with $\Psi(x)\downarrow$, if $R_i \upharpoonright x \downarrow = \Psi(x) \upharpoonright x$ we say that x is *R_i-confirmed* at stage s. Now, if at some later stage we find a value x such that $R_i(x)\downarrow$ and $\Psi(x) \neq R_i(x)$, then we find the α such that $\Psi(x) = \Phi_e^\alpha$ and add elements into A so that α becomes a prefix of A. On the priority tree this corresponds to the outcome k_s. We will protect this with priority $\mathcal{L}_{e,i}$. This action ensures $\Phi_e^A \neq R_i$ and terminates the strategy. All strategies of lower priorities are then injured and must be reset. If we never find such an x, this means that either Φ_e^A is partial, or R_i is, or $\Psi = \Phi_e^A = R_i$ and by construction the running time of Ψ is polynomial in the running time of Φ_e^A.

The above is enough for a single strategy in isolation. We now consider the modifications necessary for a single $\mathcal{L}_{e,i}$ in interacting with *lower* priority $\mathcal{P}_f$ strategies which might want to add elements into A. The final key to the Bayer-Slaman proof is the following. Suppose that at some stage s a strategy of lower priority wants to add an interval $[y, s]$ of elements into A. The problem is that the computations on this configuration might be *slow*. Perhaps for some x of length $\leq s$ we have not as yet seen $\Phi_e^{A_s \cup [y,s]}(x)\downarrow$. Even more importantly, we don't even know that the value of this will agree with the value $\Psi(x)$ we have already defined. (The problem with a c.e. set construction is that if we move away from a configuration we cannot go back.)

The idea is the following. R_i has to *confirm* the computations, before we move to $A_s \cup [y, s]$. That is, if $n = n_{e,i,s}$ is the largest number for which we have defined $\Psi(n)$ and $m = m_{e,i,s}$ is the largest number for which we have seen that

$$\Psi \upharpoonright m = \Phi_e^A \upharpoonright m,$$

we would need to wait for a stage t where[1]

$$m_{e,i,t} > n_{e,i,s}$$

to validly move A to $A_t \cup [y, t]$. Suppose that at stage s, $\mathcal{P}_f$ asks us to make A extend $A_s \cup [y, s]$. We *know* that if we do this action, that is make A_t extend $A_s \cup [y, s]$, at some stage $t > s$, then after stage t the only possible future A-configurations will be ones extending $A_s \cup [z, t]$ (and hence $A_s \cup [z, s]$ for $z \leq y$) or extending $A_s \cup [q, u]$ for $q \geq t$, and $u > t$ and hence extending $[y, s]$. We will call possible A-configurations extending $A_s \cup [y', t]$ for *followers* $y' \leq y$ *y-admissible at stage t*. Thus, as we see below, in the construction we will be defining $\Psi(n)$ on new values n only using subsets of the overall tree of possible A-configurations. To wit: the strategy is predicated on the assumption that $R_i = \Psi_e^A$. Thus, the strategy "knows" that the computations will be R_i-confirmed. Hence it is guessing that in the future A will move to a y-admissible configuration. Now, if the longest R_i-confirmed $\Psi(x)$ computation at $m_{e,i,s}$, is not associated with a y-admissible configuration, we cannot yet move to there. More subtly, while waiting for the confirmation of the existing Φ_e^A computations, it would not be sensible to issue new $\Psi(m)$ computations for $m > m_{e,i,s}$ *using y-inadmissible configurations*, since this process might not terminate. Thus the strategy is clear: For a single $\mathcal{L}_{e,i}$ wanting to meet a $\mathcal{P}_f$ by making A_t enter $A_s \cup [y, t]$ for some $t \geq s$, we will only issue new $\Psi(m)$ values for $m > m_{e,i,s}$ using $\Phi_e^\alpha(m)[t]$-computations for y-admissible α.[2] We will do so until $\Psi \upharpoonright n_{e,i,s}$ is R_i-confirmed, or some y-admissible α gives us $\Phi_e^\alpha \upharpoonright n_{e,i,s} \neq \Psi \upharpoonright n_{e,i,s}$. Whilst we are waiting we will let $\mathcal{L}_{e,i}$ assert control and ask that A_t extends A_s and in the tree construction, we'd be playing outcome τw.

Now one of four things can happen.

[1] The reader should note the t for the m function, and the s for the n. More on this point below.

[2] The reader should note that we are not waiting for accessibility of the node representing $\mathcal{L}_{e,i}$ to issue these descriptions, but at every stage $t \geq s$. It is only that we are constraining what oracles to use.

1. While we are waiting for the R_i confirmation for y, an even higher priority $\mathcal{P}_{f'}$ asks us to extend $A_s \cup [y', t]$ at some $t > s$. In this case we'd replace y by y' and continue but now using y'-consistent configurations. (In the construction, by the way we assign followers it will be that $y' < y$. Moreover this can only happen a finite number of times as the priority is increasing.) The remaining cases assume that we are stuck on the highest priority y.
2. R_i-confirmation for y occurs at some stage $t > s$. In this case we would be free to let A_{t+1} extend $A_s \cup [y, t]$. This corresponds to outcome $\tau\infty$. The reader should note that after this stage, if we enumerate any new $\Psi \upharpoonright q$ computations they must be consistent with R_i, assuming the hypothesis of $\mathcal{L}_{e,i}$, namely $\Phi_e^A = R_i$. If they were not, then $\mathcal{L}_{e,i}$ would assert control and ask that A extend the α giving this R_i-inconsistent computation, winning it forever.
3. We never see an R_i-confirmation for y. In this case, A will extend A_s with priority $\mathcal{L}_{e,i}$, and we have a *global win* for $\mathcal{L}_{e,i}$. The true outcome of τ is w. (The reader should note that even though the outcome might be w, we might still issue infinitely many new values for Ψ.) As with all tree arguments, there will be another version of $\mathcal{P}_f$ guessing this Σ_2^0-outcome for $\mathcal{L}_{e,i}$, and it is this version which will meet $\mathcal{P}_f$.
4. Some y-consistent α gives us $\Phi_e^\alpha \upharpoonright n_{e,i,s} \neq \Psi \upharpoonright n_{e,i,s}$. Suppose that this disagreement occurs on argument $x \leq n_{e,i,s}$. In this last case, we would abandon this version of $\mathcal{P}_f$, and $\mathcal{L}_{e,i}$ would assert control to ask that A should extend one of α or β, where β was the configuration which gave $\Psi(x)$. Note that β will be of the form of (an initial segment of) A_t, or some $A_s \cup [q, t]$ for some q and α of the form $A_s \cup [r, t]$. For a single $\mathcal{L}_{e,i}$, we would choose $z = \max\{r, q\}$, and let $A_{t+1} = A_s \cup [z, t]$. We would be waiting for $R_i(x) \downarrow$ and choose one of the two to extend to diagonalize $\mathcal{L}_{e,i}$. Note that $\mathcal{L}_{e,i}$ will request that $A_{t'}$ extend A_{t+1} for $t' \geq t + 1$, until R_i chooses. This action would correspond to playing τk_w while we wait, and then τk_s should we see which R_i chooses and diagonalize. Note that this choice might never happen, but in either case we have diagonalized $\mathcal{L}_{e,i}$. We will say that τ *has transitioned into a Friedberg requirement,* in that our action will be the same as we would have done for some P_f: we want to move A.

Now we consider two low for speed requirements $\mathcal{L}_{e,i}$ and $\mathcal{L}_{e',i'}$ say at nodes τ_1 and τ_2, respectively. This would mean that $\mathcal{L}_{e,i}$ has higher priority than $\mathcal{L}_{e',i'}$. In the usual method of tree arguments, if $\tau_1 w \leq_L \tau_2$, then each time $\tau_1\infty$ is played we'd initialize τ_2, meaning that all of its simulations and the like would need to start again. The case of interest is $\tau_1\infty \preceq \tau_2$.

As above, we'd be in a situation where the construction desires us to extend $A_s \cup [y, t]$ for some $t \geq s$, either for the sake of some $\mathcal{P}_f$ guessing $\tau_2\infty$, or some $\mathcal{L}_{e'',i''}$ at some τ_3 guessing $\tau_2\infty$ (in the sense of Case 4 above, which is also a Friedberg-type situation). Before we would accede to this request, *both* of the τ_i's need to be made happy, meaning both R_i- and $R_{i'}$-confirmed. This happens in the obvious way.

First we would continue to extend A_s guessing $\tau_2 f$ until τ_2 processed the situation. Hence $\mathcal{L}_{e',i'}$ would only be issuing new Ψ_{τ_2} computations based on its view of y-consistent computations, that is y-τ_2-consistent computations, which only use α extending $A_s \cup [y', t]$ for $y' \leq y$. Clearly this might never return, and hence τ_1 needs to continue to issue new $\Psi_{\tau_1}(x)$ computations on all possible configurations (for only 2 requirements). If τ_2 never returns, τ_1 does not care. If τ_2 returns, it will be in one of the 4 situations above. That is, either it will either have morphed in to a Friedberg action, and then in the two requirement scenario, it is treated precisely as we did for one requirement by τ_1, or τ_1 returns as in Case 2. Now, τ_1, will work to extend $A_s \cup [y, t]$, and be exactly as in the original one requirement scenario. The only difference is that the version τ_2 guessing $\tau_1\infty$ will only be issuing descriptions Ψ_{τ_2} consistent with its guess, and hence y-τ_2-consistent, and now τ_1 only looks for y-consistent computations also.

