What Percentage of Programs Halt?

Laurent Bienvenu[1(✉)], Damien Desfontaines[2], and Alexander Shen[3,4]

[1] LIAFA - CNRS and Université Paris 7, Paris, France
laurent.bienvenu@liafa.univ-paris-diderot.fr
[2] Google Inc., Zurich, Switzerland
damien@desfontain.es
[3] LIRMM - CNRS and Université Montpellier, Montpellier, France
[4] On leave from IITP RAS, Moscow, Russia
alexander.shen@lirmm.fr

Abstract. Fix an optimal Turing machine U and for each n consider the ratio ρ_n^U of the number of halting programs of length at most n by the total number of such programs. Does this quantity have a limit value? In this paper, we show that it is not the case, and further characterise the reals which can be the limsup of such a sequence ρ_n^U. We also study, for a given optimal machine U, how hard it is to approximate the domain of U from the point of view of coarse and generic computability.

1 Introduction

1.1 Motivation

The title of this paper, 'What percentage of programs halt?' is intentionally provocative; obviously, the answer depends on the programming language. To make this question reasonable, we need to put some restrictions on the programming language (=interpreter). Following the theory of algorithmic information, we consider "optimal programming languages". That is, we consider an optimal Turing machine U (see below for the exact definition) and look, for each n, at the fraction ρ_n^U of inputs of length at most n on which U halts (among all inputs of those lengths). It is well known that the sequence ρ_n^U is not computable (knowing the exact values of ρ_n^U, one can solve the halting problem). What else can be said about it? For example, can ρ_n^U converge to some limit? As we will see, this cannot happen (Theorem 4). What can then be said about the limit points of ρ_n^U? They are Martin-Löf random numbers, even relative to $\mathbf{0}'$ (Theorem 5). What are the possible values of $\limsup \rho_n^U$? All $\mathbf{0}'$-lower semicomputable $\mathbf{0}'$-random numbers (Theorem 6; for $\liminf$ similar question remains open).

In the second part of the paper we build on these results to study a related question: can we somehow approximate the domain of U? That is, can we find an algorithm that tells us whether $U(p)$ terminates or not, giving the correct answer for most inputs p? This question may be formalized in different ways. For most of them, the answer will not depend on the particular choice of optimal machine, with the notable exception of Theorem 15.

M.M. Halldórsson et al. (Eds.): ICALP 2015, Part I, LNCS 9134, pp. 219–230, 2015.
DOI: 10.1007/978-3-662-47672-7_18

All these questions are quite natural and similar results appeared in different settings. In 1974 Nancy Lynch [11] considered similar questions for a more restricted class of machines that are optimal in some effective sense, as defined by Schnorr [15]. Later the question for some specific universal machine was studied by Hamkins and Miasnikov who showed [7] that the halting problem can be approximated in this case. They considered Turing machines with one-sided tape. (Their result implies that corresponding universal machine is not optimal and thus not effectively optimal.) The criterion for the domains of optimal machines (a set is a domain for some optimal machine if and only if it is a computably enumerable set such that the complexity of the number of strings of length at most n in this set is $n - O(1)$) was obtained by Calude, Nies, Staiger, and Stephan [4]. The recursion-theoretic properties of different versions of approximate computability have been studied by Downey, Jockusch, and Schupp [6]. See also Antti Valmari [16] who provides a survey of some other results, including the ones from [14] and [8]. Our goal in this paper is to provide a unified approach that allows us to give simple proofs of known results (sometimes in a more general form) and establish some new ones.

1.2 Definitions and Notation

For a set E, by χ_E we denote the characteristic function of this set. If E is finite, $|E|$ denotes its cardinality. We write 'log' for base 2 logarithms.

We denote by $\{0,1\}^*$ the set of all (finite) binary strings, by $\{0,1\}^n$ the set of strings of length n and by $\{0,1\}^{\leqslant n}$ the set of strings of length at most n. The length of a string x is denoted by $|x|$. We denote by $\{0,1\}^\omega$ the set of infinite binary sequences. They are also identified with real numbers in $[0,1]$ in binary notation; we mention the cases when the non-uniqueness (the same number has two representations) creates problems.

For a partial computable function f, the domain of f, denoted by $\mathrm{dom}(f)$, is the set of inputs on which f halts. A *machine* is a partial computable function from $\{0,1\}^*$ to $\{0,1\}^*$. An input p of a machine M is sometimes referred to as a *program*, and if $M(p) = x$, we say that p is a *description of* x (relative to M), or that x is the *output* of program p.

