Searching for Shortest and Least Programs[☆]

Cristian S. Calude[a], Sanjay Jain[b], Wolfgang Merkle[c], Frank Stephan[b,d]

[a]*Department of Computer Science, University of Auckland*
Private Bag 92019, Auckland, New Zealand
cristian@cs.auckland.ac.nz

[b]*Department of Computer Science, National University of Singapore*
13 Computing Drive, COM1, Singapore 117417, Republic of Singapore

[c]*Institut für Informatik, Universität Heidelberg*
Mathematikon, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany

[d]*Department of Mathematics, National University of Singapore*
10 Lower Kent Ridge Road, S17, Singapore 119076, Republic of Singapore

Abstract

The Kolmogorov complexity of a string x is defined as the length of a shortest program p of x for some appropriate universal machine U, that is, $U(p) = x$ and p is a shortest string with this property. Neither the plain nor the prefix-free version of Kolmogorov complexity are recursive but for both versions it is well-known that there are recursive exact Solovay functions, that is, recursive upper bounds for Kolmogorov complexity that are infinitely often tight. Let a coding function for a machine M be a function f such that $f(x)$ is always a program of x for M. From the existence of exact Solovay functions it

[☆]S. Jain was supported in part by NUS grant C252-000-087-001; furthermore, S. Jain and F. Stephan have been supported in part by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2016-T2-1-019 / R146-000-234-112. Part of the work was done while F. Stephan was on Sabbatical Leave to Auckland and, later, when C. S. Calude was on Sabbatical Leave to Singapore. Furthermore, W. Merkle worked on this while he was visiting the National University of Singapore on invitation for the IMS Programme "Aspects of Computation" in August / September 2017.

Email addresses: `cristian@cs.auckland.ac.nz` (Cristian S. Calude), `sanjay@comp.nus.edu.sg` (Sanjay Jain), `merkle@math.uni-heidelberg.de` (Wolfgang Merkle), `fstephan@comp.nus.edu.sg` (Frank Stephan)

URL: `http://www.math.uni-heidelberg.de/logic/merkle/merkle.html` (Wolfgang Merkle)

follows easily that for every universal machine there is a recursive coding function that maps infinitely many strings to a shortest program. Extending a recent line of research, in what follows it is investigated in which situations there is a coding function for some universal machine that maps infinitely many strings to the length-lexicographically least program. The main results which hold in the plain as well as in the prefix-free setting are the following. For every universal machine there is a recursive coding function that maps infinitely many strings to their least programs. There is a partial recursive coding function (defined in the natural way) for some universal machine that for every set maps infinitely many prefixes of the set to their least programs. Exactly for every set that is Bennett shallow (not deep), there is a recursive coding function for some universal machine that maps all prefixes of the set to their least programs. Differences between the plain and the prefix-free frameworks are obtained by considering effective sequences $I_1, I_2, \ldots$ of mutually disjoint finite sets and asking for a recursive coding function for some universal machine that maps at least one string in each set I_n to its least code. Such coding functions do not exist in the prefix-free setting but exist in the plain setting in case the sets I_n are not too small.

1. Introduction

Algorithmic Information Theory investigates among other topics the notion of an universal machine and the degree to which extent it can be inverted. It is well-known that neither the plain variant C nor the prefix-free variant H of description complexity can be effectively computed; indeed, it is even impossible that there is a partial recursive function with an infinite domain such that whenever this function is defined it coincides with the description complexity of the input [9, 15, 24, 30]. Closely related to determining the description complexity of a string is the question of whether and how one may find a shortest program for a given string, that is, a program for the string that witnesses its description complexity. The set of shortest programs has been studied in recursion theory for quite a long time [8, 12, 26, 35, 37, 38]. In a recent line of research the focus has been shifted to obtaining lists of programs that comprise the length-lexicographically least program, or least program, for short, of a given string with respect to a given universal machine U. For enumerating a list containing the least program of x, a list of size $O(|x|)$ is sufficient [7]; for each constant $c \in \{1, 2, \ldots\}$ there is a universal machine where the list size is bounded by $|x|/c + 1$. Recently it was shown that if one allows slightly larger lists of size polynomial in the length $|x|$ of x, instead of just enumerating one can actually compute a list of candidates which includes a program of x that has shortest length up to an additive constant [4, 5, 6, 39, 40, 41, 42, 43]. However it depends on the universal machine whether it is possible to include the least program in such a list [4].

This paper studies a weaker approach: The aim is to provide a program for input x such that for infinitely many x the program is actually a shortest or the least program of x. The exact constraints vary for different theorems; for example, the programs may be provided via recursive function(s); we investigate when such constraints can be satisfied and when they cannot. While in some areas of Algorithmic Information Theory the choice of the universal machine does not matter, as for example, in studies of the growth behaviour of initial segment complexities of sets [24, 25], other results are machine-dependent [4, 10, 13, 18]. In the present work several results depend heavily on the choice of the universal machine, as one aims for least programs and not just for shortest programs or even for programs which are shortest up to an additive constant. Furthermore, some notable differences between plain complexity C and prefix-free complexity H are proved, in particular when dealing with the question whether one can obtain least programs for at least one string in an effectively given sequence of finite sets of strings.

2. Notation

In this section we introduce the notation used throughout the paper. By $\mathbb{N} = \{0, 1, 2, \ldots\}$ we denote the set of natural numbers; its elements will be usually denoted by letters i, j, k, l, m, n. A STRING is a word over the binary alphabet $\{0, 1\}$, the set of all strings is denoted by $\{0, 1\}^*$ and the empty string is denoted by ε. The letters x, y, z stand for strings and $|x|$ denotes the length of the string x; if a letter denotes a program, that is, an input to a machine, it may also be denoted by p or q.

As usual, natural numbers are identified with strings via the order-preserving one-one and onto map from $\mathbb{N}$ equipped with the usual strict order to $\{0, 1\}^*$ equipped with the length-lexicographical order. Unless explicitly specified otherwise, a set is either a set of natural numbers or, equivalently, a set of strings. Furthermore, a set A is identified with its characteristic sequence $A(0)A(1)\ldots$ where $A(n) = 1$ if $n \in A$ else $A(n) = 0$. Let $x \preceq y$

denote that x is a prefix of the string y and for a set A, let $x \preceq A$ denote that $x = A(0)A(1)\ldots A(|x|-1)$. A set of strings is prefix-free in case for every string y in the set, the set does not contain any string $x \preceq y$ different from y.

The term machine refers to a Turing machine that computes a partial mapping from strings to strings. Machines are denoted by letters U, V and W. The domain of a machine U, denoted by $\mathrm{dom}(U)$, is the set of programs on which U halts. A machine U is prefix-free if and only if its domain is a prefix-free set. Occasionally, machines may be attributed as being plain in order to emphasise that the machine is not necessarily prefix-free.

In the sense of Algorithmic Information Theory, a plain universal machine U is a plain machine such that for every other machine V there is a constant c, depending only on U and V, such that for every p in the domain of V there is a q such that $U(q) = V(p)$ and $|q| \leq |p| + c$. A plain machine U is universal by adjunction for plain machines if for every machine V there exists a string q depending only on U and V such that for every program p in the domain of V it holds that $U(qp) = V(p)$, whereas in case $V(p)$ is undefined then so is $U(qp)$. The notions prefix-free universal and universal by adjunction for prefix-free machines are defined likewise, with plain replaced by prefix-free.

For a machine V, the value of plain complexity of a string x with respect to V is defined as

$$\mathrm{C}_V(x) = \min(\{|p| : V(p) = x\}),$$

where we let the value be infinite in case the minimisation is over the empty set. In case of a prefix-free machine V, we denote C_V by H_V and call the latter the prefix-free complexity with respect to V (the notation H_V was used in some earlier papers [7, 13, 16, 25], in the literature also the notation K_V is used in place of H_V). The plain complexity $\mathrm{C}(x)$ of a string x is just $\mathrm{C}_{U_{\mathrm{plain}}}(x)$ for some fixed plain universal machine U_{plain}, and the prefix-free complexity $\mathrm{H}(x)$ of x is $\mathrm{C}_{U_{\mathrm{pref}}}(x)$ for some fixed prefix-free universal machine U_{pref}.