Naturally, there will be a version of $\mathcal{L}_{e'i'}$ guessing $\tau_1 f$ which would act keeping A_u extending A_t (the stage that τ_2 was confirmed), and pressing τ_1 to confirm.

There are no difficulties extending this reasoning to more than 2 $\mathcal{L}_{e,i}$'s, using similar inductive strategies. We turn to some formal details, although we believe the reader could easily fill them in themselves at this point.

The construction. The priority tree will have nodes ρd for $d \in \{1, 0\}$, for each node ρ having odd length equal to $2e+1$ and associate with ρ a version $\mathcal{P}_\rho$ of $\mathcal{P}_e$. With each node τ of even length $2\langle e, i\rangle$, we put τc on the tree for $c \in \{k_s, k_w, \infty, w\}$, associate a version of $\mathcal{L}_{e,i}$, $\mathcal{L}_\tau$ with τ. Then the construction works in the obvious way. Beginning with the empty string λ, we generate a string TP_s which looks correct at stage s. This will have length $\leq s$.

Case 1: Suppose that we have gotten to ν and $|\nu| = 2e + 1$. In this case we will write ρ for ν to emphasize that it is a $\mathcal{P}$-type node.
1. It $\mathcal{P}_\rho$ is already met (and has not been re-initialized), then $\rho 1$ will be the next node to be processed.
2. If $\mathcal{P}_\rho$ is not met and has no follower, we give $\mathcal{P}_\rho$ a fresh follower y if it does not have one, and let $TP_s = \rho 0$.
3. If $\mathcal{P}_\rho$ already has a follower y, see if $y \in W_{e(\rho),s}$. If $y \notin W_{e(\rho),s}$, the next node to be examined at this stage will be $\rho 0$.
4. If $y \in W_{e(\rho),s}$, we set $TP_s = \rho 1$. If $y \in A_s$ then $\mathcal{P}_\rho$ is met, and we will keep A_t extending A_s with priority ρ. Otherwise, for each $\tau\infty \preceq \rho$, declare τ's target $t(\tau, s + 1) = s$ (which is greater than $\max\{n_{\rho,s}, t(\tau, s)\}$ and simpler to write) where $2\langle e, i\rangle = |\tau|$. For the longest such τ, declare that until the next τ-stage, τ-consistent strings will only be of the form $A_s \cup [y', t]$ for $t \geq s$ and followers $y' \leq y$. We will say that y is associated with τ. In the case this is the stage we declared $\mathcal{P}_\rho$ active, we let $TP_s = \rho 1$.

   If there is no such $\tau\infty \preceq \rho$, we will let $A_{s_+1} = A_s \cup [y, s]$. Let $TP_s = \nu 1$, and declare $\mathcal{P}_\nu$ as met.

Case 2: If $|\nu| = 2\langle e, i\rangle$. For clarity we will write τ for ν to emphasize that this is a $\mathcal{L}$-type requirement.

1. This is the first time we visit ν (since initialization), give it target $t(\tau, s+1) = 1$. Let $TP_s = \tau w$.
2. If τ has been declared met and this has not been initialized, the next node is τk_s.
3. (i) If we see two τ-admissible A_s-configurations $\alpha = A_s \cup [z, s]$ and $\beta = A_s \cup [y, s]$ and some x with $\Phi_e^{\alpha}(x) \downarrow \neq \Phi_e^{\beta}(x) \downarrow$ and wlog $z < y$, then we would say that τ has become Friedberg, and we wish to make A_t extend β at some future stage. In this case $TP_s = \tau k_w$. We will need to treat τ now as we did $\mathcal{P}_\nu$ above. That is, for each $\tau'\infty \preceq \tau$, declare τ''s new target $t(\tau', s+1) = s$ where $2\langle e', i'\rangle = |\tau'|$. For the longest such τ', declare that until the next τ-stage, τ'-admissible strings will only be of the form $A_s \cup [y', t]$ for $t \geq s$ and $y' \leq y$. We will say that y is associated with τ'.

   (ii) If τ has become Friedberg and we are visiting it again, declare that τ is *waiting*. Promise that A_t will extend A_{s+1} with priority τ henceforth unless the next item applies, playing τk_w.

   – If τ has become Friedberg and is waiting, there will be two τ-admissible configurations α and β and an argument x where $\Phi_e^{\alpha}(x) \neq \Phi_e^{\beta}(x)$. If $R_{i(\tau)}(x) \downarrow [s] \neq \Phi_e^{A_s}$, declare that $\mathcal{L}_\tau$ is met, and play outcome τk_s, as we will do henceforth. Promise that A_t extends A_s with priority τk_s. If $R_{i(\tau)}(x) = \Phi_e^{A_s}$, we will need to move to the other configuration, which will be of the form $A_s \cup [y, s]$ for some y. Again we cannot do this immediately, but will treat this as a new Friedberg action if this is the first time $R_i(x)$ has chosen. We declare τ as pending. Again we will treat the request to move to $A_s \cup [y, s]$ in the same way as we did P_ν as above.[3] Let $TP_s = \tau k_w$. If there is no $\tau'\infty \preceq \tau$ we set $A_{s+1} = A_s \cup [y, s]$ and $TP_s = \tau k_s$. We declare $\mathcal{L}_\tau$ as met and promise to keep A_t extending A_s for $t > s$ with priority τk_s.

For the rest of this case we are assuming that τ has not become Friedberg.

1. If $t(\tau, s)$ is defined see if $R_{i(\tau)}$ has confirmed $\Psi_\tau \upharpoonright t(\tau, s)$, using τ-admissible A_s-configurations α. If this admissibility was determined by some y associated with some P_σ for σ extending τ then for each $\tau'\infty \preceq \tau$, declare their targets to be $t(\tau', s) = s$ where $2\langle e, i\rangle$ is $|\tau'|$. For the longest such τ', we associate y with τ'. The τ'-admissible configurations are now determined as we did for $\mathcal{P}_\nu$, as those associated with $y' \leq y$. If there is no such τ' define $TP_s = \tau\infty$, and let $A_{s+1} = A_s \cup [y, s]$. This action will meet the active $\mathcal{P}_\xi$ with follower y is there was one. If there instead was a $\mathcal{L}_\xi$ generating y then if ξ was pending, it will now be met. Otherwise make the $\mathcal{L}_\xi$ generating y waiting but now extending one of the two strings which kill it.
   If τ is not associated with any y at this stage, then we declare that $t(\tau, s+1) = t(\tau, s) + 1$, and that $\tau\infty$ is the next node.
   Finally, if $t(\tau, s)$ is defined and $R_{i(\tau)}$ has not confirmed $\Psi_\tau \upharpoonright t(\tau, s)$, declare τw to be the next node.

Finally, at the end of stage, we increase the definition of any $\Psi_\tau(x)$ for any τ consistent with TP_s and where $\Phi^{\alpha}(x) \downarrow [s]$ for the first time for α τ-admissible, initializing things strictly right of it.

This ends the construction.

The verification proves by simultaneous induction that if $\nu \prec TP = \lim_s TP_s$, then if ν has odd length, it will receive attention finitely often and will be met, and if ν has even length, if $\nu k_i \prec TP$, it is met by diagonalization, if $\nu\infty \prec TP$ then Ψ_τ correctly emulates $\Phi_{e(\nu)}$, and $m(\nu, s) \to \infty$, so that $R_{i(\nu)} = \Phi_{e(\nu)}^{A}$, and if $\nu w \prec TP$, then $m(\nu, s) \not\to \infty$. In the odd case we would appoint a follower at some stage s. By induction we may assume that ν is visited infinitely often. Moreover as this is the true path some follower appointed will be immortal. If this follower y never occurs in $W_{e(\nu),s}$ there is nothing to prove as then $\nu 0 \prec TP$ and will be played each time ν is visited from some point onwards. If the follower is realized by occurring in $W_{e(\nu),s}$, then by assumption we will visit ν again, and put y into A_{s+1}, and henceforth play $\nu 1$.

In the case that ν has even length, suppose that we are at stage where we are never again strictly left of ν, nor dealing with diagonalizations of strictly higher priority. Then Ψ_τ will never again be initialized, and ν has priority. If we saw ν-consistent strings α, β we would thereafter play outcome νk_w or νk_s, and ν would be met as it would be turned into a Friedberg requirement and hence met for the same reason as for ν odd. Assuming that this does not occur, we can suppose that any pair of ν-admissible strings give the same answer as Ψ_ν when used as oracles for $\Phi_{e(\nu)}$. Note that if $m(\nu, s) \not\to \infty$, then from some point onwards, we always play νw when we visit ν. Furthermore, this means that either Φ_e^{A} is not total since there must be some argument upon which no ν-admissible potential A paths (which *are the A-paths if* $\nu\infty \prec TP$) gives a halting computation, or R_i is not total. So we may also suppose that $m(\nu, s) \to \infty$. The construction ensures that we only ever move to τ-consistent strings from some point onwards. Numbers which will enter A will also come from nodes η extending $\nu\infty$, or be canceled at $\nu\infty$ stages. This mechanism of the construction means that definitions of Ψ_ν come from ν-admissible and confirmed computations, before we change to a new admissible path. Thus Ψ_ν correctly emulates $\Phi_{e(\nu)}^{A}$. The result follows. □

Within the c.e. sets, one would expect that a low for speed c.e. set would be one with little computational power, in the same way that sets low for 1-randomness are all (super-)low (see Nies [18]). The next theorem is therefore quite surprising.