By C and K we respectively denote the plain and prefix-free versions of Kolmogorov complexity. We assume that the reader has some background in computability theory, Kolmogorov complexity and algorithmic randomness (see, e.g., [5,10,13,17]).

Definition 1. *A machine U is said to be* optimal *if for every machine M there is a constant c_M such that whenever $M(p) = x$, there is a q such that $|q| \leqslant |p| + c_M$ and $U(q) = x$.*

This definition is used to define plain Kolmogorov complexity: if U is optimal, then $C_U(x) = \min\{|q| \colon U(q) = x\}$ is the plain Kolmogorov complexity function (defined up to $O(1)$ additive term).

In the rest of this paper, we assume that U is a fixed optimal Turing machine. Let H_n be the number of programs of length at most n on which U halts, and

let ρ_n^U be the *fraction* of programs of length at most n on which U halts among all programs of lengths at most n. For simplicity, we define $\rho_n^U = H_n/2^{n+1}$, even though technically there are only $2^{n+1}-1$ programs of length at most n. Since we are only concerned in the asymptotic behaviour, this does not matter (to make things completely formal we could also add an extra program of length 0).

2 Counting How Many Programs Halt

2.1 The Complexity of H_n

The following easy lemma is well-known (see for example [17]).

Lemma 2. *For all n, $C(H_n|n) = C(H_n) = n$ with $O(1)$-precision.*

Proof. Indeed, $C(H_n|n) \leqslant C(H_n) \leqslant n$ with $O(1)$-precision since $H_n \leqslant 2^{n+1}$. Conversely, if we have a program q that maps n to H_n and is d bits shorter than n, we may take $O(\log d)$-bit self-delimiting description of d and append q; the resulting string allows us to reconstruct d, then q, then $n = |q| + d$, then $H_n = q(n)$. Then we find all strings of length at most n where U is defined, and a string z that has no description of size at most n. This gives $C(z) > n$ and $C(z) \leqslant O(\log d) + n - d + O(1)$ at the same time, so $d = O(1)$. □

The next lemma extends this result to *approximations* for H_n.

Lemma 3. *Let N be an integer such that $|N - H_n| \leqslant 2^k$. Then $C(N|n) \geqslant n - k - K(k|n) - O(1)$ and $K(N|n) \geqslant n - k - O(1)$.*

Proof. Let N be an approximation of H_n with error at most 2^k, and let t be the program of length $C(N|n)$ that maps n to N. We can reconstruct H_n given n, t and the difference $N - H_n$ (first we reconstruct N and then H_n). So $C(H_n|n) \leqslant C(t, N - H_n|n)$. The pair $(t, N - H_n)$ can be described by appending t to the self-delimiting description of $N - H_n$ (the latter requires $k + K(k|n)$ bits), or we could use self-delimiting program t and append plain description of $N - H_n$ (the latter requires $k + O(1)$ bits).[1] This gives respectively $n \leqslant C(N|n) + k + K(k|n) + O(1)$ and $n \leqslant K(N|n) + k + O(1)$. □

2.2 Limit Points of ρ_n^U

Lemma 3 can be used to get some information about ρ_n^U. Assume that r_n is some computable sequence of rational numbers. How close can it approximate ρ_n^U? The complexity $K(r_n|n)$ is $O(1)$, so Lemma 3 gives a constant upper bound for $n - k$, so $k = n - O(1)$, which means an absolute error for H_n of size at least $\Omega(2^n)$, so $|\rho_n - r_n|$ is separated from 0 for sufficiently large n.

In particular, taking $r_n = 0$ or $r_n = 1$, we see that $\varepsilon \leqslant \rho_n^U \leqslant 1 - \varepsilon$ for some $\varepsilon > 0$ and for all sufficiently large n.

[1] In other words, we use the inequality $C(u, v) \leqslant K(u) + C(v)$ in two different ways.

We will use this lemma to show that ρ_n^U has no limit. Note first that it is very easy to construct a particular optimal machine V such that ρ_n^V does not converge. (For example, it is easy to construct an optimal machine V defined only on inputs of even length, then ρ_{2n}^V and ρ_{2n+1}^V differ by factor 2: the numerator is the same and the denominators differ by factor 2.) The next theorem shows that ρ_n^U *never* converges, no matter which optimal machine we choose.