More details can be found in the textbooks and introductory texts on algorithmic randomness by Calude [9], Chaitin [14], Downey and Hirschfeldt [15], Kolmogorov [23], Li and Vitányi [24] and Nies [30] as well as in the books on recursion theory by Odifreddi [31, 32], Rogers [34] and Soare [36].

3. Solovay functions

We start this section with the definition of Solovay functions.

Definition 1. *Let g and h be functions from $\{0,1\}^*$ to $\mathbb{N}$. Then g is a* Solovay function *for h in case g is an infinitely often tight upper bound for h up to an additive constant c, that is,*

$$\forall x\ (h(x) \leq g(x) + c) \qquad \text{and} \qquad \exists^\infty x\ (g(x) \leq h(x) + c), \tag{1}$$

and we say g is exact *at x in case $g(x) = h(x)$. In case (1) holds with $c = 0$, the function g is said to be an* exact Solovay function *for h. The function g is a* Solovay function *if g is a Solovay function for prefix-free complexity. The notion* exact Solovay function *as well as the notion* Solovay function for plain complexity *and its exact variant are defined likewise.*

Let M be a machine. Then g is a Solovay function *for M in case g is a Solovay function for the function C_M, and the notion of an* exact Solovay function *is extended to machines in a similar fashion.*

Note that in some published literature, a Solovay function has not only to satisfy the above requirements, but has also to be recursive. Observe that by definition, a function g is a Solovay function if and only if it is a Solovay function for the prefix-free universal machine used to define prefix-free complexity, and that this equivalence extends to exact Solovay functions. Furthermore, similar equivalences hold for plain complexity.

The next two remarks review the facts that there are recursive Solovay functions for plain as well as for prefix-free complexity [15].

Remark 2. *The function that maps a string to its length is a Solovay function for plain complexity. This holds because for some constant c and all n the plain complexity of all strings of length n is at most $n+c$ but is at least n for some of these strings. The latter holds because there are strictly less than 2^n programs of length strictly less than n.*

Remark 3. *The function g defined by*

$$g(x) = \begin{cases} |p|, & \textit{in case } x = \langle p, y, t\rangle \textit{ and } \mathrm{U}_{\mathrm{pref}}(p) = y \textit{ in exactly } t \textit{ steps}, \\ 2|x|, & \textit{otherwise}, \end{cases}$$

is a recursive Solovay function, where $\langle \cdot, \cdot, \cdot \rangle$ denotes as usual an effective and effectively invertible tupling function. We prove first that g is an upper bound for H *up to an additive constant. This is obvious for all strings x for which the second case in the definition of g applies. For all strings x for which the first case applies, the corresponding string p satisfies*

$$\mathrm{H}(x) = \mathrm{H}(p) \leq |p| \quad \textit{up to an additive constant.} \tag{2}$$

The two relations hold, first, because the strings x and p can be mutually computed from each other and, second, because p is in the domain of $\mathrm{U}_{\mathrm{pref}}$. *For the infinitely many such x and p where p is a shortest program for* $\mathrm{U}_{\mathrm{pref}}(p)$, *the less-than-or-equal sign in* (2) *can be replaced by equality up to an additive constant, consequently g is Solovay function.*

By a similar argument, for any superlinear time-bound t, the time-bounded version H^t *of prefix-free Kolmogorov complexity is a recursive Solovay function [19].*

By the following remark, as far as computational difficulty is concerned, there is not much difference between Solovay functions and exact Solovay functions. In particular, any function h that has a recursive Solovay function also has a recursive exact Solovay function.

Remark 4. *Let g be a Solovay function for some function h. Then there are natural numbers n_0 and integers c_0 and c_1 such that the function $\widetilde{g}$ defined by*

$$\widetilde{g}(n) = \begin{cases} \max\{g(n) - c_0, 0\}, & \textit{in case } n \leq n_0, \\ \max\{g(n) - c_1, 0\}, & \textit{otherwise}, \end{cases}$$

is an exact Solovay function for h. For a proof, let the integers c_0 and c_1 be equal to the minimum and the limit inferior, respectively, of the differences $g(n) - h(n)$. Finally, choose n_0 so large that for all n strictly larger than n_0 the corresponding difference is at least c_1.

4. Least programs and coding functions

Definition 5. *Let x and p be strings. For a given machine M,*

- *p is a* PROGRAM *of x for M in case $M(p) = x$;*

- *p is a* SHORTEST PROGRAM *of x for M in case p is a program for x that has minimum length among all programs for x;*

- *p is the* CANONICAL SHORTEST PROGRAM *of x for M in case p is a shortest program of x for M that occurs first among all shortest programs of x for M in some fixed enumeration of the domain of M;*

- *p is the* LEAST PROGRAM *of x for M in case p is a program for x that is minimum with respect to length-lexicographical order among all programs for x.*

The notions just defined are extended to plain and prefix-free complexity in the natural way. For example, a string p is a SHORTEST PROGRAM *for a string x with respect to prefix-free complexity* H *in case p is a* SHORTEST PROGRAM *of x for the universal machine U_{pref} that was used to define* H.

Definition 6. *Let M be a machine. A partial function f from $\{0,1\}^*$ to $\{0,1\}^*$ is a* PARTIAL CODING FUNCTION *for M in case $M(f(x))$ is defined and equal to x for all x where f is defined. A* CODING FUNCTION *for M is a partial coding function for M that is total.*

A partial coding function f for M MAPS A STRING x TO ITS LEAST PROGRAM *in case $f(x)$ is defined and is the least program of x for M. Related notions such as* MAPPING A STRING TO THE SHORTEST PROGRAM *are defined likewise.*

In what follows, we will focus on effectively given partial and total coding functions for plain and prefix-free universal machines.

Convention 7. *By definition, a coding function is a coding function with respect to some specific machine, called the machine* ASSOCIATED *with f. Whenever we refer to a coding function f without explicitly mentioning the associated machine, the latter will either be not relevant or understood from the context. For example, we have already introduced the notation of f mapping a string to the least program, which is meant as an abbreviation for f mapping a string to the least program for the associated machine.*

By the following remark, Solovay functions and coding functions that map infinitely many strings to their least programs are closely related.

Remark 8. *In case f is a coding function for some machine M that maps infinitely many strings to a shortest program, by definition the function $x \mapsto |f(x)|$ is an exact Solovay function for M.*

Conversely, let g be an exact Solovay function for some machine M, and let $p_0, p_1, \ldots$ be an enumeration of the domain of M. Then a coding function that maps infinitely many strings to a shortest program is obtained by mapping x to p_i where i is minimum such that $M(p_i) = x$ and $|p_i| \leq g(x)$ hold.

In what follows negative results about the existence of coding function will often be stated in terms of Solovay functions. For example, by Remark 8, in case for a certain machine there is no recursive exact Solovay function, then *a fortiori* any recursive coding function for this machine cannot map infinitely many strings to neither shortest nor least programs.

For further use, the next remark introduces universal machines with unique program lengths; such universal machines were also considered by Figueira, Stephan and Wu [18]. For such machines, every shortest program is already the least program.

Remark 9. *There is a plain as well as a prefix-free universal machine that has* UNIQUE PROGRAM LENGTHS *in the sense that for every string x there is at most one program of each length, that is, for all p and p' in the domain of the considered universal machine U, it holds that*

$$U(p) = U(p') = x \text{ and } |p| = |p'| \quad \text{implies} \quad p = p'. \tag{3}$$

For a proof, let V be either a plain or a prefix-free universal machine and let $p_0, p_1, \ldots$ be an enumeration of its domain. Transform V into a universal machine U as required of the same type by declaring U undefined on all programs p_i where for some index $j < i$ the strings $V(p_j)$ and $V(p_i)$ and the lengths of p_i and p_j are the same.

5. Computing least programs for arbitrary universal machines

In this section it is demonstrated that for all plain or prefix-free universal machines there is a recursive coding function that maps infinitely many strings to their least program.