[3] That is, again each $\tau'\infty \preceq \tau$, declare τ''s target $t(\tau', s+1) = s$ where $2\langle e', i'\rangle$ is $|\tau'|$. For the longest such τ', declare that until the next τ-stage, τ'-admissible strings will only be of the form $A_s \cup [y', t]$ for $t \geq s$ and followers $y' \leq y$. We will say that y is associated with τ'.

Theorem 3.2. *If A is non-computable, c.e. and of low Turing degree (i.e. $A' \equiv_T \emptyset'$), then A is* not *low for speed.*

Proof. Assume that A is not computable, is c.e., and is low. Let (Φ_e, p_e) be an enumeration of pairs of one functional and one polynomial with coefficients in $\mathbb{N}$. We will build a Turing functional Ψ and a computable set R such that $\Psi^A = R$. This is our global requirement and we make the following global commitment: if a value $R(n)$ gets defined at some stage, $\Psi^X(n)$ is immediately defined to be equal to $R(n)$ for all X's on which $\Psi^X(n)$ is still undefined. We want to satisfy, for each e:

$$(\mathcal{R}_e) : \Phi_e \text{ does not compute } R \text{ in time } p_e(time(\Psi^A))$$

thus proving that A is not low for speed. The strategy for a single requirement $(\mathcal{R}_e)$ is the following. Throughout the construction, we build a 'verifier', i.e., a partial computable S such that $S(e, .)$ is the attempt by the $(\mathcal{R}_e)$-strategy to guess A, that is, the $(\mathcal{R}_e)$-strategy will try to have $S(e, l) = A \upharpoonright l$ for all l (which is doomed to failure as A is non-computable). We also define an auxiliary functional Θ common to all strategies whose index we know in advance, and use the lowness of A to obtain a computable 0-1 valued function $h(., .)$ such that $\lim_t h(e, t)$ exists for all e, and equals 1 when $\Theta^A(e) \downarrow$, 0 otherwise. (Informally, $\Theta^X(e) \downarrow$ means that a prefix of X is believed to be a prefix of A at some stage of the strategy for $(\mathcal{R}_e)$, and this will cause the strategy to enter Case 3 as described below.)

For the $(\mathcal{R}_e)$-strategy, we will need a variable l which is initially set to 0; for each l we will also have a pair of variables (s_l, t_l) initially set to 0 (l, s_l and t_l are 'local' variables, other strategies will have their own l, s_l and t_l). Moreover, R, S and Θ are initially undefined everywhere. Then the procedure does the following.

Step I. Increase l by 1 and take for witness w_l the smallest integer which has not been picked as a witness by any strategy for any requirement.

Step II. Assign to s_l the value of the current stage. Set $\Psi^{A_{s_l} \upharpoonright l}(w_l) = 0$, and let t_l be the time taken by this computation.

Step III. Simulate $\Phi_e(w_l)$ during $p_e(t_l)$ steps of computation, and distinguish three cases.

Case 1: $\Phi_e(w_l)$ returns 1 in $\leq p_e(t_l)$ steps. In this case, we set $R(w_l) = 0$ and $R(w_i) = 0$ for all other witnesses w_i currently used by our $(\mathcal{R}_e)$-strategy. As indicated above, we further commit to having $\Psi^A(w_l) = 0$ even after potential future A-changes. This way we ensure $\Phi_e \neq R = \Psi^A$, thus immediately satisfying $(\mathcal{R}_e)$. We then mark this requirement as satisfied (meaning it will never receive attention again), stop the strategy for this requirement and move on to other requirements.

Case 2: $\Phi_e(w_l)$ returns 0 in $\leq p_e(t_l)$ steps. In this case, we do not define $R(w_l)$ just yet. Instead, we first check whether for all $i < l$, $S(e, i) = A_{s_l} \upharpoonright i$.

(a) If it is not the case, let i be smallest such that $S(e, i) \neq A_{s_l} \upharpoonright i$. This means that $A \upharpoonright i$ has changed between stages s_i and s_l, and so $A_{s_i} \upharpoonright i$ is not a true initial segment of A. Thus, even though we had set $\Psi^{A_{s_i} \upharpoonright i}(w_i) = 0$ (during Step II for i), we are free to set $\Psi^{A_{s_l} \upharpoonright i}(w_i) = 1$ (and commit to keeping $\Psi^A(w_i) = 1$), and then we can finally set $R(w_i) = 1 = \Psi^A(w_i) \neq \Phi_e(w_i)$. We also set $R(w_j) = 0$ for all other witnesses w_j currently used by our $(\mathcal{R}_e)$-strategy, mark requirement $(\mathcal{R}_e)$ as satisfied and terminate the strategy.
(b) If it is indeed the case, set $S(e, l) = A_{s_l} \upharpoonright l$ and then go back to Step I.

Case 3: $\Phi_e(w_l)$ is still undefined after $p_e(t_l)$ steps. In this case, we set $\Theta^{A_{s_l} \upharpoonright l}(e) \downarrow$ (which should be interpreted as signalling that we have entered Case 3). Observe that if $A_{s_l} \upharpoonright l$ is a true prefix of A, this implies $\Theta^A(e) \downarrow$ and therefore we would have $\lim_t h(e, t) = 1$. We again distinguish two subcases.

(a) The current value $h(e, s_l)$ is 0. Then we wait for a stage $t > s$ such that either $h(e, t) = 1$ or $A_t \upharpoonright l \neq A_{s_l} \upharpoonright l$ (one of the two must happen as we explained above). If the former happens first we move to subcase (b) below. If the latter happens first, we go back to Step II.
(b) The current value $h(e, s_l)$ is 1. We then set $R(w_l) = 0$ and commit to $\Psi^A(w_l) = 0$ and further set $R(w_i) = 0$ for all other witnesses, and commit to $\Psi^A(w_i) = 0$ as well. Then we wait – possibly forever – for a later stage $t > s_l$ where $h(e, t) = 0$ *and* $A_t \upharpoonright l \neq A_{s_l} \upharpoonright l$ (moving on to other requirements while waiting). If this happens, we come back to our $(\mathcal{R}_e)$-strategy where we left it and go back to Step I.

We claim that this strategy satisfies the requirement $(\mathcal{R}_e)$. First let us see why the strategy eventually stops acting. We can only find ourselves in Case 3b of Step III finitely many times during the whole strategy as each passage through this case causes a flip of $h(e, .)$, and we know $h(e, .)$ converges. Similarly, Case 2b of Step III can also happen only finitely often, because otherwise for every i, for every $l > i$, $S(e, i) = A_{s_l} \upharpoonright i$ which, since the sequence s_l tends to ∞, means $S(e, i) = A \upharpoonright i$ for all i, and thus A would be computed by $S(e, .)$, a contradiction. Since Case 3b and Case 2b are the only cases that cause

l to change by going back to Step I, l must eventually stabilize. Now, once l has reached its final value, Case 3a can only happen finitely many times at level l since $A_s \upharpoonright l$ can only change 2^l times.

Thus we either eventually end up in Case 1, or Case 2a, or a terminal Case 3b (i.e., the strategy enters Case 3b and waits there forever). This proves that the strategy stops acting eventually. To see that it further succeeds, notice that Case 1 and Case 2a guarantee immediate success. It remains to check the scenario of a terminal Case 3b. Suppose the terminal Case 3b happens for some $A_{s_l} \upharpoonright l$ which is not a prefix of A, this would mean that $\Psi^A(w_l)$ has not been defined yet and thus, should nothing else happen, we would have $\lim_t h(e,t) = 0$ and would see a change in $A \upharpoonright l$, thus leaving this occurrence of Case 3b, a contradiction. So $A_{s_l} \upharpoonright l$ is indeed a prefix of A and by construction $\Psi^A(w_l)$ returns $0 = R(w_l)$ in a number of steps t_l while $\Phi_e(w_l)$ does not return in less than $p_e(t_l)$ steps, thus the requirement is satisfied.

To finish the proof, it remains to notice a few things. First, this is an injury-free construction, different strategies pick different witnesses and thus do not interact with each other. It thus suffices to address all requirements by dovetailing. Moreover each strategy, when it stops acting, has defined $R(w)$ for all its used witnesses w (be it in Case 1, Case 2a or Case 3b). Given the way witnesses are picked at Step I, this guarantees that R will be defined everywhere. □

It is important to note that the above result fails to hold outside of the c.e. setting.