Theorem 4. *The sequence* $(\rho_n^U)_{n\in\mathbb{N}}$ *does not converge.*

Proof. Consider a computable sequence r_n that is everywhere dense in $[0,1]$ (say, enumerates all rational numbers in $[0,1]$). If ρ_n^U has some limit ρ, then ρ_n^U is close to ρ for *all sufficiently large* n while r_n is close to ρ for *infinitely many* n, so the difference $r_n - \rho_n^U$ cannot be separated from 0. □

Now that we know that the sequence ρ_n^U does not converge, one can study its limit points. The next theorem shows that any limit point of the sequence must be quite complex, indeed, Martin-Löf random relative to $\mathbf{0}'$.

Theorem 5. *All limit points of* $(\rho_n^U)_{n\in\mathbb{N}}$ *are Martin-Löf random relative to* $\mathbf{0}'$.

Proof. For this proof we need to use a theorem by Miller [12] (see also [1] for a simple proof): a real number (a bit sequence) $x \in [0,1]$ is Martin-Löf random relative to $\mathbf{0}'$ if and only if there is a constant c such that for every prefix σ of x, there is a finite string τ extending σ such that $C(\tau) \geqslant |\tau| - c$.

Suppose x is a limit point of ρ_n^U. First note that x cannot be a rational number (otherwise the constant sequence $r_n = r$ approximates ρ_n^U), so x has a unique binary representation. Let σ be a prefix of x and let k be the length of σ. Split $[0,1]$ into 2^k equal intervals of size 2^{-k}. Then x is strictly inside one of these intervals (this interval consists of all binary extensions of σ). Since x is a limit point, some ρ_n^U also belongs to this interval. Recall that ρ_n^U is a binary fraction $H_n/2^{n+1}$ (here it is important that we use this denominator, not $2^{n+1}-1$; of course, this does not change the limit points). Therefore, H_n (considered as a string of length $n+1$ with leading zeros) is an extension of σ, and $C(H_n) \geqslant |H_n| - O(1)$ due to Lemma 2, so it remains to use Miller's result. □

We do not know whether the converse holds, i.e., whether any real that is Martin-Löf random relative to $\mathbf{0}'$ is a limit point of some sequence ρ_n^U for some optimal U. However, we can give a full characterisation of the reals that are lim sup's of those sequences.

Theorem 6. *The* lim sup *of* $(\rho_n^U)_{n\in\mathbb{N}}$ *is upper semicomputable relative to* $\mathbf{0}'$ (*and Martin-Löf random relative to* $\mathbf{0}'$ *by the previous theorem*). *Moreover, the converse holds: every real in* $[0,1]$ *that is upper semicomputable relatively to* $\mathbf{0}'$ *and Martin-Löf random relative to* $\mathbf{0}'$ *is the* lim sup *of* ρ_n^V *for some optimal machine* V.

Proof. Let us consider first a simpler question. Assume that X is an arbitrary computably enumerable set, i.e., the domain of some machine, not necessarily an optimal one; x_n is the number of strings of length n in X, and X_n is the number of strings of length at most n in X (so $X_n = x_0 + \ldots + x_n$). Consider the upper density of X, i.e., $\limsup X_n/2^{n+1}$. Which reals can appear as upper densities of computably enumerable sets?

Lemma 7. *A real number x in $[0,1]$ is the upper density of some computably enumerable set X if and only if x is upper semicomputable relative to $\mathbf{0}'$.*

Proof. In one direction: $X_n/2^{n+1}$ is a uniformly lower semicomputable sequence of reals, and one can show (see, e.g., [6]) that lim sup of such a sequence is upper semicomputable relative to $\mathbf{0}'$.

Reverse direction: assume that x is upper semicomputable relative to $\mathbf{0}'$. It is known that x can be represented as $\limsup k_n$ for some computable sequence k_n of rational numbers (see [6] or [17]). Then $x = \lim_n K_n$, where $K_n = \sup(k_n, k_{n+1}, \ldots)$ form a uniformly lower semicomputable sequence. We may assume without loss of generality that $K_n \in [0,1]$ (since the limit is in $[0,1]$) and that K_n are rational numbers with denominator 2^n (by rounding; note that the resulting sequence K_n may not be computable, only lower semicomputable). Then we consider a computably enumerable set X that contains exactly K_n strings of length n (here we use that K_n are lower semicomputable). It is easy to see that the upper density of X is x; in fact, the density (the limit, not only lim sup) exists and is equal to x, since the fraction of n-bit strings in X converges to x as $n \to \infty$. □

It remains to show that for x that are not only upper semicomputable relative to $\mathbf{0}'$ but also Martin-Löf random relative to $\mathbf{0}'$, the set X can be made a domain of an optimal machine. Our next step is the following simple observation.