Theorem 10. *Let U be a plain universal machine. Then there is a recursive coding function f for U that maps infinitely many strings to their least programs.*

Given any partial recursive and unbounded function h, it can be arranged that in addition there are infinitely many x with $f(x)$ being the least program of x for U such that $h(x)$ is defined and $|f(x)| \leq h(x)$.

Proof. Let $V(p)$ be the first string x found, different from all other $V(p')$ with p' length-lexicographically smaller than p, with $|p| \leq |x|$ and $h(x) \geq 2 \cdot |p|$; as for every length there exist infinitely many such strings x, $V(p)$ is defined for all p. Furthermore, as U is universal, there is a least constant c such that for infinitely many n, all $p \in \{0,1\}^n$ satisfy that $C(V(p)) \leq |p| + c$; note that $c \geq 0$, as otherwise there would be $2^{|p|}$ strings being generated by $2^{|p|} - 1$ programs of U. Now let n be so large that $n \geq c$ and there is no length $m \geq n$ such that, for all $p \in \{0,1\}^m$, $C(V(p)) < m + c$. Now one can search iteratively for values $m_0, m_1, \ldots$ satisfying the following conditions for all k and all $p \in \{0,1\}^{m_k}$:

- $n \leq m_k < m_{k+1}$;
- $h(V(p)) < m_{k+1}$;
- $|V(p)| < m_{k+1}$;
- $C(V(p)) \leq m_k + c$.

Note that the last condition together with the definition of V implies that $m_k \leq |V(p)| < m_{k+1}$ for all $p \in \{0,1\}^{m_k}$. Note that for each k one of the $p \in \{0,1\}^{m_k}$ satisfies $C(V(p)) = m_k + c$, by the choice of c. Now let c' be the largest number such that, for almost all k, each $p \in \{0,1\}^{m_k}$ satisfies either $C(V(p)) < m_k + c$ or there are at least c' programs q of length $m_k + c$ with $U(q) = V(p)$; note that c' exists, as for all x, the number of q with $U(q) = x \wedge C(x) = |q|$ is bounded by a constant. Now there is a k' such that all $k \geq k'$ satisfy this above condition. Let X be the set of all x such that there is a $k \geq k'$ and a $p \in \{0,1\}^{m_k}$ with $V(p) = x$.

The set X is recursive, as one can search for the $k \geq k'$ with $m_k \leq |x| < m_{k+1}$ and, if it exists, check for all $p \in \{0,1\}^{m_k}$ whether $V(p) = x$. For the $x \in X$ one can then enumerate the programs q with $U(q) = x$ and $|x| \leq m_k + c$ until (1) c' programs q with $|q| = m_k + c$ are found or (2) one program q with $|q| < m_k + c$ is found. In case (1), $f(x)$ is the lexicographically least of the c' programs; in case (2), $f(x) = q$ for the program q found (note that in these cases, $|f(x)| \leq m_k + c \leq 2m_k \leq h(x)$, by the definition of V). If $x \notin X$, $f(x)$ is the first program q found such that $U(q) = x$.

The so defined function f is a coding function for U and it satisfies for infinitely many k that there is a $p \in \{0,1\}^{m_k}$ such that $C(V(p)) = m_k + c$ and there are only c' many programs q with $|q| = m_k + c \wedge U(q) = V(p)$. For these strings $V(p)$, $f(V(p))$ is the least program. □

Theorem 11. *Let U be a prefix-free universal machine. Then there is a recursive coding function f for U that maps infinitely many strings to their least programs.*

Given any partial recursive and unbounded function h, it can be arranged that in addition there are infinitely many x with $f(x)$ being the least program of x for U such that $h(x)$ is defined and $|f(x)| \leq h(x)$.

Proof. By arguments as in Remarks 3 and 4, let g be a recursive exact Solovay function g for the given machine U. Let $E(x)$ be the set of programs of x for U that have length bounded by $g(x)$, that is,

$$E(x) = \{p \colon U(p) = x \text{ and } |p| \leq g(x)\}.$$

Consider the sizes of the sets $E(x)$ for strings x at which g is exact. By the coding theorem [15] there is a constant upper bound on these sizes, hence the limit inferior of these size is a natural number d. Fix n_0 such that $E(x)$ has size at least d for all strings of length at least n_0 at which g is exact.

For all x of length strictly less than n_0, let $f(x)$ be any program for x, say, the first one in some fixed enumeration of the domain of U. For all other x, enumerate the set $E(x)$ until at least d arbitrary or at least one member of length strictly less than $g(x)$ have been enumerated, then let $f(x)$ be equal to the length-lexicographically least string that has been enumerated so far. This way f is a recursive coding function for U. Furthermore, for the infinitely many x of length at least n_0 at which g is exact and $E(x)$ has size d, the string $f(x)$ is indeed the least program for x.

In order to demonstrate the second assertion, fix some arbitrary partial recursive unbounded function h. Let ℓ be a recursive and strictly monotonic function on strings such that for all x

$$h \text{ is defined at } \ell(x) \text{ and } \mathrm{H}(\ell(x)) \leq h(\ell(x)). \tag{4}$$

Such a function can be constructed inductively. In order to define $\ell(x)$, assume that for all z length-lexicographically strictly smaller than x the strings $\ell(z)$ are already defined and have all length strictly less than n. Search for a string z of length at least n that satisfies condition (4) with $\ell(x)$ replaced by z. There are infinitely many such strings because the function H has no partial recursive unbounded lower bound. Furthermore, such a string can be found effectively by simulating computations of $h(z)$ and approximating $\mathrm{H}(z)$ from above in parallel for all z of length at least n. Among all such strings, let $\ell(x)$ be equal to the one that is found first.

The strictly monotonic function ℓ has a recursive range and can be inverted on its range effectively, hence for all x the values of $\mathrm{H}(x)$ and of $\mathrm{H}(\ell(x))$ differ only by a constant c. Since the Solovay function g is exact, we have for all x

$$\forall x\ (\mathrm{H}(\ell(x)) \leq g(x) + c) \quad \text{and} \quad \exists^\infty x\ (g(x) \leq \mathrm{H}(\ell(x)) + c). \tag{5}$$

By an argument very similar to the one in Remark 4, restricted to strings of the form $\ell(x)$, we can assume that (5) holds for $c = 0$ and all x. Let

$$E_\ell(x) = \{p \colon U(p) = \ell(x) \text{ and } |p| \leq g(x)\}.$$

There are infinitely many x such that $g(x)$ is equal to $\mathrm{H}(\ell(x))$ and almost literally as in Theorem 10 we can argue that the limit inferior of the size of the set $E_\ell(x)$ over all such x is a constant d. Fix n_0 such that E_ℓ has size at least d for all such x of length at least n_0.

For all y not in the range of ℓ or of length strictly less than n_0, let $f(y)$ be any program for y. For all other y, that is, for y of the form $\ell(x)$ of length at least n_0, enumerate $E_\ell(x)$ until at least d arbitrary strings or at least one string of length strictly less than $g(x)$ have been enumerated, and then let $f(\ell(x))$ be equal to the length-lexicographically least string that has been enumerated so far. This way f is a recursive coding function for U. Furthermore, for the infinitely many x where

$$n_0 \leq |x|, \quad \mathrm{H}(\ell(x)) = g(x), \quad \text{and} \quad |E_\ell(x)| = d,$$

the string $f(\ell(x))$ is indeed the least program for $\ell(x)$ and $\mathrm{H}(\ell(x))$ is at most $h(\ell(x))$ by construction of ℓ. $\square$

6. Time-complexity of coding functions

Theorem 12. *Let t be a recursive function. There is a plain as well as a prefix-free universal machine U such that*

(i) *the machine U is universal by adjunction and has unique shortest programs,*

(ii) *there is a recursive coding function for U that maps infinitely many strings to their least programs,*

(iii) *for every partial recursive coding function g for U, every machine that computes g will run for at least $t(|x|)$ steps on almost all inputs x that g maps to their least programs.*

Proof. We give the proof for the prefix-free case and omit the proof of the plain case. Both proofs are literally the same except that the choice of the machine V has to be adapted in the natural way.

