Computing Measure as a Primitive Operation in Real Number Computation

Christine Gaßner
Universität Greifswald, Germany
gassnerc@uni-greifswald.de

Arno Pauly
Department of Computer Science, Swansea University, UK
https://www.cs.swan.ac.uk/~cspauly/
arno.m.pauly@gmail.com

Florian Steinberg
INRIA, Sophia Antipolis, France
fsteinberg@gmail.com

Abstract

We study the power of BSS-machines enhanced with abilities such as computing the measure of a BSS-decidable set or computing limits of BSS-computable converging sequences. Our variations coalesce into just two equivalence classes, each of which also can be described as a lower cone in the Weihrauch degrees.

We then classify computational tasks such as computing the measure of Δ^0_2-set of reals, integrating piece-wise continuous functions and recovering a continuous function from an $\mathrm{L}_1([0,1])$-description. All these share the Weihrauch degree lim.

2012 ACM Subject Classification Theory of computation → Abstract machines; Theory of computation → Turing machines; Mathematics of computing → Point-set topology; Mathematics of computing → Integral calculus

Keywords and phrases BSS-machine, Weihrauch reducibility, integrable function, Lebesgue measure, computable analysis

Digital Object Identifier 10.4230/LIPIcs.CSL.2021.22

1 Introduction

Computing over abstract data types using register machines makes the fully realistic notion of computability that computable analysis uses more approachable and easier to use in applications. This is well known and the central topic of work such as [9, 38]. While the models we consider in this paper can compute non-computable functions, they still allow us to discuss algorithms in a general sense as they are used e.g. in numerical analysis/scientific computing (cf. [30, 25]). We report on two separate but related investigations.

In the first, we consider extensions of the Blum-Shub-Smale (BSS) machines [1] for computing functions of type $f\colon \mathbb{R}^* \to \mathbb{R}^*$. We explore the strength of a machine that can compute the measure of a BSS-decidable set in a single step. A concrete inspiration for our operations is found in the ν-operator studied by Moschovakis [29]. The ability to compute the measure of a decidable set can also be seen as an analogue to counting classes such as $\sharp\mathrm{P}$ in the context of BSS-computation – except that here, we gain computability-theoretic strength, rather than just efficiency[1].

We also consider adding a command that returns the limit of a BSS-computable converging sequence as a primitive. It turns out that all our enhanced BSS-machines can already compute all real functions computable in the sense of computable analysis. We can therefore use the

[1] This is a different approach to the one taken by Bürgisser and Cucker [13], though.

framework of Weihrauch reducibility [8] to describe the resulting computational strength. The usefulness of Weihrauch reducibility to study algebraic computation models had already been observed in [30]. More generally, the use of topological concepts in this context was pioneered in [18].

Our second line of investigation looks into representations (in the sense of computable analysis) of real function classes, continuing a programme initiated in [35]. For example, if we merely have the information on a function $f\colon \mathbb{R} \to \mathbb{R}$ befitting an integrable function, but we know f to actually be continuous, how complicated is it to obtain the information about f as a continuous function? Classically, we know that any measurable function $f\colon [0,1] \to [0,1]$ is already integrable, but already for Δ^0_2-measurable functions this implication no longer holds computably. This is a connecting point to our exploration of BSS-machines, as BSS-computable functions are Δ^0_2-measurable, and the representation of integrable functions essentially allows us to compute all integrals of this function. Again, Weihrauch reducibility will be the framework we use to describe the complexity of these translations.

Overview of the paper

Our results are collated in three theorems that can be found in Section 3.1 and in the introduction of Section 4 respectively. The paper is structured such that either of the Sections 3 and 4 can be read independently by a reader familiar with the content of Section 2.

The general outline of the paper is as follows: In Section 2 we provide the necessary background on computable analysis and Weihrauch reducibility. A reader already familiar with these topics may skip this part.

Section 3 starts off by recollecting the model of computing with generalized register machines whose registers can hold properly infinite objects. Section 3.1 states our two main theorems about such algebraic models of computation: Each of Theorem 15 and Theorem 16 describes an equivalence class of computational models. The remainder of the section is devoted to their proofs.

Section 4 is about measurability and integrability and states our main result Theorem 24 right away. The theorem says that the Weihrauch degree of various operations that correspond to finding the measure of a set or the integral of a function is lim. The rest of the section explains and proves the individual parts of theorem.

2 Background from computable analysis

Let us start with some computable analysis as its viewpoint is important throughout the paper. Introductionary sources on computable analysis that go way beyond what is the scope of this paper include the seminal textbook [40]. A briefer introduction can be found in [10], an approach in a language close to that of this paper in [31].

In computable analysis computations on abstract structures like the real numbers $\mathbb{R}$, are carried out by means of representations. A **representation** of a set X is a partial surjective mapping $\delta\colon \subseteq \mathbb{N}^{\mathbb{N}} \to X$ from Baire space to that set. A representation may be understood to assign to each element x of the mathematical structure X a set $\delta^{-1}(x) \subseteq \mathbb{N}^{\mathbb{N}}$ of names. Each name of an abstract object $x \in X$ encodes this object by providing on demand information about it. A **represented space** is a pair $\mathbf{X} = (X, \delta_{\mathbf{X}})$ of a set X and a representation $\delta_{\mathbf{X}}\colon \subseteq \mathbb{N}^{\mathbb{N}} \to X$ of that set. For convenience we often replace the two copies of natural numbers in Baire space with countable sets that come with canonical enumerations.