Lemma 8. *If some real $x \in [0,1]$ is the upper density of the domain of some optimal machine, the same is true for $x/2$ and $(1+x)/2$.*

(In terms of binary representation $x/2$ is $0x$, and $(1+x)/2$ is $1x$.)

Proof. For $x/2$ we just "shift" the domain of the optimal machine by adding leading 0 to all the arguments. For $(1+x)/2$ we do the same and also add all strings starting with 1 to the domain (with arbitrary values, e.g., they all can be mapped to an empty string). In both cases the machine remains optimal, the complexity increases only by 1. □

Deleting the first bit preserves randomness, so we may assume without loss of generality that x (that is random and upper semicomputable relative to $\mathbf{0}'$) is smaller than $1/2$ (starts with 0), and then apply Lemma 8 to add leading ones.

Now we are ready to use another known result: *every random upper semicomputable x is Solovay complete among upper semicomputable reals* (all properties are considered relative to $\mathbf{0}'$); according to one of the equivalent definitions of Solovay completeness, this means that for every other upper semicomputable

(relative to $\mathbf{0}'$) y and for large enough N there exists another upper semicomputable (relative to $\mathbf{0}'$) z such that $x = y/N + z$. This result combines the work of Calude et al. [3] and Kučera-Slaman [9] (see [2] for a simplified proof). Technically these papers consider lower semicomputable reals instead of upper semicomputable ones. However, an upper semicomputable real is just the opposite of a lower semicomputable real, randomness is stable under sign change, and Solovay reducibility, although often restricted to numbers in $[0, 1]$, extends naturally to all real numbers (again, see [2]), so the result also holds for upper semicomputable reals. Also, we need a relativized version of their result to $\mathbf{0}'$; as usual, relativization is straightforward.

So let us assume that $x \in (0, 1/2)$ and $x = y/2^d + z$ where y is the upper density for some optimal machine U and z is upper semicomputable relative to $\mathbf{0}'$. (The large denominator N is chosen to be a power of 2.) Now we combine two tricks used for Lemmas 7 and 8. Namely, we apply Lemma 7 to $2z$ (note that $z < 1/2$), and then add leading 1's to all the strings in the corresponding set. This gives us density z while using only right half of the binary tree (strings that start with 1). Then we add d zeros to all strings in the domain of U as we did when proving Lemma 8; this gives us density $y/2^d$ using only left half of the binary tree (actually, a small part of it, if d is large). Then we combine both parts and get an optimal machine (since the left part is optimal) with upper density $y/2^d + z$ as required. (Note that in general lim sup is not additive, but in our case we have not only lim sup, but limit in one of the parts, so additivity holds.) □

3 Approximating the Halting Problem

3.1 Generic and Coarse Computability

Instead of just counting the terminating programs of bounded length, one can also look at a related question: *is there an algorithm which, given p, predicts whether or not $p \in \mathrm{dom}(U)$, and is right "most of the time"*? This is a rather informal question; to make it formal we have to specify what we mean by 'predict', and 'most of the time'. There are several ways to do this, and two paradigms in particular have received a lot of attention in the recent literature, the so-called *coarse computability* and *generic computability*. For both of them, "being right most of the time" is understood as "being right on a set of density 1". (Recall that the upper density $\bar{\rho}(A)$ of a set $A \subseteq \{0,1\}^*$ is $\limsup_n |A \cap \{0,1\}^{\leqslant n}|/2^{n+1}$, the lower density $\underline{\rho}(A)$ is $\liminf_n |A \cap \{0,1\}^{\leqslant n}|/2^{n+1}$, and when the two are equal, their common value is called the density of A. Sometimes the density is defined for sets of natural numbers and all the initial segments are considered, not only powers of 2, but for density 1 this does not matter.)

The difference between coarse computability and generic computability lies in the prediction model. In coarse computability, the predictor is a *total* computable function which given an input $p \in \{0,1\}^*$ should always return 0 or 1 (meaning "$p \notin \mathrm{dom}(U)$" and "$p \in \mathrm{dom}(U)$" respectively), but is allowed to be incorrect sometimes, as long as the set of errors has density zero. In the generic

computability model, the predictor function is still 0/1-valued, but is allowed to be partial as long as its domain has density 1, *and whenever a* 0/1*-prediction is made, it must be correct*. Formally, we have the following definitions.