By Theorem 11 fix a recursive coding function f for some prefix-free universal machine V such that V is universal by adjunction and f maps infinitely many strings to their least programs. Let $p_0, p_1, \ldots$ be an enumeration of the domain of V.

Let $X = \{f(x) : x \in \Sigma^*\} \cup \{p_s : |p_s| \neq |f(V(p_s))|$ and for all $s' < s, [|p_s| \neq |p'_s|$ or $V(p_s) \neq V(p_{s'})]\}$.

Note that for each program p, there is exactly one program q of length $|p|$ in X such that $V(p) = V(q)$.

For all s let $U(111p_s) = V(p_s)$ and for all $p_s \in X$, for some a_s in $\{0, 1\}$ to be determined later, let $U(0\, a_s\, p_s) = V(p_s)$. On all other inputs, U does not halt. In this way only programs of form $0a_sp_s$ are the shortest for any x, and the shortest programs are least programs. This way U inherits from V the property of being universal by adjunction for prefix-free machines. Furthermore, whenever $f(x) = p_s$ is a shortest program of some string x for V, the program $0a_sp_s$ is a unique shortest program of x for U. The bit a_s will be computable from s, hence $x \mapsto 0\, a_s\, f(x)$ is a recursive coding function for U that maps infinitely many strings to their least programs.

Let $x_s = V(p_s)$ for all s. Let $M_0, M_1, \ldots$ be an enumeration of all Turing machines. The construction proceeds in stages $s = 0, 1, \ldots$ and a_s is determined during stage s. Initially, no index e is diagonalised. Say an index e

requires attention at stage s in case $e < s$ and

$$M_e(x_s) \text{ outputs } 0\,b\,p_s \text{ for some } b \text{ in } \{0,1\} \text{ in at most } t(|x_s|) \text{ steps.}$$

In case there are indices that require attention and are not yet diagonalised, consider the least such index e, let $a_s = 1 - b$ for the corresponding value of b, and declare this index e to be diagonalised. Otherwise, if there is no such index, let $a_s = 0$. By construction, for all s the machine U does not halt on $0\,(1 - a_s)\,p_s$. On the other hand, in case an index e is diagonalised during stage s, the machine M_e maps x_s to $0\,(1 - a_s)\,p_s$, hence the partial function computed by M_e is not even a partial coding function for U.

For a priority construction as above, it follows by a standard argument that every index that requires attention infinitely often will eventually be diagonalised. In case the latter assertion were false, let index e be its least counterexample, hence in particular all smaller indices are eventually diagonalised or require attention only at finitely many stages. We obtain the contradiction that at some stage the least not yet diagonalised index that requires attention is e but e is not diagonalised during that stage.

Now fix any index e such that M_e maps infinitely many x in time at most $t(|x|)$ to some shortest program q of x for U. By construction of U, for each such x there is a unique s such that q has the form $0\,b\,p_s$ and x is equal to x_s, hence e requires attention at stage s. Consequently, the index e requires attention infinitely often, and thus will eventually be diagonalised. $\square$

7. Finding least programs for strings in given finite sets

Beyond the question, whether for a given universal machine U there is a recursive or partial recursive coding function that maps infinitely many strings to least programs for U, we may ask whether it can be arranged that the coding function maps some or all strings in certain given sets to least programs. More precisely, given a plain or prefix-free universal machine U and an effective sequence $I_1, I_2, \ldots$ of finite and mutually disjoint sets of binary strings, we ask whether there is a recursive or partial recursive coding function for U

(i) that maps all strings in infinitely many sets I_n to their least program;

(ii) that maps some string in each set I_n to its least program.

For both questions, we require that the sequence $I_0, I_1, \ldots$ is given effectively.

Definition 13. *The* CANONICAL INDEX *of a finite set I of natural numbers is $\sum_{n \in I} 2^{-n}$. The finite set with canonical index r is denoted by* D_r. *A* RECURSIVE ARRAY *is a sequence $I_0, I_1, \ldots$ of mutually disjoint nonempty sets such that I_n is equal to* $\mathrm{D}_{r(n)}$ *for some recursive function r.*

By definition, all sets in a recursive array are finite. By the usual identification of natural numbers and binary strings, in what follows the sets D_r will be viewed as sets of binary strings, that is, the function $r \mapsto \mathrm{D}_r$ becomes a bijection between $\mathbb{N}$ and the finite sets of binary strings.

First we consider the case of mapping all strings in infinitely many sets I_n to their least program. The answer to Question (i) depends on the size of the sets I_j. In the special case of sets $I_1, I_2, \ldots$ where each set I_j is a singleton that contains the string that is identified with the natural number j, by Theorems 10 and 11 we obtain the affirmative answers that for every plain or prefix-free universal machine there is a recursive coding function that maps all strings in infinitely many sets I_n to their least program. However, for sets I_n of a certain minimum size, we obtain the following negative answer.

Proposition 14. *Let $I_0, I_1, \ldots$ be a recursive array such that*

$$\lim_{n \to +\infty} (\log |I_n| - \mathrm{C}(n)) = +\infty.$$

Then there is no partial recursive coding function f for a plain universal machine U such that f maps for infinitely many n all strings in I_n to their least program. A similar assertion is true in the prefix-free setting, that is, with plain replaced by prefix-free and C *replaced by* H.

Proof. We give the proof for the plain case and omit the almost identical considerations for the prefix-free case. Assume for a proof by contradiction that there were a coding function f and a machine U as in the theorem. Let s be the partial recursive function defined as follows. In case $f(x)$ is defined for all strings x in I_n, let $s(n)$ be equal to the string in I_n with the length-lexicographical greatest value of $f(x)$, while $s(n)$ is undefined for all other n. If we let k_n be equal to $\lfloor \log |I_n| \rfloor$, then there are at most $2^{k_n} - 1 < |I_n|$ programs of length strictly less than k_n, hence some string in I_n has plain complexity of at least k_n with respect to U. By construction of s, the latter is true for the string $s(n)$ for the infinitely many n where f maps all strings in I_n to their least program for U. On the other hand, by universality of U there is a constant c such that for all n where $s(n)$ is defined, the complexity of $s(n)$ with respect to U is at most $\mathrm{C}(n)+c$. So there are infinitely many n such that the complexity of $s(n)$ with respect to U is at least k_n and at most $\mathrm{C}(n)+c$, which contradicts the assumption on the size of the sets I_n. $\square$

From the standard upper bounds $\log n + c$ and $\log n + 2 \log \log n$ for the plain and prefix-free complexity of n, the following corollary of Proposition 14 is immediate.

Corollary 15. *Let $I_0, I_1, \dots$ be a recursive array.*

If $|I_n| - n$ goes to infinity, then there is no partial recursive coding function f for a plain universal machine such that f maps for infinitely many j all strings in I_j to their least program.

If the size of I_n is at least n^2 then there is no partial recursive coding function f for a prefix-free universal machine such that f maps for infinitely many j all strings in I_j to their least program.

In the remainder of this section, we consider the case of mapping some string in each set I_n to their least program, that is, we consider Question (ii). For a start, we note that when U is a plain universal machine, then the machine V with $V(1p) = U(p)$ and $V(0p) = p$ satisfies that the mapping $p \mapsto 0p$ maps in each interval $I_n = \{0,1\}^n$ at least one string to its least program. Now we show that we obtain a negative answer in the case of coding functions for prefix-free machines.

Proposition 16. *Let $I_0, I_1, \dots$ be a recursive array and let U be a prefix-free universal machine. Then there is no recursive coding function for U that, for all n, gives the least program for some string in I_n.*

By Remark 8 and the definition of prefix-free coding function, Proposition 16 is an immediate consequence of the following proposition.