Theorem 3.3. *There exists a low, non-computable set X which is low for speed.*

Proof. See next section (Theorem 4.2). □

We next ask whether Theorem 3.2 is tight in terms of the lowness/highness hierarchy. Recall that A is low_2 if $A'' \equiv_T \emptyset''$.

Theorem 3.4. *There is a low_2 c.e. set that is low for speed.*

Proof. The formal details of the proof of Beyer-Slaman Theorem are sufficiently complex that they hide the main ideas. In this proof we will indicate the modifications necessary to make the proof work to make the set A low_2, without resorting to writing the whole proof in detail.

This time we must build A to satisfy the requirements of the Beyer-Slaman Theorem $\mathcal{P}_e$ and $\mathcal{L}_{e,i}$, and additionally:

$$\mathcal{N}_e : (\Delta_e^A \text{ is total}) \to (\Theta_e \text{ is total}),$$

where (Θ_e) is a sequence of functionals we build. And Δ_e represents the e-th Turing procedure (We use Δ_e so as to separate this requirement from the $\mathcal{L}_{e,i}$. Since we will be using a finitely branching priority tree we can replace $\mathcal{N}_e$ by the following:

$$\mathcal{N}_e : \exists^\infty s(\Delta_e^A \text{ believed to be total at } s) \to \Delta_e^A \text{ total}.$$

We have a node η devoted to his requirement. η will have two outcomes $\infty < f$. So by this re-statement, we mean that each time we believe Δ_e^A to be total playing $\eta\infty$, we will increment Θ_e, and hence Σ_3^A will be Δ_3^0, as the true path of the priority tree will be Π_2^0.

The basic idea is that at stages where the length of convergence $\ell(e,s) = \max\{x \mid \forall y \leq x \Delta_e^A(y)\downarrow\}$ increases we would like to play $\eta\infty$ and then preserve this computation.

To achieve this goal, we will add to the priority tree, not only mother nodes η where we build Θ, but below the outcome $\eta\infty$ nodes of the form ν_{η_x} trying to preserve $\Theta^A(x)[s]$, at any stage s we visit them. These nodes have only one outcome o, since if $\eta\infty \prec TP$, we must enforce ν_{η_x} is met.

Of course, interpolated in these sub-requirements will be the ρ-type nodes (for $\mathcal{P}_f$) and τ-nodes ($\mathcal{L}_{e',i}$), with these η_x nodes arbitrarily deep down the tree, all occurring at some level for a fixed x. Note that they *only* occur below the ∞ outcome of η. Whenever we pass some ν_{η_x} and the length $\ell(\eta,s) > x$, we play outcome o, meaning that we think we can preserve this computation and initialize all requirements of lower priority if this is the first time since ν_{η_x} was initialized.

We now consider the interactions between these requirements. The key tension is that once we agree to move to an A configuration of the form $\hat{A} = A_s \cup [n,s]$, then we are more or less committed to this. This is because the Friedberg requirement (or one morphed into Friedberg) which began this request has limited the admissibility of α's which it allows various $\tau\infty$ requirements to build their simulations Ψ_τ. If now we were denied the ability to move to $\hat{A}$ at some future stage, then this belief is wrong. This fact means that if a lowness type requirement asks us to preserve the A_s current configuration, this causes real conflict.

As we have seen, this timing conflict is strong enough that it is impossible to resolve if we want to make A low, but we have a way around this if A needs only be low_2.

$\mathcal{P}$-type requirements above η or $\mathcal{L}$ ones that have morphed into Friedberg ones will simply initialize η. This can happen only finitely often.

Consider an $\mathcal{L}$-type requirement at a node τ. If $\tau\infty \preceq \eta$, then there is no real problem, since η will await the conclusion of any pending confirmations before it actually believes the $\ell(\eta,s)$ computation. That is, if we desired at some stage to

make A extend $A_s \cup [y, s]$ and we send confirmation requests up the priority tree if this request had reached τ and τ was waiting for $R_{i(\tau)}$ to grow to confirm $\Psi_\tau \upharpoonright t(\tau, s)$, we would not pass $\tau\infty$ in the construction until the request was processes, and when the highest priority request was processed, we would set $A_{t+1} = A_t \cup [y, t]$. Thus we would not even see an η computation to preserve, except at a stage when there are no pending computations.

As always, the difficult configuration is when we have $\eta\infty \preceq \tau$.

Now for a ρ requirement with $\eta\infty \preceq \rho \preceq \tau$ we would allow this to initialize τ as the action is finitary, and the same is true of an ν_{η_x} in place of ρ. This is similarly true for $\eta\infty \preceq \tau\infty \preceq \rho \preceq \nu_{\eta_x}$, in the sense that ν_{η_x} would be initialized when ρ acts.

Thus the key configuration is $\eta\infty \preceq \tau\infty \preceq \nu_{\eta_x} \prec \rho$, since there are infinitely many such ρ.

The problem is that at a ρ stage we might issue a desire to make $\hat{A} = A_s \cup [y, s]$ be the new configuration. ρ is waiting to do this until confirmation by τ, but in the interim ν_{η_x} wakes up and declares that wants to preserve some computation, whilst we are committing to $\hat{A} = A_s \cup [n, s]$ should we get R_τ confirmation.

The solution to this is to realize that this cannot happen. In the cone below $\eta\infty$ the only outcome of η_x is o. That is, the follower y of ρ can only be appointed *after* we have seen $\ell(\eta, s) > x$, and at a $\tau\infty$ stage. Thus it will be too big to impact the computation for x.

The remaining details go through as before. □

We can also combine the same ideas (dump construction together with awaiting for certification) with a more or less standard proof that there exists an incomplete c.e. set A of high Turing degree (i.e., $A' \equiv_T \emptyset''$) to get the following. A little care is needed with the coding markers for highness.

Theorem 3.5. *There is a high c.e. set A which is low for speed.*

Proof. We sketch the modifications needed for the highness requirements. To achieve this, for each e, we need coding markers $c(e, i, s)$ such that if $e \in$ Tot then for almost all i, $c(e, i, s) \in A$, and if $e \in$ Fin then only finitely many enter. The location of a coding marker may be moved by the entry of some $z \leq c(e, i, s)$ into A_s. We set aside separate parts of ω for each e.

Now the new highness $\mathcal{H}_e$ requirements replace $\mathcal{P}_j$. Consider the action at a node ρ. At stage s, ρ will have a least unused coding marker $c(e, i, s)$ which is not restrained. This can be thought of as taking the place of y. We see that φ_e halts on another input so that Tot looks correct, and we desire to put $c(e, i, s)$ into A. Again the target is $A_t \cup [c(e, i, s), t]$ for some $t \geq s$. The machinery in the Bayer-Slaman Theorem will allow this almost always along the true path. Now, when we finally get to do this, at that stage, we will select new coding locations for

(i) All $c(e, j, t + 1)$ for $j > i$, and for
(ii) Any $c(e', j', t)$ where some $c(e, i, s) < c(e', k, t)$ for some k.

All we need to do is choose these so that they only move finitely often in the Fin case. Thus, for example, initially we might have $c(0, 0, s) < c(1, 0, s)$. Suppose that at stage t we enumerate $[c(0, 0, s), t]$ into A and $c(1, 0, s)$ was still alive at stage t, then we'd ensure that $c(0, 1, t + 1) > c(1, 0, t + 1)$, so that $\mathcal{H}_0$ could not force $\mathcal{H}_1$'s markers to ∞. With such a sharing strategy, there are no new problems and the construction then goes through along the same lines. □

It might seem tempting to conjecture that if $\mathbf{a}$ is a degree computably enumerable in, and strictly above, $\mathbf{0}'$ then there is some low for speed c.e. set A with $A' \in \mathbf{a}$. That is, the *only* restriction is lowness in terms of the jump hierarchy. However, this is not true. Bayer [3] showed that no c.e. set of promptly simple degree can be low for speed. And Shore [21] and Cooper [8] each showed that there are $\mathbf{a}$ degrees which are computably enumerable in, and strictly above, $\mathbf{0}'$ and such that if $A' \in \mathbf{a}$, then A has promptly simple degree. Thus no such set can be low for speed.

Note that we do not say in Theorem 3.5 that A is Turing incomplete, because this in fact follows from lowness for speed, as we will see in Proposition 5.6.

4. How big is LFS?

Bayer and Slaman showed that whether LFS is meager or not... depends on the answer to P vs NP question! More precisely, if P = NP, then LFS is co-meager (more precisely, every 2-generic is low for speed), while if P ≠ NP, then LFS is meager (more precisely, every weakly 1-generic is not low for speed).[4] They left as an open question whether LFS has Lebesgue measure 0 or 1 (by Kolmogorov's 0/1-law, it has to be one or the other). One might expect that, just like the meagerness of LFS depends on the P vs NP question, its measure depends on complexity-theoretic assumptions, such as the 'P vs BPP' question. This is not the case: we show that LFS is – unconditionally – a nullset.

[4] One may consult [9, Section 2.24] for an introduction to effective genericity.