Definition 9. *A set* $A \subseteq \{0,1\}^*$ *is* coarsely computable *if there exists a total computable function* $f : \{0,1\}^* \to \{0,1\}$ *such that the set* $\{p \mid f(p) = \chi_A(p)\}$ *has density* 1. *A set* $A \subseteq \{0,1\}^*$ *is* generically computable *if there exists a partial computable function* $f : \{0,1\}^* \to \{0,1\}$ *such that* $\mathrm{dom}(f)$ *has density* 1 *and* $f(p) = \chi_A(p)$ *for all* $p \in \mathrm{dom}\, f$.

These two notions are incomparable: a computably enumerable set can be coarsely computable but not generically computable and vice-versa (see [6]). The initial informal question we started with can now be precisely formulated: if U is an optimal machine, can $\mathrm{dom}(U)$ be coarsely computable? generically computable? The answer is no, even if we allow the approximating function f to be both non-total and sometimes wrong — still requiring that it is correct for most inputs. Moreover, f has $\Omega(1)$ fraction of errors among strings of length at most n, *for all sufficiently large* n, not only for infinitely many n (as it is needed to show that f is not coarsely/generically computable). Similar results were obtained in a slightly different setting in [14]; we provide a simple argument that requires only optimality and covers both generic and coarse computability.

Theorem 10. *For every partial computable function* $f : \{0,1\}^* \to \{0,1\}$ *there exists some* $\varepsilon > 0$ *so that the fraction of strings* x *of size at most* n *where* f *is undefined or gives a wrong answer* $(f(x) \neq \chi_{\mathrm{dom}\, U}(x))$ *exceeds* ε *for all sufficiently large* n.

Proof. We repeat the proof of Lemma 3. Knowing n and some bound 2^{n-d} for the number of errors (of both types: f is undefined or the value is wrong) that f makes for strings of length at most n, we wait until f becomes defined on all strings of those lengths except for 2^{n-d} many. Then we count the number of positive answers; it differs from H_n by at most $O(2^{n-d})$. The difference can be specified by $n-d+O(1)$ bits, so the complexity $C(H_n|n)$ is bounded by $K(d|n)+(n-d)+O(1)$, where $O(1)$-constant depends on f. The bound $C(H_n|n) \geqslant n - O(1)$ then implies that $d - K(d|n) \leqslant O(1)$, so $d = O(1)$. This provides the required bound $2^{-O(1)}$ for the fraction of errors for all large enough n. □

3.2 Allowing a Small Density of Errors and 'Infinitely Often'-Success

The constant ε in Theorem 10 may depend on the predictor f. Can we prove a stronger result where the same ε is used for all predictors? It is indeed possible if we only want the predictor to have a lot of errors for *infinitely many* lengths, not for all sufficiently large ones. The result of this type was obtained in [8]; we provide a simple proof of its version for arbitrary optimal machines. More precisely, let us consider the following definition (which makes sense when α is close to 1).

Definition 11. *Let $\alpha \in [0,1]$. A set $A \subseteq \{0,1\}^*$ is* α-coarsely computable *if there exists a total computable function $f : \{0,1\}^* \to \{0,1\}$ such that the set $\{p \mid f(p) = \chi_A(p)\}$ has lower density at least α. A set $A \subseteq \{0,1\}^*$ is* α-generically computable *if there exists a partial computable function $f : \{0,1\}^* \to \{0,1\}$ such that $\mathrm{dom}(f)$ has lower density at least α and $f(p) = \chi_A(p)$ for all $p \in \mathrm{dom}(f)$.*

Although we saw that there was no implication between being generically computable and coarsely computable, there *is* such a link in the quantified setting. Namely, if a set is α-generically computable, it is β-coarsely computable for any $\beta < \alpha$. Indeed, consider some rational threshold r between β and α. We know that for infinitely many lengths the fraction of answers provided by generic predictor f, exceeds r. These lengths can be ultimately discovered (by waiting until the fraction exceeds r). Let us consider a fast growing computable sequence that contains only these lengths (not necessarily all of them). For lengths in this sequence we know r-fraction of correct answers and give arbitrary answers for the rest. (There is a small technical problem since these answers could be incompatible with the answers chosen previously, for smaller lengths. But if the lengths in the sequence grow fast enough, this small change is compensated by the difference between β and r.)