Proposition 17. *Let $I_0, I_1, \dots$ be a recursive array and let U be a prefix-free universal machine. Then there is no recursive exact Solovay function for U that for all n is exact at some string in I_n.*

Proof. For a proof by contradiction, assume that there was a Solovay function g as detailed in the proposition. For every i let n_i be equal to the least index n that satisfies

$$\sum_{x \in I_n} 2^{-g(x)} \leq 2^{-2i-1}.$$

Such an index exists and can be found effectively because for the upper bound g of H_U the sum of $2^{-g(x)}$ over all strings x is finite. By the Kraft-Chaitin Theorem, there is a prefix-free machine V such that $\mathrm{H}_V(x)$ is at most $g(x) - i$ for every string $x \in I_{n_i}$. This holds because we have

$$\sum_{i \in \mathbb{N}} \sum_{x \in I_{n_i}} 2^{i-g(x)} \leq \sum_{i \in \mathbb{N}} 2^i \cdot 2^{-2i-1} = 1.$$

Since U is universal, the machine V witnesses that $\mathrm{H}_U(x)$ is, up to an additive constant, at most $g(x) - i$ for every i and every string x in I_{n_i}. Consequently, it is impossible that g is exact for some string in each interval I_n. □

Consider the set of strings on which a recursive exact Solovay function for a prefix-free universal machine is exact. As an easy consequence of Proposition 17, we obtain next that the principal function of such a set is never dominated by a recursive function.

Proposition 18. *Let a recursive exact Solovay function for a prefix-free universal machine be given. Then there is no recursive function r such that for all n, the Solovay function is exact on at least n strings among the first $r(n)$ strings.*

Proof. Assume the proposition were false and this would be witnessed by an exact Solovay function g and a recursive function r, where we can assume that r is strictly monotonic. We define inductively a partition of the set of all strings into consecutive nonempty finite intervals $I_0, I_1, \ldots$, that is, there is a strictly increasing sequence $n_0, n_1, \ldots$ such that the union of the intervals I_0 through I_j contains the least n_j strings in length-lexicographical order. Let n_0 be equal to $r(1)$ and let n_{j+1} be equal to $r(n_j + 1)$. By construction, the Solovay function g is exact on some string in each set of the recursive array $I_0, I_1, \ldots$, which contradicts Proposition 17. □

Remark 19. *Given a coding function f for some machine M, let*

$$\mathrm{L}(M, f) = \{x \in \{0,1\}^* \colon f(x) \textit{ is the least program of } x \textit{ for } M\} \qquad (6)$$

be the set of strings x that f maps to the least program of x with respect to M. If now f is recursive, the set $\mathrm{L}(M, f)$ is co-r.e., that is, has a recursively enumerable complement. For a proof, it suffices to observe that for all x, *the string $f(x)$ is a program of x for M, hence the complement of* $\mathrm{L}(M, f)$ *contains exactly the strings x such that there is a program of x for M that is length-lexicographically strictly smaller than $f(x)$.*

The following lemma extends by virtually the same proof to prefix-free universal machines; however, a stronger form of the corresponding assertion has already been obtained as Proposition 17.

Lemma 20. *Let $I_0, I_1, \dots$ be a recursive array and let f be a recursive coding function for some plain universal machine U. Then the limit inferior of the function*

$$r \colon n \mapsto |I_n \cap \mathrm{L}(U, f)|,$$

where $\mathrm{L}(U, f)$ *is the set in (6), is either* 0 *or* $+\infty$.

Proof. Assume for a contradiction that the limit inferior of r was a nonzero natural number d. Fix some natural number n_0 such that $r(n) \geq d$ for all $n \geq n_0$. Consider a machine M that on an input of the form 1^k works as follows. The machine M enumerates the complement of the set $\mathrm{L}(M, f)$ until some index $n \geq n_0$ is found such that all but d strings in I_n have already been enumerated and among these d strings there is some string x such that $|f(x)| > 2k$. Letting x_k be the least string with the latter property among these d strings, the machine M outputs x_k. By construction and universality of U, there is a constant c such that for all k, we have

$$2k < |f(x)| = \mathrm{C}_U(x_k) \leq \mathrm{C}_M(x_k) + c \leq k + c,$$

which is a contradiction for all $k \geq c$. Here the equation holds since the set of d strings from which x_k is chosen is of the form $I_n \cap \mathrm{L}(M, f)$ for some $n \geq n_0$, hence $f(x_k)$ is the least program of x_k for M. $\square$

By Lemma 20, given a recursive array $I_0, I_1, \dots$ and a recursive coding function f for some plain universal machine, the number of strings per set I_j which f maps to their least program either is equal to 0 for infinitely many indices j or goes to infinity. Whether the latter can be achieved may depend on the sizes of the sets in the recursive array and may further require to use an appropriate universal machine, as is asserted next in Theorems 21 and 22.

Theorem 21. *Let $I_0, I_1, \dots$ be a recursive array such that the set I_j has size of at least 2^j. Then there are a plain universal machine U and a recursive* *coding function f for U that maps at least one string in each set I_j to its least program. In addition, the coding function f can be chosen such that whenever the string $f(x)$ is a shortest program, it is already the least program.*

Proof. Let f_0 be a recursive coding function for some plain universal machine U_0. For all p in the domain of U_0, let the machine U output $U_0(p)$ on both of the inputs $10p$ and $110p$. Furthermore, for all n, let $U(0\,p_i^n) = x_i^n$ where p_i^n and x_i^n are the i-th string of length n and in I_n, respectively, for $i = 1, \ldots, 2^n$. On all other inputs, let U be undefined. Then the function f defined by

$$f(x) = \begin{cases} 0\,p_i^n, & \text{if } x = x_i^n \text{ for some } i, n \text{ with } i < 2^n \text{ and } n \leq |f_0(x)| + 5, \\ 110 f_0(x), & \text{otherwise,} \end{cases}$$

with associated universal machine U is a coding function as required. For each n, there are at most $2^n - 1$ strings of length strictly less than n, hence for some i the shortest program of x_i^n for U_0 has length at least n and thus $0\,p_i^n$ is the least program of x_i^n for U. Furthermore, a program $f(x)$ of the form $110p$ cannot be a shortest program for U because $10p$ is a program for the same string. Consequently, in case $f(x)$ is a shortest program of x for U, it follows that $f(x)$ is the unique program of x for U of the form $0p$ and is indeed the least program of U. Finally, in order to ensure that $f(x)$ is the least program whenever it is a shortest program, it suffices to choose U_0 such that whenever a string x is in I_n, the least program p of x for U_0 does not have length $n-1$, hence the only possible least programs $0p_i^n$ and $10p$ of x for U differ in length. □

Theorem 22. *Let $I_0, I_1, \ldots$ be a recursive array. Then there is a plain universal machine U such that every partial recursive coding function for U, for some j, fails to map every string in I_j to a shortest program. A similar assertion is true in the prefix-free setting, that is, with plain replaced by prefix-free.*

Proof. We give the proof for the plain case and omit the very similar argument for the prefix-free case. Let $\varphi_0, \varphi_1, \ldots$ be an acceptable numbering of all partial recursive functions. We construct a plain universal machine U such that that for all e the partial function φ_e maps no string in I_e to a least program. Fix some plain universal machine V. For all strings p in the domain of V, let $U(10p) = V(p)$. For all such p, let $U(0p) = V(p)$ in case $V(p)$

is in I_e and $\varphi_e(V(p)) = 10p$; let U be undefined on all other strings.

In order to verify the construction, fix an arbitrary index e and assume that φ_e is a partial coding function for U that is defined on some string x in I_e. Then $\varphi_e(x)$ is equal to up for some string u in $\{0, 10\}$, where $U(up) = x$, hence $V(p) = x$. In case $u = 10$, we have $U(0p) = x$, hence up is not a shortest program for x. In case $u = 0$, the machine U is undefined on up, hence φ_e is not a partial coding function for U. □

8. Least programs for prefixes of a set

Recall that a set A can be identified with its characteristic sequence. This way it makes sense to speak of the prefixes $A(0)A(1)\dots A(n-1)$ of the set A. We investigate the question whether there are partial recursive or recursive coding functions that map all or at least infinitely many prefixes of some set to least programs. By the next two theorems, the latter question about mapping infinitely many prefixes to least programs is answered in the affirmative in the plain as well as in the prefix-free setting. More precisely, in both settings this can be achieved for all sets by some fixed partial recursive coding function and for exactly the sets that are Bennett shallow by recursive coding functions.