Theorem 4.1. *The set* LFS *has measure* 0*. Indeed, no Schnorr random is low for speed.*

Proof. We first build a computable set R with the following properties: (1) the set R contains at most one string of any given length and (2) for all e, if Φ_e computes R, then $time(\Phi_e, x) > 2^{|x|}$ for almost all x. To do so, we declare at the beginning of the construction all indices e 'active', and for every n in order, do the following. We compute $\Phi_e(x)$ during $2^{|x|}$ stages of computation for all x of length n and all currently active $e \leq n$. If for some pair (e, x) we see that $\Phi_e(x)$ converges, we take the smallest such pair in the lexicographic order (smallest e first, then smallest x), we diagonalize against Φ_e by setting $R(x) = 1 - \Phi_e(x)$ (thus ensuring $\Phi_e \neq R$), as well as $R(y) = 0$ for all $y \neq x$ of length n, and then declare e inactive from now on. If no such pair (e, x) exists, set $R(y) = 0$ for all y of length n. This finishes the construction of R. It is clear that R is computable (it is even in EXPTIME) and that it contains at most one string of each length. To verify property (2), suppose Φ_e computes R, and observe that for any x such that $time(\Phi_e, x) \leq 2^{|x|}$, the only way Φ_e can escape being diagonalized against by R is when some Φ_i with $i < e$ is diagonalized against on strings of length n, but this can only happen e times. Thus if Φ_e does indeed compute R, it must do so in time greater than $2^{|x|}$ for almost all x.

Now we need to see how to speed up computations with a Schnorr random oracle. Consider the following procedure Ψ. Given oracle Z and input x, $\Psi^Z(x)$ first splits Z (viewed as a binary sequence) as $Z = \zeta_1\zeta_2\ldots$ with $|\zeta_i| = i$ and $\Psi^Z(x)$ returns 0 if $x = \zeta_{|x|}$ (thus the resulting computation is polynomial in $|x|$), and $\Psi^Z(x) = R(x)$ otherwise, using a fixed procedure to compute R. So there is a polynomial p such that for any Z, $time(\Psi^Z, x) \leq p(|x|)$ for infinitely many x's. Furthermore, we can only have $\Psi^Z(x) \neq R(x)$ if $x = \zeta_{|x|}$ and $\zeta_{|x|}$ happens to be the only string of its length in R. This has probability at most $2^{-|x|}$ ('at most' because R can also have no string of length $|x|$ at all) if Z is chosen at random. This means that, by setting $C_n = \{Z \mid (\exists x)\, |x| = n \wedge \Psi^Z(x) \neq R(x)\}$, we have $\lambda(C_n) \leq 2^{-n}$.

The C_n's are uniformly computable clopen subsets of $\{0,1\}^\omega$ because Ψ is a tt-functional. Thus, a Schnorr random A can only belong to finitely many C_n's (see for example [5, Lemma 1.5.9]), meaning that $\Psi^A(x) = R(x)$ for almost all x. Thus there is a finite variation $\hat{\Psi}$ of Ψ such that $\hat{\Psi}^A = R$, and $\hat{\Psi}^A(x)$ is computed in polynomial time for infinitely many x while $time(\Phi_i, x) > 2^{|x|}$ for any Φ_i computing R and almost all x. This shows that A is not low for speed. □

On the other hand, one can show that LFS is large in the set-theoretic sense, in that it has the size of the continuum. While this easily follows from the results of the next section, we can prove more, namely, we can build a perfect Π^0_1 class (i.e., a Π^0_1 class with no isolated point) of non-computable low for speed sets.

Theorem 4.2. *There exists a perfect* Π^0_1 *class all of whose members are non-computable and low for speed.*

To prove this theorem we shall use the notion of function trees (a.k.a. *f-trees*; here we follow the terminology of [22, Chapter 12]). An f-tree is a total[5] function $T : \{0,1\}^* \to \{0,1\}^*$ such that for every σ, $T(\sigma 0)$ and $T(\sigma 1)$ are strict extensions of $T(\sigma)$ and $T(\sigma 0) \perp T(\sigma 1)$. We say that τ is a node of T if $\tau \in rng(T)$, and the level of the node $T(\sigma)$ is $|\sigma|$. The children of $T(\sigma)$ are $T(\sigma 0)$ and $T(\sigma 1)$. The *level* of node $T(\sigma)$ is $|\sigma|$. A tree T' is a sub-f-tree of T, which we denote by $T' \preccurlyeq T$ when every node of S is a node of T. An infinite binary sequence Z is a path on an f-tree T if infinitely many prefixes of Z are nodes of T. The set of paths of T is denoted by $[T]$.

Proof (of Theorem 4.2). We want every member A of our Π^0_1 class to satisfy requirements

$$\mathcal{P}_e : \text{If } R_e \text{ is total, then } A \neq R_e,$$

and

$$\mathcal{L}_{e,i} : \text{If } \Phi_e^A = R_i \text{ total, then some } \Psi \text{ computes } R_i \text{ in time polynomial in } time(\Phi_e^A),$$

which we order in some effective fashion.

We prove the theorem by a full approximation over total computable function trees, that is, we start with an f-tree T_0 which will evolve over time, and denoting T_s the s-th version of our tree, we will ensure that

$$T_0 \succcurlyeq T_1 \succcurlyeq T_2 \succcurlyeq T_3 \succcurlyeq \ldots$$

and then take $\mathcal{C} = \bigcap_s [T_s]$ as our Π^0_1 class.

For every s, T_{s+1} will be obtained from T_s via the following type of transformation. Given an f-tree T, a string σ and two strict extensions σ', σ'' of σ, we say that $\tau' = T(\sigma')$ and $\tau'' = T(\sigma'')$ *become the children of* $\tau = T(\sigma)$ to mean that we are transforming the f-tree T into a tree $\hat{T}$ where for all ξ, $\hat{T}(\sigma 0\xi) = T(\sigma'\xi)$ and $\hat{T}(\sigma 1\xi) = T(\sigma''\xi)$, and $\hat{T}(\zeta) = T(\zeta)$ for all other strings ζ. (Note that this transformation implies $\hat{T} \preccurlyeq T$).

Define T_0 to be the tree whose paths are exactly the elements of $\mathbb{S}$ defined in Section 2, defined in a canonical way (the nodes of level l are strings of length s_l, where s_l is the l-th element of S). For any given s, the nodes of T_s of level k

[5] f-trees are defined in [22] as partial functions, but for our purposes total f-trees are enough.

are responsible for making all paths below them satisfy the k-th requirement. When changing a tree from T_s to T_{s+1}, some nodes experience a level change, in which case they begin implementing the strategy for their new requirement.

The strategy (in isolation) for a node τ to satisfy $\mathcal{P}_e$ is easy: Wait for a stage t such that $R_e[t]$ is defined on an initial segment of $\mathbb{N}$ longer than both children ξ_0 and ξ_1 of τ. Since ξ_0, ξ_1 are incomparable, we can find $i \in \{0,1\}$ such that $\xi_i \perp R_e$ and we let the children of ξ_i become the children of τ.

The strategy (in isolation) for a node τ to satisfy $\mathcal{L}_{e,i}$ follows the same idea as in Theorem 2.1 that is, the strategy is to construct a functional Ψ which runs in parallel Φ_e^ξ for all nodes ξ below τ that are still in the f-tree T. By the sparseness of T (remember that we started with T_0 to be the tree whose nodes are members of S), every $\Phi_e^\xi(x)$ is emulated in time $poly(time(\Phi_e^\xi, x))$. Whenever we find an x such that $\Psi(x)$ is still undefined and $\Phi_e^\xi(x)\downarrow$ for some node ξ below τ, we set $\Psi(x) = \Phi_e^\xi(x)$.

If at any point of time we see a z such that $\Psi(z)$ is already defined, having copied some computation $\Phi_e^{\xi_0}(z)$, and some ξ_1 below τ such that $\Phi_e^{\xi_1}(z)\downarrow \neq \Psi(z)$, then ξ_0 and ξ_1 immediately become children of τ (if such a z is never found, the requirement is satisfied by fiat). Then, the τ-strategy now simply waits for R_e to be defined on an initial segment of $\mathbb{N}$ longer than both ξ_0 and ξ_1, then finds i such that $\xi_i \perp R_e$ and let the children of ξ_i become the children of τ.

As usual, the only possible real conflict between strategies arises when a node τ follows an $\mathcal{L}_{e,i}$-strategy, building its functional Ψ by copying some computations of type $\Phi_e^\xi(x)$ (with ξ below τ and $x \in \mathbb{N}$), but at some point a node τ' below τ asks to thin out the f-tree and thus to remove from the f-tree some nodes ξ below τ whose computations have been copied already. For two strategies, this conflict is solved in a similar way as in the proof of Theorem 3.1. Instead of allowing τ' to perform its action immediately, τ asks for confirmation, i.e., waits to see $R_e(x)$ being defined on all x currently in the domain of Ψ. Meanwhile, τ and τ' treat the set V of nodes that τ' wants to remove as already dead, that is, the nodes from V are ignored by Ψ when it tries to copy some Φ_e-computations and the ξ_0 and ξ_1 that τ' wants to pick as children are now in charge of the requirement just below the τ'-requirement in order of priority. Additionally, every minimal element ζ of V (i.e., an element of V that has no ancestor in V) is asked to satisfy (on top of its own requirement) the same requirement as τ' using its own strategy. Note that since τ treats any such ζ as dead, it will not interfere with ζ's strategy.