Theorem 12. *There exists $\alpha < 1$ such that $\mathrm{dom}(U)$ is neither α-coarsely computable nor α-generically computable.*

Proof. This result, like Theorem 10, remains true even if we allow errors of both types (as before), and the proof is similar. Proving Theorem 10, we noted that $C(H_n|n) \leqslant K(d|n) + (n-d) + O(1)$, if some (fixed) algorithm f has fraction of errors at most 2^{-d} on strings of length at most n. Here the constant in $O(1)$ depends on f. We also can treat f as a parameter; the same argument gives then $C(H_n|n) \leqslant K(d|n) + (n-d) + K(f|n) + O(1)$, where $K(f|n)$ is the prefix conditional complexity of an algorithm computing f, given n. Now $O(1)$ is the same for all f. It remains to note that for every computable f there are infinitely many n such that $K(f|n) = O(1)$, where the constant does not depend on f (or n). For example, we may consider n whose binary representation starts with self-delimited encoding of the program for f. The rest of proof remains the same, and we get the same ε for all f (but only for n that make f simple). □

The next natural question is whether we can combine both results and beat each predictor for all sufficiently large lengths, still using the same ε for all predictors. The following definition formalizes this question; we use dual notions where lower density is replaced by upper density (and "i.o." stands for "infinitely often").

Definition 13. *Let $\alpha \in [0,1]$. A set $A \subseteq \{0,1\}^*$ is* α-i.o.-coarsely computable *if there exists a total computable function $f : \{0,1\}^* \to \{0,1\}$ such that the set $\{p \mid f(p) = \chi_A(p)\}$ has an upper density of at least α. A set $A \subseteq \{0,1\}^*$ is* α-i.o.-generically computable *if there exists a partial computable function $f : \{0,1\}^* \to \{0,1\}$ such that $\mathrm{dom}(f)$ has an upper density of at least α and $f(p) = \chi_A(p)$ for all $p \in \mathrm{dom}(f)$.*

Now the situation changes.

Theorem 14. *For any $\alpha < 1$, $\mathrm{dom}(U)$ is α-i.o.-coarsely computable.*

Note however that $\mathrm{dom}(U)$ is never 1-i.o.-computable due to Theorem 10.

Proof sketch. The proof is similar to the argument above (that relates generic and coarse computations). Consider the value $\rho = \limsup \rho_n^U$, and consider some rational number r that is smaller than ρ but very close to it (the difference is less that $1 - \alpha$). There are infinitely many lengths for which the fraction of terminating computations exceeds r, and these lengths can be discovered ultimately, so we can consider a computable fast increasing sequence containing only those "good" lengths (not necessarily all of them). For each length in this sequence, we run U until we get r-fraction of terminating programs, and use the results for coarse prediction. The positive answers are guaranteed to be correct, while the negative answers may be incorrect. But the fraction of incorrect answers ultimately becomes less than $1 - \alpha$, since for large n the values ρ_n^U can only slightly exceed ρ (and therefore r). Again we should be careful enough to consider a fast growing sequence of lengths, so that small lengths do not interfere with large ones. □

This argument provides i.o.-coarse computability but not i.o.-generic computability. In fact, the latter may depend on the choice of the optimal machine (the rare situation we mentioned in the introduction).

Theorem 15. *There exists an optimal machine U_1 such that for any $\alpha < 1$ the set $\mathrm{dom}(U_1)$ is α-i.o.-generically computable. But there also exists an optimal machine U_2 such that $\mathrm{dom}(U_2)$ is not α-i.o.-generically computable for some $\alpha < 1$.*

The second statement appeared (in a bit different setting) in [11].

Proof sketch. In fact, we can use any "left-total" optimal machine as U_1. A machine is called *left-total* if for each n it is defined on some initial segment of $\{0,1\}^n$ in lexicographical order. (Any other computable ordering on $\{0,1\}^n$ will work.) In other words, if such a machine is defined on some string, it is also defined on all preceding strings of the same length.

It is easy to construct a left-total optimal machine U_1 by transforming a given optimal machine U into a total one: when a new description of length n is discovered for U, we add to U_1 a description of the same object using the lexicographically first string not used earlier.