Theorem 23. *There exist a prefix-free universal machine U and partial recursive coding function f for U such that for every set A, f maps infinitely many prefixes of A to their least program. A similar assertion is true for the plain case.*

Proof. The proof is given for the prefix-free case, the similar proof for the plain case is omitted. In the prefix-free case, one starts with a prefix-free universal machine V from which a coding function f as required and its associated universal machine U are constructed. Let c be a constant that is so large that for every string x and every bit a it holds that $\mathrm{H}_V(xa) \geq \mathrm{H}_V(x) + 3 - 2c$. For all n, let

$$I_n = \{x \in \{0,1\}^* \colon 2^n \leq |x| < 2^{n+1}\}$$

and for x in I_n, for the unique m such that

$$|x| = 2^n + m, \quad \text{let} \quad J_x = \{4m - c, \dots, 4m + c\}.$$

Fix some enumeration $p_0, p_1, \ldots$ of the domain of V and let $x_s = V(p_s)$. For the scope of this proof, call the string p_s a *special program* if the length of p_s is in J_{x_s}, and call x a *special string* if the least program of x for V is a special program, (that is, if the prefix-free complexity of x with respect to V is in J_x).

Now one constructs a new prefix-free universal machine U by doing the following, successively for $s = 0, 1, \ldots$:

- let $U(p_s) = V(p_s)$ in case p_s is not a special program;
- let $U(p_s 0^{k_s}) = V(p_s)$ for the unique natural number k_s such that $p_s 0^{k_s}$ has length $\max J_{x_s}$ in case p_s is the first special program of x_s, that is, in case p_s is a special program but for all $j < s$, the program p_j is not special or is not a program of x_s;

however U remains undefined on all other inputs. By construction, for every special string x its least program for U is the unique program of x for U of length $\max J_x$. Therefore the partial recursive function coding function f for U defined as follows maps all special strings to a least program. On input x, if an effective search for a program of x for U of length $\max J_x$ terminates successfully, then $f(x) = p$ for the first such program p that has been found, else $f(x)$ remains undefined.

We conclude by showing that among the prefixes of any given set A there are infinitely many special strings. So fix a set A and an index n and let $z_0 \prec \ldots \prec z_{2^n-1}$ be the prefixes of A in the set I_n. For all sufficiently large n, the assertion

$$\mathrm{H}_V(z_i) \leq \max J_{z_i} \tag{7}$$

is false in case $i = 0$ and true in case $i = 2^n - 1$, because $\max J_{z_i}$ is equal to the constant c in the former case and is equal to $2|z_i| + c - 2$ in the latter case. Consequently, for such n there is a least index r such that $1 < r \leq 2^n - 1$ and (7) is true with i replaced by r. In order to show that in this situation z_r is special, by definition it suffices to show that $\mathrm{H}_V(z_i)$ is at least $\min J_{z_r}$. The latter follows by a sort of discrete continuity argument. More precisely, we have

$$\begin{aligned} \mathrm{H}_V(z_r) &\geq \mathrm{H}_V(z_{r-1}) + 3 - 2c \geq \max J_{z_{r-1}} + 4 - 2c \\ &= \max J_{z_r} - 4 + 4 - 2c = \min J_{z_r}, \end{aligned}$$

where the relations hold by choice of c, by minimality of r and by using two times the definition of the intervals J_x. $\square$

Remark 24. *The partial recursive prefix-free coding function constructed in the proof of Theorem 23 could be changed at finitely many arguments in order to map at least one string in each of the sets I_j to its least program, where $I_0, I_1, \ldots$ is the recursive partition of $\{0,1\}^*$ defined in the proof above. Recall that by Proposition 17, this cannot be achieved by a recursive prefix-free coding function.*

Theorem 23 suggests to ask for which sets there is not just a partial recursive but actually a recursive coding function for some plain or prefix-free universal machine that maps infinitely many prefixes of the set to a least program. Theorem 27, proven below, shows that such recursive coding functions exist exactly for the sets that are Bennett shallow, that is, are not Bennett deep.

The notion of Bennett depth was introduced by Bennett [2] and has been further investigated recently by Moser and Stephan [28, 29] as well as by Downey, MacInerney and Ng [17], who provide various examples of deep and non-deep sets in a recursion-theoretic setting. Similar investigations were also carried out in a complexity-theoretic setting [1, 21, 27], though often with a somewhat different approach to define the notion of depth. In the case of sets, Bennet's notion of depth can be defined as follows.

Definition 25. *A set A is* BENNETT DEEP FOR PLAIN COMPLEXITY *in case for every recursive upper bound g for plain complexity* C*, we have*

$$\lim_{n\to\infty} (g(A(0)\ldots A(n-1)) - \mathrm{C}(A(0)\ldots A(n-1))) = \infty.$$

The notion of a set that is BENNETT DEEP FOR PREFIX-FREE COMPLEXITY *is defined likewise, and in both settings a set is* BENNETT SHALLOW *if it is not Bennett deep.*

Proposition 26. *A set A is Bennett shallow for plain complexity if and only if there is a recursive Solovay function g for plain complexity that up to an additive constant c is tight on infinitely many prefixes x of A in the sense that $g(x) - \mathrm{C}(x) \leq c$. A similar equivalence holds in the prefix-free case.*

Proof. We show the equivalence in the plain setting, as the proof for the prefix-free case is essentially the same. If the set A is Bennett shallow for plain complexity, then any function g that witnesses the latter fact according to Definition 25 is already a Solovay function as required. Conversely, given such a Solovay function g where d is the supremum of the values $\mathrm{C}(x)-g(x)$, then d is finite and the function $g+d$ is a recursive upper bound for C that witnesses that A is not Bennett deep. □

If there exists a recursive coding function f for some plain universal machine that maps infinitely many prefixes of A to their least program, then the function $g\colon x \mapsto |f(x)|$ is a Solovay function that is tight on infinitely many prefixes of A, hence A is Bennett shallow according to Proposition 26. By the following theorem, this implication can be reversed.

Theorem 27. *A set A is Bennett shallow with respect to plain complexity if and only if there exists a recursive coding function for some plain universal machine that maps infinitely many prefixes of A to their least program.*

A similar assertion holds in the prefix-free setting, that is, with plain replaced by prefix-free on both sides of the equivalence.

Proof. The proof is given for the prefix-free setting; the similar proof for the plain setting is omitted. The direction from right to left follows by the discussion preceding the theorem.

Concerning the reverse implication, let A be any set that is Bennett shallow with respect to prefix-free complexity. Fix some prefix-free universal machine V that is universal by adjunction and recall that H_V is the prefix-free complexity with respect to V. By shallowness of A and replicating the constructions in Remarks 3 and 4, one obtains a recursive Solovay function g for H_V such that $\mathrm{H}_V(x) \leq g(x)$ for all x and such that for some natural numbers c_1 and c there are infinitely many prefixes x of A such that

$$g(x) \leq \mathrm{H}(x) + c_1 < \mathrm{H}_V(x) + c, \tag{8}$$

where the first inequality follows by Proposition 26 and the second by universality of V. For these infinitely many prefixes x of A, consider the remainders modulo c of the values $\mathrm{H}_V(x)$ and fix some remainder c' that is attained for infinitely many such x. For every natural number ℓ, let

$$I_\ell = \{c\ell + c', c\ell + c' + 1, \ldots, c(\ell+1) + c' - 1\}$$

and observe that the sets I_ℓ partition the set $\mathbb{N} \setminus \{0, \ldots, c'-1\}$. By choice of c' and (8), there are then infinitely many prefixes x of A such that for some ℓ it holds that

$$\min I_\ell = H_V(x) \leq g(x) \leq \max I_\ell.$$

Furthermore, fix a recursive one-to-one enumeration $p_0, p_1, \ldots$ of the domain of V and a string r of length greater than c such that r is incompatible with all the p_i, that is, none of the prefixes or extensions of r, including r itself, is in the domain of V. Next define the recursive equivalence relation $\sim$ on the natural numbers by

$$s \sim t \quad \text{if and only if} \quad V(p_s) = V(p_t) \text{ and } |p_t|, |p_s| \in I_\ell, \text{ for some } \ell.$$

That is, for all indices i in the same equivalence class of the relation $\sim$, the strings p_i are programs for V of the same string x and of similar lengths, more precisely, all these lengths fall into the same interval I_ℓ. The goal is to carry over from V to U only one program from each equivalence class. In order to make U universal by adjunction, however, new programs are added that are so long that none of them will be a shortest program for some string with respect to U.