Like before, one of three things can happen:

- The confirmation never comes, meaning in particular that R_e is partial. This ensures that τ satisfies its requirement.
- For some x in the domain of Ψ, $R_e(x)$ becomes defined but $R_e(x) \neq \Psi(x)$. Since $\Psi(x)$ was obtained by copying a computation $\Phi_e^\xi(x)$ for some ξ below x, τ ensures the satisfaction of its requirement by making the children of ξ become its children.
- The confirmation comes and all values $\Psi(x)$ we were waiting confirmation on are indeed equal to $R_e(x)$. In this case, τ permits τ' to thin out the tree in the way τ' asked for, which it immediately does.

So in the three above cases, τ gets to satisfy its $\mathcal{L}_{e,i}$-requirement. Let us now look at the τ'-requirement. The second case corresponds to an injury, so what happens to τ' does not matter since some new node will be asked to satisfy what was previously the τ'-requirement. In the third case, τ' gets to perform the action it wants, so all is well. Let us examine the first case, where the confirmation never comes. In this case, τ' is never allowed to perform its action. However, below τ' we have two types of nodes: ξ_0, ξ_1, the nodes that would have been the eventual children of τ' if τ' had been allowed to carry on with its strategy, and the other nodes, namely the set V. Below ξ_0 and ξ_1 the τ'-requirement is satisfied (precisely because in both $\mathcal{P}$-strategies and $\mathcal{L}$-strategies, if the node that carries out the strategy asks to pick new children at least once, then its final children force the satisfaction of the requirement (i.e., all their extensions, in the f-tree or not, satisfy the requirement) and nodes in V will also satisfy the τ'-requirement they are asked to satisfy, because they can freely follow their strategy without interference from τ.

Thus, if we add this conflict resolution method between $\mathcal{L}$-strategies and strategies of lower priority, it is straightforward to see that, by finite injury, the final f-tree $T_\infty = \lim_s T_s$ (which is well defined because a node can only change its children finitely many times) is such that its paths form a perfect Π_1^0-class, and satisfy all requirements. □

The low basis theorem asserts that every Π_1^0 class contains a member of low Turing degree, thus we get Theorem 3.3 as an immediate corollary.

Finally, there is one last notion of size for subsets of $\{0,1\}^\omega$ that is dear to computability theorists, namely, a set is 'large' if it contains a Turing upper cone and is 'small' if it disjoints from a Turing upper cone. Martin's Turing determinacy theorem tells us that any Borel set which is closed under Turing equivalence must be either large or small on this account. The set LFS is indeed Borel (this is easy to see from the definition), but it is not closed under Turing equivalence, so Martin's theorem does not apply. In the next section, we will use a classical result from complexity theory to show that LFS is in fact disjoint from a Turing upper cone (Theorem 5.7).

5. Lowness for speed and Turing degrees

While lowness for speed is not closed under Turing equivalence, the following question is nonetheless interesting:

Which sets are Turing equivalent to some low for speed X? Which sets compute some non-computable low for speed X?

We denote by **LFS** and **LFS*** the set of Turing degrees that contain a low for speed set and a non-computable low for speed set, respectively. One of the main results of Bayer [3] is that not all degrees are in **LFS**. Indeed, there exists a c.e. degree $\mathbf{a} \notin \mathbf{LFS}$. The main question left open by Bayer regarding **LFS** is whether it is downward closed under $\leq_T$ or closed under join. We give a negative answer to both questions. To show that it is not downward closed, we need the following extension of Theorem 3.2 to degrees.

Theorem 5.1. *For any low c.e. degree* $\mathbf{a} > \mathbf{0}$*, we have* $\mathbf{a} \notin \mathbf{LFS}$*.*

Proof. We now suppose that A only is of c.e. degree (as opposed to being c.e.) and is low. Let (A_s) be a Δ^0_2 approximation of A, and let μ be its modulus of convergence, i.e., $\mu(n)$ is the smallest s such that all $\{A_t \mid t \geq s\}$ have the same prefix of length n (which must thus be the prefix of A of length n). Observe that μ is a lower semi-computable function, and let μ_s be its approximation at stage n (setting $\mu_0(n) = 0$ for all n). On the other hand μ cannot be computable because $A_{\mu(n)} \upharpoonright n = A \upharpoonright n$ for all n, which would make A computable. The fact that A has c.e. degree is equivalent to the fact that μ is A-computable (see [22, Theorem 3.6.5]), so let M be a functional such that $M^A(n) = \mu(n)$ for all n.

The rest of the proof is very similar to that of Theorem 3.2, so we only indicate which modifications are needed. This time, the function $S(e, .)$ we build is going to be an attempt to compute μ. The three steps for the $\mathcal{R}_e$-strategy are now:

Step I. Increase l by 1 and take for witness w_l the smallest integer which has not been picked as a witness by any strategy for any requirement.

Step II. Let s_l be the value of the first stage beyond the current stage at which we have $M^{A_{s_l}}(l) = \mu_{s_l}(l)$. Set $\Psi^{A_{s_l}\upharpoonright l}(w_l) = 0$, and let t_l be the time taken by this computation.

Step III. Simulate $\Phi_e(w_l)$ during $p_e(t_l)$ steps of computation, and distinguish three cases.

Case 1: $\Phi_e(w_l)$ returns 1 in $\leq p_e(t_l)$ steps. In this case, we set $R(w_l) = 0$ and $R(w_i) = 0$ for all other witnesses w_i currently used by our $(\mathcal{R}_e)$-strategy, further commit to having $\Psi^A(w_l) = 0$ even after potential future A-changes, mark this requirement as satisfied, stop the strategy for this requirement and move on to other requirements.

Case 2: $\Phi_e(w_l)$ returns 0 in $\leq p_e(t_l)$ steps. In this case, we do not define $R(w_l)$ yet, but check whether for all $i < l$, $S(e, i) = \mu_{s_l}(i)$.

(a) If it is not the case, let i be smallest such that $S(e, i) \neq \mu_{s_l}(i)$. This means that $A \upharpoonright i$ has changed between stages s_i and s_l. Thus, $A_{s_i} \upharpoonright i$ is not a true initial segment of A because we had $M^{A_{s_i}}(i) = \mu_{s_i}(i)$, and we know that $M^A(i) = \mu(i) \geq \mu_{s_l}(i) > \mu_{s_i}(i)$. Thus, even though we had set $\Psi^{A_{s_i}\upharpoonright i}(w_i) = 0$ (during Step II for i), we are free to set $\Psi^{A_{s_l}\upharpoonright i}(w_i) = 1$ (and commit to keeping $\Psi^A(w_i) = 1$), and then we can finally set $R(w_i) = 1 = \Psi^A(w_i) \neq \Phi_e(w_i)$. We also set $R(w_j) = 0$ for all other witnesses w_j currently used by our $(\mathcal{R}_e)$-strategy, mark requirement $(\mathcal{R}_e)$ as satisfied and terminate the strategy.
(b) If it is indeed the case, set $S(e, l) = A_{s_l} \upharpoonright l$ and then go back to Step I.

Case 3: $\Phi_e(w_l)$ is still undefined after $p_e(t_l)$ steps. In this case, we set $\Theta^{A_{s_l}\upharpoonright l}(e) \downarrow$ (which should be interpreted as signalling that we have entered Case 3). Observe that if $A_{s_l} \upharpoonright l$ is a true prefix of A, this implies $\Theta^A(e) \downarrow$ and therefore we would have $\lim_t h(e, t) = 1$. We again distinguish two subcases.

(a) The current value $h(e, s_l)$ is 0. Then we wait for a stage $t > s$ such that either $h(e, t) = 1$ or $A_t \upharpoonright 1 \neq A_{s_l} \upharpoonright l$ (one of the two must happen as we explained above). If the former happens first we move to subcase (b) below. If the latter happens first, we go back to Step II.
(b) The current value $h(e, s_l)$ is 1. We then set $R(w_l) = 0$ and commit to $\Psi^A(w_l) = 0$ and further set $R(w_i) = 0$ for all other witnesses, and commit to $\Psi^A(w_i) = 0$ as well. Then we wait – possibly forever – for a later stage $t > s_l$ where $h(e, t) = 0$ *and* $A_t \upharpoonright l \neq A_{s_l} \upharpoonright l$ (moving on to other requirements while waiting). If this happens, we come back to our $(\mathcal{R}_e)$-strategy where we left it and go back to Step I.

The verification is the same. □

Corollary 5.2. LFS *is not downward closed under* $\leq_T$*, even within c.e. degrees.*

Proof. Let $\mathbf{a} > \mathbf{0}$ be a c.e. degree in **LFS** whose existence was explained in Section 3. By Sacks's splitting theorem [20], there is a low c.e. degree $\mathbf{0} < \mathbf{b} < \mathbf{a}$. By Theorem 5.1, $\mathbf{b} \notin \mathbf{LFS}$. □

The next theorem will show that while not every c.e. degree contains a low for speed member, every non-zero c.e. degree **a** *bounds* a degree $\mathbf{b} \in \mathbf{LFS}$. Recall Bayer's result that whether 2-generics are low for speed or not depends on the 'P vs NP' question. When it comes to the *degree* of generics, we have that every 1-generic is Turing-equivalent to a set that is low for speed, independently of complexity-theoretic assumptions.