Now we need to show that for a left-total machine U_1 its domain $\mathrm{dom}(U_1)$ is α-i.o.-generically computable. The idea is simple: for the left-total machine knowing the number of n-bits strings in its domain determines what are these strings. And if we know this number with some precision, we can guarantee both the positive and negative answers except for some interval in the middle (its length is the difference between the upper and lower bounds). So we use the same trick as before, but for strings of the same length. Let us see how this can be done.

Let ρ'_n be the number of strings of length n in the domain of U_1, and let $\rho' = \limsup \rho'_n$. Fix some rational threshold that is smaller than ρ' but very close to it. If it is given to us as an advice, together with the position after which ρ'_n exceed ρ' only by a very small margin, we can effectively find lengths where we can generically compute U_1 with a small fraction of omissions. Again we can form a computable increasing sequence of lengths with this property, and construct a generic predictor that is quite precise for these lengths and undefined on all other lengths (to avoid false answers for the cases where we do not have enough information).

However, this is not enough for us, since in our definition the fraction of prediction failures is calculated in the set of all strings of length *at most* n, and even if we know everything for n-bit strings, this covers only half of the strings in question. So in this way we cannot make the error less than $1/2$.

But we can repeat the trick: consider the lengths that just precede the lengths in the subsequence. For them we have no information yet, but we may consider corresponding ρ'_n and guess the lim sup *for this subsequence*. Then, using some rational threshold close to this lim sup, and the position after which ρ'_n exceed this lim sup only by a small margin, we can get a (computable increasing) subsequence of lengths where we have guaranteed good approximations *for two subsequent lengths*, thus reducing the error from $1/2$ to $1/4$ (approximately).

Now we can repeat the trick finitely many times and get arbitrary small error. Note that for this we need only finitely many bits of advice, so this still gives a partial computable predictor. The first statement is proven.

For the second part we use the argument provided by Lynch [11]. We can take the standard universal machine as U_2: let $U_2(0^e1p) = M_e(p)$ where M_e is eth machine in a standard enumeration. It remains to show that the domain of U_2 is not α-i.o.-generically computable for some $\alpha < 1$. It is because some special computably enumerable set is embedded into this domain with fixed density. Here are the details.

Post has shown that there exist *simple sets*, i.e., computably enumerable sets whose complement is infinite but does not contain an infinite enumerable subset. It is easy to construct a very sparse simple set S (either by adapting the original Post's construction or taking the set of strings whose Kolmogorov complexity is very small compared to their length). Such a set can be α-i.o.-generically computable only for very small α. Indeed, our predictor gives only finitely many negative answers (otherwise we get an enumerable infinite subset of the complement). Also it can give positive answers only for a very sparse set (in all lengths), since the entire set S is sparse (and positive answers form a subset).

It remains to take machine M_e whose domain is S, and note that the strings of the form 0^e1p form a fixed-density subset in the set of all strings; let δ be this density. A generic predictor for the domain of U_2 that gives error less than $\delta/2$ for infinitely many n, will provide i.o.-generic prediction for S with threshold approximately $\delta/2$, which is not possible. □

3.3 The Probabilistic Case

Most of the results proven in this section are negative, i.e., they show that $\mathrm{dom}(U)$ is hard to approximate *deterministically*. Does the situation change if try to get such approximations *probabilistically?* This can be understood in several ways; in the sequel we use the approach motivated by mass problems in Medvedev's sense. Let us start by giving the corresponding definitions. We consider machines with random oracle (a sequence of independent fair coin tosses).

Definition 16. *A set* $A \subseteq \{0,1\}^*$ *is* coarsely probabilistically computable *if there exists an oracle machine* Γ^X *with random oracle* X *such that the event "*Γ^X *computes a total function such that the set* $\{p \mid \Gamma^X(p) = \chi_A(p)\}$ *has density* 1*" has positive probability. A set* $A \subseteq \{0,1\}^*$ *is* generically probabilistically computable *if there exists an oracle machine* Γ^X *such that the event "*$\mathrm{dom}(\Gamma^X)$ *has density* 1 *and* $\Gamma^X(p) = \chi_A(p)$ *for all* $p \in \mathrm{dom}(\Gamma^X)$*" has positive probability. The notions of* α-coarsely probabilistically computable*,* α-generically probabilistically computable*,* α-i.o-coarsely probabilistically computable*, and* α-i.o.-generically probabilistically computable *are defined in a similar way.*

One could think that allowing probabilistic computations does not change the situation. Admittedly, it does not change it much. All the results above remain the same in the probabilistic case (with more complicated proofs), with the exception of 1-i.o.-coarse computability.