We construct a prefix-free universal machine U and a coding function f for U as required. For each s and for $x = V(p_s)$,

let $U(rp_s) = x$ and if there is no $t < s$ with $t \sim s$ then let $U(p_s) = x$.

Let $f(x)$ be the first program p for U found by some effective search such that $U(p) = x$, $r \not\preceq p$ and $|p| \leq \max I_\ell$ for the ℓ with $g(x) \in I_\ell$. Observe that such a p exists for almost all x by the construction of U and the fact that $\mathrm{H}(x) \leq g(x)$. So f is a recursive coding function for U. For the infinitely many prefixes x of A, where $\mathrm{H}_V(x)$ is minimum in some interval I_ℓ, the latter interval contains $g(x)$ and is the least interval that contains the length of a program p with $U(p) = x$; the function f maps all these strings x to least programs of U. $\square$

In the remainder of this section we ask whether there is a set such that all its prefixes are effectively mapped to their least programs by a coding function, for some plain or prefix-free universal machine. For both settings, the answer is affirmative in the case of partial recursive coding functions. Before we state the latter in Theorem 29, we show in Remark 28 that for certain universal machines there cannot be such a partial recursive coding function and we observe that all such sets have positive Hausdorff dimension in Theorem 30. This implies that such sets are also complex, as defined by Kjos-Hanssen, Merkle and Stephan [22]. We then conclude the section by observing that for recursive coding functions the answer is negative in the prefix-free setting and is unknown in the plain setting.

Remark 28. *There is a universal plain machine U such that for every partial recursive function f there is a length such that f maps no string of this length to its least program for U. A similar assertion holds in the prefix-free setting, that is, with universal replaced by universal for prefix-free machines.*

In order to obtain a machine U as above, in both, the plain and the prefix-free setting, it suffices to construct U such that for all e and all strings x of length e where $\varphi_e(x)$ is defined, the length of the latter string is odd if and only if the length of the least program of x for U is even.

Theorem 29. *There is partial recursive coding function f for some plain universal machine that is universal by adjunction such that f maps all prefixes of some set to their least program. A similar assertion holds in the prefix-free setting.*

Proof. The proof is given for the prefix-free case, the similar proof for the plain case is omitted. Let V be any prefix-free universal machine. There are a set A and a constant c such that, if we let for all strings x

$$J_x = \{\lfloor |x|/2 \rfloor - c, \ldots, \lceil |x|/2 \rceil + c\},$$

then for every prefix x of A, the prefix-free complexity of x with respect to V is in J_x [11, 33]. Fix some enumeration $p_0, p_1, \ldots$ of the domain of V and let $x_s = V(p_s)$. For the scope of this proof, call the string p_s a *special program* if the length of p_s is in J_{x_s}, and call x a *special string* if the least program of x for V is a special program. Note that all prefixes of A are special strings.

Now one constructs a new prefix-free universal machine U by doing the following, successively for $s = 0, 1, \ldots$,

- let $U(p_s) = V(p_s)$ in case the program p_s is not special;
- let $U(p_s 0^{k_s}) = V(p_s)$ for the unique natural number k_s such that $p_s 0^{k_s}$ has length $\max J_{x_s}$ in case p_s is the first special program for x_s (that is, in case p_s is a special program but p_j is not a special program for all $j < s$ where $x_j = x_s$);

while U remains undefined on all other inputs. By construction, for every special string x its least program for U is the unique program of x for U of length $\max J_x$. Therefore the partial recursive coding function f for U defined as follows maps all special strings and, in particular, all prefixes of A to their least program. On input x, if an effective search for a program of x for U of length $\max J_x$ terminates successfully by finding p, then $f(x) = p$ else $f(x)$ remains undefined.

We conclude by extending the domain of U in order to obtain a prefix-free machine U' that is universal by adjunction and where its coding function f maps every special string to its least program for U'. Fix some prefix-free machine W that is universal by adjunction and a constant d such that for every string its prefix-free complexity with respect to U and with respect to W differ by at most d. Let r be a string that has neither a prefix nor an extension in the domain of U and let

$$U'(p) = U(p) \text{ for all } p \in \mathrm{dom}(U) \text{ and } U'(r0^d q) = W(q) \text{ for all } q \in \mathrm{dom}(W).$$

By construction, every program of U is a program of U' with the same output and every least program for U is a least program for U'. Hence f is a coding function for U' that maps every special string to its least program for U'. In the case of plain complexity, the last part of the proof becomes slightly less elegant: for p and q as above, the corresponding programs for U' are $0p$ and $1^{d+2}q$, where then $x \mapsto 0f(x)$ is a coding function as required. □

Theorem 30. *If there is a partial recursive plain or prefix-free coding function that maps all prefixes of some set A to a shortest program then A has positive constructive Hausdorff dimension. More precisely, there is a rational constant $q > 0$ such that the Kolmogorov complexity of $A \upharpoonright n$ is at least qn for almost all n.*

Proof. The proof is done for the plain case and the prefix-free case is similar. Fix any universal machine U. Assume that there is a partial-recursive coding function f which provides a shortest program for all prefixes of A. Let $g(x)$ be the shortest prefix of x such that $|f(g(x))| \geq |f(y)|$ for all $y \preceq x$. Intuitively, $g(x)$ denotes the prefix of x with the largest code (as given by f). For any $d \in \mathbb{N}$, let $S_d = \{x : |\{y : g(y) = x\}| \geq 2^{2d+4}\}$. Note that for $x \in S_d$, there are at least 2^{2d+4} y such that $|f(x)| \geq |f(y)| \geq C_U(y)$. Thus, for any length n, there are at most 2^{n-2d-3} x's such that $|f(x)| = n$, and $x \in S_d$. For each $x \in S_d$, assign a unique string u_x of length $|f(x)| - d - 1$, and let $V(1^d 0 u_x) = x$ (note that there are enough such u's available by the argument above, and V is well defined). By universality of U, there exists a constant c such that for all x, all V-programs for x have length larger than $C_U(x) - c$.

Now consider x with $x = g(x)$ and x being a prefix of A. By hypothesis $f(x)$ is a U-shortest program for x. Thus, V does not have a program of length $|f(x)| - c$ or smaller for these x. Now, one of the strings $A \upharpoonright (|x|+1)$, $\ldots$, $A \upharpoonright (|x| + 2^{2c+4})$ must have Kolmogorov complexity (with respect to U) at least $|f(x)| + 1$: otherwise there would be at least 2^{2c+4} strings y with $g(y) = x$, and thus V has a program $1^c 0 u_x$ for x of length $|f(x)| - c$, contradicting the above established fact that V cannot have programs for such x of length $|f(x)| - c$ or shorter.

Inductively, using the above argument, in every interval $[2^{2c+4} \cdot n, \ldots, 2^{2c+4} \cdot (n+1) - 1]$ of natural numbers there is an m such that $C_U(A \upharpoonright m) \geq n$. For the other m in this interval, $C_U(A \upharpoonright m) \geq n - c'$ for some constant c' (as all prefixes of A of length in the above interval can be obtained from each other by adding/deleting a suffix bounded by length 2^{2c+4}). Thus there is a rational number q (any q with $0 < q < 2^{-2c-4}$ would do) such that $C_U(A \upharpoonright m) \geq qm$ for almost all m. This implies that A has at least constructive Hausdorff dimension q, in short, A has positive constructive Hausdorff dimension. □

Note that the above result gives some motivation for the construction of the witness set in Theorem 29. Furthermore, it suggests to ask whether Theorem 29 can be strengthened by replacing the partial recursive coding functions by recursive ones. In the setting of prefix-free complexity the answer is negative, that is, there is no recursive coding function for some prefix-free universal machine that maps all prefixes of some set to their least program. Such a coding function would contradict Proposition 17, by an easy argument similar to the one in Remark 24. In the setting of plain complexity, we state this question as an open problem.