Theorem 5.3. *Every 1-generic degree* **g** *belongs to* $\mathbf{LFS}^*$.

Proof. We get this result by refining the proof of Theorem 2.1. In that proof, we built an X low for speed by finite extension, and ensuring that X was a subset of $S = \{0^{2^n} \mid n \in \mathbb{N}\}$. For $G \subseteq \mathbb{N}$, let $S_G = \{0^{2^n} \mid n \in G\}$. We claim that when G is 1-generic, $S_G = \{0^{2^n} \mid n \in G\}$ is low for speed (and clearly $S_G \equiv_T G$). In the proof of Theorem 2.1, if we let $\mathcal{U}_{e,i}$ be the effectively open set of those Z such that for some n, $\Phi_e^{S_Z}(n)$ and $R_i(n)$ both converge to different values, we know that G, being 1-generic, is either in $\mathcal{U}_{e,i}$ (hence satisfying the requirement $\mathcal{R}_{e,i}$ as per case (a)), or in the interior of the complement of $\mathcal{U}_{e,i}$, which precisely corresponds to case (b), hence the requirement is also satisfied in this case. □

We can derive a number of useful corollaries from this theorem. First of all, we get a new proof that LFS has the size of the continuum since $G \mapsto S_G$ is one-to-one, and there are continuum many 1-generic G. We also get another proof of the existence of a set of low degree that is low for speed (Theorem 3.3). Indeed, take a Δ^0_2 1-generic G; the corresponding set S_G is low for speed and is low as it is both Δ^0_2 and of GL_1 (Indeed a result of Jockusch [11] states that every 1-generic degree G is GL_1, that is, $G' \equiv_T G \oplus \emptyset'$; when $G \leq_T \emptyset'$, this is equivalent to $G' \equiv_T \emptyset'$).

A similar idea allows us to show that **LFS** contains a non-trivial interval in the Turing degrees.

Corollary 5.4. *There is a degree* $\mathbf{a} > \mathbf{0}$ *such that every* $\mathbf{0} \leq \mathbf{b} \leq \mathbf{a}$ *is in* **LFS**.

Proof. By a result of Haught [10], if **a** is a Δ^0_2 1-generic degree, every $\mathbf{b} > \mathbf{0}$ below **a** is of 1-generic degree. Then the result follows immediately from Theorem 5.3. □

Another interesting corollary is that every non-computable c.e. set bounds a non-computable low for speed set. Likewise almost every set, in the measure-theoretic sense, *bounds* a non-computable low for speed set.

Corollary 5.5. *Every non-zero c.e. degree bounds a member* $\mathbf{LFS}^*$, *every* 2*-random degree bounds a member of* $\mathbf{LFS}^*$.

Proof. This is simply because every non-zero c.e. degree and every 2-random degree bounds a 1-generic degree [17,13]. □

After generic degrees, let us move to random degrees. We have seen in Theorem 4.1 that LFS is a nullset, and in fact no Schnorr random is low for speed. Interestingly, this result does not extend to Turing degrees: by a result of Nies et al. [19], every high degree has a Schnorr random member and we have seen that there is a low for speed of high degree. This leaves open the possibility that almost all X are *Turing-equivalent* to a low for speed set. This would be similar to the category situation where – under the reasonable assumption $\mathsf{P} \neq \mathsf{NP}$ – the set LFS is meager (as proven by Bayer and Slaman) but the set of A's whose *degree* is in **LFS** is co-meager (Theorem 5.3). This is not the case however: if we increase the algorithmic randomness level from Schnorr randomness to Martin-Löf randomness, then the distinction disappears.

Proposition 5.6. *If* **a** *is, or simply bounds, a Martin-Löf random degree then* $\mathbf{a} \notin \mathbf{LFS}$ *(equivalently: if* $A \in \{0,1\}^\omega$ *computes a Martin-Löf random,* A *is not low for speed).*

Note that this shows in particular that $\{A \mid deg(A) \in \mathbf{LFS}\}$ has measure 0, and also that any $A \geq_T \emptyset'$ is not low for speed, as Chaitin's Ω number is Martin-Löf random and Turing equivalent to $\emptyset'$.

Instead of proving Proposition 5.6 directly, we will prove a stronger theorem for DNC degrees. We recall that a diagonally-non-computable (DNC) function is a total function $f : \mathbb{N} \to \mathbb{N}$ such that $f(e) \neq \varphi_e(e)$ whenever $\varphi_e(e)$ is defined, where $(\varphi_e)_e$ is an effective enumeration of partial computable functions from $\mathbb{N}$ to $\mathbb{N}$; a DNC degree is a degree which contains – or, equivalently, bounds – a DNC function.

Theorem 5.7. *If* **a** *is a DNC degree, then* $\mathbf{a} \notin \mathbf{LFS}$ *(equivalently: if* $A \in \{0,1\}^\omega$ *computes a DNC function,* A *is not low for speed).*

Proposition 5.6 follows from this theorem because DNC degrees are closed upwards, and by a result of Kučera [16], every Martin-Löf random degree is a DNC degree.

Proof. The proof of this theorem relies on the proof of a classical computational complexity theorem, namely Blum's speed-up theorem [6] (see also [15, Theorem 32.2]), which asserts that for every sufficiently fast-growing computable function f,

there exists a computable set R which admits no fastest algorithm in that for every i such that $\Phi_i = R$, there is a j such that $\Phi_j = R$ and $f(time(\Phi_j, x)) \leq time(\Phi_i, x)$ for almost every x.

Let us first discuss how to prove Blum's speed-up theorem. As it happens, we have already seen a proof with similar features, namely the proof of Theorem 4.1 (this is no coincidence, as it is the proof Blum's speed-up theorem that inspired this other proof). We build R by diagonalization against all Φ_i, where for all x in order we try to find an active $i \leq |x|$ such that $\Phi_i(x)$ converges in less than $f^{|x|-i}(|x|)$ steps (here the exponent is understood as composition: f^0 is the identity, and $f^{n+1} = f \circ f^n$) and if such an i is found, we diagonalize against Φ_i by setting $R(x) = 1 - \Phi_i(x)$ for the smallest such i, and declare i inactive from that point on. If no such i is found, set $R(x) = 0$. Obviously R is computable, and the same argument as in the proof of Theorem 4.1 shows that for any Φ_i computing R, one must have $time(\Phi_i, x) \geq f^{|x|-i}(|x|)$ for almost all x. Now, suppose Φ_e is a functional that computes R. We need to show that there is another functional which computes R much faster than Φ_e. Fix a large k and assume we are given as 'advice' the finite list σ_k of indices $i < k$ such that Φ_i eventually gets diagonalized against (and therefore i becomes inactive) in the construction of R. Now, we can compute R via the following procedure. In a first phase, simply follow the construction of R as described above, until we reach a point where all $i \in \sigma_k$ have become inactive. At this point, we know (only because we know σ_k!) that none of the $\{\Phi_i \mid i < k\}$ are relevant for the construction of R on future x. Thus, we enter a second phase where to compute each $R(x)$, we only need to simulate, for $k \leq j \leq |x|$, $\Phi_j(x)$ during $f^{|x|-j}(|x|)$ steps of computation. By dovetailing, this can be done in $poly(|x| \cdot f^{|x|-k}(|x|))$ (the polynomial being independent of k) which, if f is fast growing enough and k large enough compared to e, is $< f\big(f^{|x|-e}(|x|)\big)$, which in turn is $< f(time(\Phi_e, x))$ for almost all x (note that such a k can be computed uniformly given e), and this finishes the proof of Blum's speed-up theorem.

Looking more closely, the core of the argument is that for a given x and a given i, if we somehow knew that Φ_i is not diagonalized against*exactly* at that point x, then we can save the computation of $\Phi_i(x)[f^{|x|-i}(|x|)]$ in the computation of $R(x)$. We already see why DNC-ness naturally comes into play. Let $\theta : \mathbb{N} \to \{0,1\}^*$ be the partial computable function such that $\theta(i)$ is the one string x on which Φ_i is diagonalized against during the construction of R, and $\theta(i) \uparrow$ if there is no such x. Having a DNC function as oracle allows us to find, for any given i, a $y \neq \theta(i)$ should $\theta(i)$ be defined, and thus one can speed up a bit the computation of $R(y)$ for this y. In fact we can do much better: by a result of Jockusch [12], having access to a DNC function allows us to uniformly avoid a finite number of values of a given partial recursive function. Here this means that with our DNC oracle, we can, uniformly in k, find a $y \neq \theta(i)$ for any $i < k$ on which θ is defined, or equivalently, a y such that none of the Φ_i with $i < k$ gets diagonalized at y. Applying Blum's argument, one can indeed use this information to truly speed up the computation of $R(y)$. We are not quite done yet however: our argument shows that once we have computed y from k we can speed-up the computation of $R(y)$, but the computation of y itself might take a long time and offset the time we save on the computation of $R(y)$.