Theorem 17. *Just like in the deterministic case, for* α *sufficiently close to* 1*:*

(i) $\mathrm{dom}(U)$ *is neither* α*-coarsely probabilistically computable nor* α*-generically probabilistically computable*;
(ii) *whether* $\mathrm{dom}(U)$ *is* α*-i.o.-generically probabilistically computable or not depends on the particular choice of machine* U;
(iii) $\mathrm{dom}(U)$ *is not* 1*-i.o.-generically probabilistically computable.*

However, unlike in the deterministic case:

(iv) $\mathrm{dom}(U)$ *is always* 1*-i.o.-coarsely probabilistically computable.*

The full proof of these statements is quite long, and is omitted due to space restrictions.

Acknowledgments. This paper is based on the work done while D.D. was visiting LIRMM (Montpellier) and Poncelet laboratory (Moscow). We thank our colleagues from both laboratories (in particular the ESCAPE team, and Kolmogorov seminar group) for hostpitality. A.S. thanks Antti Valmari for interesting discussion (during RuFiDiM seminar in Turku) that was the starting point for some of the arguments in this paper. Thanks also go to three anonymous referees for helpful feedback. The authors also acknowledge the support of the Templeton Foundation.

References

1. Bienvenu, L., Muchnik, A., Shen, A., Vereshchagin, N.: Limit complexities revisited [once more]. Technical report (2012). arxiv:1204.0201
2. Bienvenu, L., Shen, A.: Random semicomputable reals revisited. In: Dinneen, M.J., Khoussainov, B., Nies, A. (eds.) Computation, Physics and Beyond. LNCS, vol. 7160, pp. 31–45. Springer, Heidelberg (2012)
3. Calude, C.S., Hertling, P.H., Khoussainov, B., Wang, Y.: Recursively enumerable reals and chaitin omega numbers. In: Meinel, C., Morvan, M. (eds.) STACS 1998. LNCS, vol. 1373, pp. 596–606. Springer, Heidelberg (1998)
4. Calude, C., Nies, A., Staiger, L., Stephan, F.: Universal recursively enumerable sets of strings. Theoretical Computer Science **412**(22), 2253–2261 (2011)
5. Downey, R., Hirschfeldt, D.: Algorithmic randomness and complexity. Theory and Applications of Computability. Springer, New York (2010)
6. Downey, R.G., Jockusch Jr., C.G., Schupp, P.E.: Asymptotic density and computably enumerable sets. Journal of Mathematical Logic **13**(02) (2013)
7. Hamkins, J.D., Miasnikov, A.: The halting problem is decidable on a set of asymptotic probability one. Notre Dame Journal of Formal Logic **47**(4) (2006)
8. Köhler, S., Schindelhauer, C., Ziegler, M.: On approximating real-world halting problems. In: Liśkiewicz, M., Reischuk, R. (eds.) FCT 2005. LNCS, vol. 3623, pp. 454–466. Springer, Heidelberg (2005)
9. Kučera, A., Slaman, T.: Randomness and recursive enumerability. SIAM Journal on Computing **31**, 199–211 (2001)
10. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications, 3rd edn. Springer, New York (2007)
11. Lynch, N.: Approximations to the halting problem. Journal of Computer and System Sciences, 9–143 (1974)
12. Miller, J.S.: Every 2-random real is Kolmogorov random. Journal of Symbolic Logic **69**(3), 907–913 (2004)
13. Nies, A.: Computability and randomness. Oxford University Press, Oxford Logic Guides (2009)
14. Schindelhauer, C., Jakoby, A.: The non-recursive power of erroneous computation. In: Pandu Rangan, C., Raman, V., Sarukkai, S. (eds.) FST TCS 1999. LNCS, vol. 1738, p. 394. Springer, Heidelberg (1999)
15. Claus Peter Schnorr: Optimal enumerations and optimal Gödel numberings. Mathematical Systems Theory **8**(2), 181–191 (1974)
16. Valmari, A.: The asymptotic proportion of hard instances of the halting problem. Technical report, November 2014. arxiv:1307.7066v2
17. Vereshchagin, N., Uspensky, V., Shen. A.: Kolmogorov complexity and algorithmic randomness (In Russian. See www.lirmm.fr/~ashen for the draft translation.). MCCME (2013)