Open Problem 31. *Is there a recursive coding function for some plain universal machine that maps all prefixes of some set to their least program?*

Acknowledgment

We thank the anonymous referees for comments that improved the presentation.

References

[1] Luis Filipe Coelho Antunes, Lance Fortnow, Dieter van Melkebeek and N. Variyam Vinodchandran. Computational depth: Concept and applications. *Theoretical Computer Science*, 354:391–404, 2006.

[2] Charles H. Bennett. Logical depth and physical complexity. *The Universal Turing Machine, A Half-Century Survey*, 227–257, 1988.

[3] Laurent Bienvenu, Rodney G. Downey, Wolfgang Merkle and André Nies. Solovay functions and K-triviality. *Journal of Computer and System Sciences*, 81:1575–1591, 2015.

[4] Bruno Bauwens, Anton Makhlin, Nikolay Vereshchagin and Marius Zimand. Short lists with short programs in short time. *Computational Complexity* 27(1):31–61, 2018.

[5] Bruno Bauwens and Alexander Shen. Complexity of complexity and strings with maximal plain and prefix Kolmogorov complexity. *The Journal of Symbolic Logic*, 79(2):620–632, June 2014.

[6] Bruno Bauwens and Marius Zimand. Linear list-approximation for short programs (or the power of a few random bits). *IEEE 29th Conference on Computational Complexity (CCC)*, 241–247, 2014.

[7] Richard Beigel, Harry Buhrman, Peter Fejer, Lance Fortnow, Piotr Grabowski, Luc Longpré, Andrej Muchnik, Frank Stephan and Leen Torenvliet. Enumerations of the Kolmogorov function. *The Journal of Symbolic Logic*, 71(2):501–528, 2006.

[8] Manuel Blum. On the size of machines. *Information and Control*, 11:257–265, 1967.

[9] Cristian S. Calude. *Information and Randomness - An Algorithmic Perspective.* Second Edition, Springer, Heidelberg, 2002.

[10] Cristian S. Calude, Michael J. Dinneen and Chi-Kou Shu. Computing a glimpse of randomness. *Experimental Mathematics*, 11(3):361–370, 2002.

[11] Cristian S. Calude, Nicholas J. Hay and Frank Stephan. Representation of left-computable ϵ-random reals. *Journal of Computer and System Sciences*, 77(4):812–819, 2011.

[12] Cristian S. Calude, Hajime Ishihara and Takeshi Yamaguchi. Coding with minimal programs. *International Journal of Foundations of Computer Science*, 12(4):479–489, 2001.

[13] Cristian S. Calude, André Nies, Ludwig Staiger and Frank Stephan. Universal recursively enumerable sets of strings. *Theoretical Computer Science* 412(22):2253–2261, 2011.

[14] Gregory J. Chaitin. Information-theoretic characterizations of recursive infinite strings. *Theoretical Computer Science*, 2(1):45–48, 1976.

[15] Rodney G. Downey and Denis R. Hirschfeldt. *Algorithmic Randomness and Complexity.* Theory and Applications of Computability. Springer, New York, 2010.

[16] Rodney G. Downey, Denis R. Hirschfeldt, André Nies and Frank Stephan. Trivial Reals. *Proceedings of the 7th and 8th Asian Logic Conferences* (7th Conference: Hsi-Tou, Taiwan 6 – 10 June 1999, 8th Conference: Chongqing, China 29 August – 2 September 2002), World Scientific, 103–131, 2003.

[17] Rod G. Downey, Michael MacInerney and Keng Meng Ng. Lowness and logical depth. *Theoretical Computer Science* 702, 23–33, 2017.

[18] Santiago Figueira, Frank Stephan and Guohua Wu. Randomness and universal machines. *Journal of Complexity*, 22:738–751, 2006.

[19] Rupert Hölzl, Thorsten Kräling, and Wolfgang Merkle. Time-bounded Kolmogorov complexity and Solovay Functions. *Theory of Computing Systems*, 52: 80–94, 2013.

[20] Sanjay Jain and Jason Teutsch. Enumerations including laconic enumerators. *Theoretical Computer Science*, 700:89–95, 2017.

[21] David W. Juedes, James I. Lathrop and Jack H. Lutz. Computational depth and reducibility. *Theoretical Computer Science*, 132:37–70, 1994.

[22] Bjørn Kjos-Hanssen, Wolfgang Merkle and Frank Stephan. Kolmogorov complexity and the recursion theorem. *Transactions of the American Mathematical Society*, 263:5465–5480, 2011.

[23] Andrey N. Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2:157–168, 1968.

[24] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Texts in Computer Science. Springer, New York, third edition, 2008.

[25] Wolfgang Merkle and Frank Stephan. On C-degrees, H-degrees and T-degrees. *Twenty-Second Annual IEEE Conference on Computational Complexity* (CCC 2007), San Diego, USA, 12 - 16 June 2007, 60–69, 2007.

[26] Albert R. Meyer. Program size in restricted programming languages. *Information and Control*, 21:382–394, 1972.

[27] Philippe Moser. On the polynomial depth of various sets of random strings. *Theoretical Computer Science*, 477:96–108, 2013.

[28] Philippe Moser and Frank Stephan. Depth, highness and DNR degrees. *Discrete Mathematics and Theoretical Computer Science*, 19(4–2):1–15, 2017.

[29] Philippe Moser and Frank Stephan. Limit-depth and DNR degrees. *Information Processing Letters* 135:36–40, 2018.

[30] André Nies. *Computability and Randomness*, Oxford Science Publications, 2009.

[31] Piergiorgio Odifreddi. *Classical Recursion Theory.* North-Holland, Amsterdam, 1989.

[32] Piergiorgio Odifreddi. *Classical Recursion Theory, Volume II.* Elsevier, Amsterdam, 1999.

[33] Jan Reimann and Frank Stephan. Hierarchies of randomness tests. *Proceedings of the Ninth Asian Logic Conference, "Mathematical Logic in Asia"*, World Scientific, Singapore, 215–232, 2006.

[34] Hartley Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

[35] Marcus Schaefer. A guided tour of minimal indices and shortest descriptions. *Archive for Mathematical Logic*, 37(8):521–548, 1998.

[36] Robert I. Soare. *Recursively Enumerable Sets and Degrees. A Study of Computable Functions and Computably Generated Sets.* Springer, Heidelberg, 1987.

[37] Frank Stephan and Jason Teutsch. Immunity and hyperimmunity for generalized random strings. *Notre Dame Journal of Formal Logic*, 49:107–125, 2008.

[38] Frank Stephan and Jason Teutsch. An incomplete set of shortest descriptions. *The Journal of Symbolic Logic*, 77(1):291–307, March 2012.

[39] Jason Teutsch. Short lists for shortest descriptions in short time. *Computational Complexity*, 23(4):565–583, 2014.

[40] Jason Teutsch and Marius Zimand. On approximate decidability of minimal programs. *ACM Transactions on Computational Theory*, 7(4):17:1–17:16, August 2015.

[41] Jason Teutsch and Marius Zimand. A brief on short descriptions. *ACM SIGACT News*, Complexity Theory Column, 47(1):42–67, 2016.

[42] Nikolay Vereshchagin. Short lists with short programs from programs of functions and strings. *Theory of Computing Systems*, 61(4):1440–1450, 2017.

[43] Marius Zimand. Short lists with short programs in short time – a short proof. *Proceedings of the Tenth CiE*, Budapest, Hungary, *Language, Life, Limits*, *Springer LNCS*, 8493:403–408, 2014.