To overcome this problem, we need to refine the above idea. Let $\langle .,.,. \rangle$ be the canonical 'tripling' function: $\langle a, b, c\rangle = \langle a, \langle b, c\rangle\rangle$ and for $1 \leq i \leq 3$, let π_3^i be the i-th projection ($\pi_3^i(\langle x_1, x_2, x_3\rangle) = x_i$, note that these functions are polynomial-time computable). Since A has DNC degree, again using Jockusch's result, A computes, via a fixed functional Ξ, a function F such that $F(k) \neq \pi_3^2(\theta(i))$ for any $i < k$ on which θ is defined (here and in the rest of the proof we identify strings and integers). Said otherwise, for *any* pair (a, c) of integers, none of the functionals $\{\Phi_i \mid i < k\}$ gets diagonalized against at $y = \langle a, F(k), c\rangle$.

Now let Ψ be the functional which computes R as follows using oracle A. On input x, it first computes the projections of x, i.e., finds k, l, m such that $x = \langle k, l, m\rangle$. Then, it tries to compute $F(k)$ via Ξ and using oracle A during m steps of computation. If it fails to do so, it then computes $R(x)$ using a fixed procedure (with no access to the oracle) and returns this value. If it does succeed to compute $F(k)$, it then checks whether $l = F(k)$. If not, $\Psi^A(x)$ again returns $R(x)$ using a fixed procedure to perform the computation, without access to the oracle. Finally, and this is the interesting case, if $F(k)$ is computed in $\leq m$ steps of computation, and $l = F(k)$, we know by definition of F that none of the $\{\Phi_i \mid i < k\}$ gets diagonalized against at x. Thus Ψ^A can compute $R(x)$ by only simulating, like in the proof of Blum's theorem $\Phi_j(x)$ during $f^{|x|-j}(|x|)$ steps of computation for $k \leq j \leq |x|$. In that case, the total computation time is still $poly(|x| \cdot f^{|x|-k}(|x|))$ because the m steps of computation needed to get $F(k)$ are simply $poly(|x|)$. For a given k, taking m sufficiently large will give enough time for the computation of $F(k)$ by A, and make $x = \langle k, F(k), m\rangle$ large enough to ensure that the computation time of $\Psi^A(x)$ is $< f(time(\Phi_e, x))$, as long as f is sufficiently fast growing. This shows that A is not low for speed. □

As an immediate corollary, we get the result mentioned in the previous section that **LFS** is disjoint from a cone in the Turing degrees, again because DNC degrees are closed upwards. Another interesting consequence is that the analogue of the low basis theorem for Π_1^0 class does not extend to lowness for speed: it is not true that every non-empty Π_1^0 class contains a member of low for speed degree. For example, one can take a Π_1^0 class of Martin-Löf randoms: all its elements have DNC degree by Kučera's theorem, and thus do not have low for speed degree.

Another corollary is that **LFS** is not closed under join.

Theorem 5.8. *There are* $\mathbf{a}, \mathbf{b} \in$ **LFS** *such that* $\mathbf{a} \vee \mathbf{b} \notin$ **LFS**.

Proof. Let G_0 be 2-generic, i.e., 1-generic relative to $\emptyset'$. Consider $G_1 = G_0 \Delta \emptyset'$ where Δ is the symmetric difference. It is easy to check that G_1 is also 2-generic. Thus S_{G_0} and S_{G_1} (defined as in the proof of Theorem 5.3) are both low for speed (again from the proof of Theorem 5.3) but $S_{G_0} \oplus S_{G_1} \geq_T G_0 \oplus G_1 \geq_T G_0 \Delta G_1 = \emptyset'$. Since $\emptyset'$ has DNC degree (this is obvious from the definition of DNC function), it follows from Theorem 5.7 that $deg(S_{G_0} \oplus S_{G_1}) \notin$ **LFS**. □

At this point, we know that the set of X's which *compute* a member of LFS* is very large: it has measure 1 and is co-meager, it contains every c.e. set, etc. We might even start thinking that *every* non-computable X computes a member of LFS*. This is not the case however, as shown by the following theorem (which contrasts Corollary 5.4).

Theorem 5.9. *There is a degree* $\mathbf{a} > \mathbf{0}$ *such that no* $\mathbf{0} < \mathbf{b} \leq \mathbf{a}$ *is in* **LFS***. *Indeed,* $\mathbf{a}$ *can be chosen to be a minimal Turing degree.*

Proof. Kumabe and Lewis [14] proved that there exists a minimal degree DNC degree. By Theorem 5.7, such a degree is not low for speed, and by minimality there is no $\mathbf{b}$ such that $\mathbf{0} < \mathbf{b} < \mathbf{a}$. □

Note that another way to obtain a degree $\mathbf{a} > \mathbf{0}$ such that no $\mathbf{0} < \mathbf{b} \leq \mathbf{a}$ is in **LFS*** is by taking $\mathbf{a}$ to be a hyperimmune-free degree containing a Martin-Löf random member, which ensures that every $\mathbf{0} < \mathbf{b} \leq \mathbf{a}$ also computes a 1-random (see [9, Corollary 8.6.2]), and apply Proposition 5.6.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This paper grew from interactions between complexity theory and classical computability theory originating from the 2012 Dagstuhl Seminar "Computability, Complexity and Randomness" (Seminar 12012). Bienvenu acknowledges support of the Agence Nationale de la Recherche (ANR-15-CE40-0016-01 RaCAF grant), Downey thanks the Marsden Fund of New Zealand (grant 16-VUW-131), and the LIRMM (University of Montpellier) where this research was undertaken. The authors would like to thank the anonymous referees for their insightful comments and suggestions.

References

[1] Eric Allender, Harry Buhrman, Michal Koucký, What can be efficiently reduced to the Kolmogorov-random strings?, Ann. Pure Appl. Logic 138 (2006) 2–19.
[2] Eric Allender, Luke Friedman, William I. Gasarch, Limits on the computational power of random strings, Inf. Comput. 222 (2013) 80–92.
[3] Robertson Bayer, Lowness for Computational Speed, PhD thesis, University of California Berkeley, 2012.
[4] Theodore Baker, John Gill, Robert Solovay, Relativizations of the $\mathcal{P} =? \mathcal{NP}$ question, SIAM J. Comput. 4 (4) (1975) 431–442.
[5] Laurent Bienvenu, Game-Theoretic Characterizations of Randomness: Unpredictability and Stochasticity, PhD thesis, Université de Provence, 2008, https://tel.archives-ouvertes.fr/tel-00332425v2.
[6] Manuel Blum, On effective procedures for speeding up algorithms, J. ACM 18 (290–305) (1971).
[7] Mingzhong Cai, Rodney Downey, Rachel Epstein, Steffen Lempp, Joseph Miller, Random strings and tt-degrees of Turing complete c.e. sets, Log. Methods Comput. Sci. 10 (3) (2014).
[8] S. Barry Cooper, A jump class of noncappable degrees, J. Symb. Log. 324 (353) (1989).
[9] Rodney Downey, Denis Hirschfeldt, Algorithmic Randomness and Complexity, Theory and Applications of Computability, Springer, 2010.
[10] Christine A. Haught, The degrees below a 1-generic degree $< 0'$, J. Symb. Log. 51 (1986).
[11] Carl Jockusch, Degrees of generic sets, in: Frank Drake, Stanley S. Wainer (Eds.), Recursion Theory: Its Generalizations and Applications, in: London Mathematical Society Lecture Note Series, vol. 45, Cambridge University Press, 1980, pp. 110–139.
[12] Carl Jockusch, Degrees of Functions with No Fixed Points, North-Holland, Amsterdam, 1989, pp. 191–201.
[13] Steven M. Kautz, Degrees of Random Sets, PhD thesis, Cornell University, 1991.
[14] Masahiro Kumabe, Andrew E.M. Lewis, A fixed-point-free minimal degree, J. Lond. Math. Soc. 80 (3) (2009) 785–797.
[15] Dexter Kozen, Theory of Computation, Springer, New York, 2006.
[16] Antonin Kučera, Measure, Π^0_1 classes, and complete extensions of PA, Lect. Notes Math. 1141 (1985) 245–259.
[17] Stuart Kurtz, Randomness and Genericity in the Degrees of Unsolvability, PhD dissertation, University of Illinois at Urbana, 1981.
[18] André Nies, Computability and Randomness, Oxford Logic Guides, Oxford University Press, 2009.
[19] André Nies, Frank Stephan, Sebastiaan Terwijn, Randomness, relativization and Turing degrees, J. Symb. Log. 70 (2005) 515–535.
[20] Gerald Sacks, On the degrees less than 0', Ann. Math. 77 (1963) 211–231.
[21] Richard A. Shore, A non-inversion theorem for the jump operator, Ann. Pure Appl. Logic 40 (1988) 277–303.
[22] Robert Soare, Turing Computability: Theory and Applications, Theory and Applications of Computability, Springer, 2016.