▶ **Example 1** (The Cauchy representation). Let (M, d) be a separable metric space with a distinguished countable dense subset D. The **Cauchy representation** assigns a mapping $p\colon \mathbb{Q}^+ \to D$ as name to $x \in M$ if each $p(\varepsilon)$ is an ε-approximation to x in that $d(x, p(\varepsilon)) \leq \varepsilon$.

The most important instance of this construction is $\mathbb{R}$ with the standard enumeration of the rationals. Other metric spaces in Cauchy representation that we encounter are the continuous functions on the unit interval $\mathcal{C}([0,1])$ and the space $\mathrm{L}_1([0,1])$ of equivalence classes of integrable functions.

Baire space comes with a natural topology and any represented space can be equipped with the final topology of its representation. The notion of **admissibility** of a representation, informally spoken, states that we can identify the represented space with the induced topological space.

Let us consider two examples of represented spaces whose induced topology is not metrizable and that can therefore not be constructed using a Cauchy representation. The first example is a finite non-discrete space that we heavily use:

▶ **Example 2** (Sierpiński space). Sierpiński space $\mathbb{S}$ is the set $\{\top, \bot\}$ equipped with the total representation $\delta_{\mathbb{S}}$ that assigns some $p\colon \mathbb{N} \to \mathbb{B}$ (where $\mathbb{B}$ is the discrete two point space) as name to $\top$ if and only if $\exists n, p(n) = 1$.

The representation of Sierpiński space is admissible and induces the topology where $\{\top\}$ is open but $\{\bot\}$ is not. Thus, the continuous functions from any topological space to Sierpiński space are exactly the characteristic functions of the open sets.

The other example is a representation of the real numbers that provides strictly less information than the Cauchy representation we usually use.

▶ **Example 3** (The lower reals). The lower reals $\mathbb{R}_<$ is the set of real numbers equipped with the representation where a bounded sequence $p\colon \mathbb{N} \to \mathbb{Q}$ of rationals is a name of its supremum $x := \sup\{p(n) \mid n \in \mathbb{N}\}$. Access to a pair of a name of $x \in \mathbb{R}_<$ and one of $-x \in \mathbb{R}_<$ is the same as having access to a name of x in the space $\mathbb{R}$ with the Cauchy representation.

We consider multifunctions between represented spaces as abstractions of computational tasks. A multifunction $f\colon \mathbf{X} \rightrightarrows \mathbf{Y}$ assigns to each $x \in \mathbf{X}$ a set $f(x) \subseteq \mathbf{Y}$ of eligible return values. The domain of f is the set of those x for which $f(x)$ is non-empty and the associated task is to produce from an x and provided that $x \in \mathrm{dom}(f)$ some $y \in f(x)$.

Computability and continuity of multivalued functions between represented spaces is defined via realizers: Some $F\colon \subseteq \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ is a realizer of a multi-valued function $f\colon \mathbf{X} \rightrightarrows \mathbf{Y}$ if it carries names of the input to names of eligible return values, i.e. $\delta_{\mathbf{X}}(p) = x$ implies $F(p)$ is defined and $\delta_{\mathbf{Y}}(F(p)) \in f(x)$. A multivalued function is called **computable** if it has a computable realizer and **continuously realizable** if it has a continuous realizer. Admissibility of the target space implies that a function is topologically continuous if and only if it is continuously realizable. Computability on Baire space is defined as existence of an oracle machine that computes the return-value whenever the input is taken as oracle.

A representation δ is called **computably translatable** to another representation δ' of the same set X, if the identity function is computable as a function from (X, δ) to (X, δ'). Unfolded this means that there exists a **translation**, i.e. a computable $T\colon \subseteq \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ such that whenever p is a name of x in the first representation, $T(p)$ is a name of x in the second representation. Intuitively, existence of a computable translation means that a name in the input representation provides at least as much information about the object as a name in the output representation. If computable translations in both directions exist the

representations are called **equivalent** and the identity function is an isomorphism between the spaces. Moreover we say that a representation is **minimal with some property** if any other representation with this property can be computably translated to it.

Finally, a metric space with a distinguished dense sequence is called a **computable metric space** if the metric is computable with respect to the Cauchy representation on itself and on the reals. All metric spaces we encounter in this paper are computable metric spaces.

2.1 Products and exponentials of represented spaces

Given represented spaces $\mathbf{X}$ and $\mathbf{Y}$ there are standard ways to construct new represented spaces. To equip the Cartesian product of the underlying spaces with a representation fix a standard pairing function $\langle\cdot,\cdot\rangle$ of the natural numbers and lift this pairing function to Baire space by $\langle p,q\rangle(n) := \langle p(n), q(n)\rangle$. A name of $(x,y) \in \mathbf{X}\times\mathbf{Y}$ is a pair of names of x and y respectively. To each $A \subseteq \mathbf{X}$ we assign a represented space $\mathbf{A_X}$ by equipping A with the corestriction of the representation of $\mathbf{X}$ to A.

Finally, we use the function space construction of computable analysis. Namely for represented spaces $\mathbf{X}$ and $\mathbf{Y}$ the space $\mathcal{C}(\mathbf{X},\mathbf{Y})$ of continuously realizable functions.

If $\mathbf{Y}$ is admissible, the continuously realizable functions are exactly the continuous functions. The computable elements of $\mathcal{C}(\mathbf{X},\mathbf{Y})$ are exactly the computable functions. The function space representation is minimal with the property that evaluation is computable.

The representation is not easy to work with but there often exist simpler equivalent representations.

▶ **Example 4** (Concrete representations for spaces of functions)**.** The computable Weierstraß approximation theorem can be understood to say that $\mathcal{C}([0,1]) \simeq \mathcal{C}([0,1],\mathbb{R})$. Here, $\mathcal{C}([0,1],\mathbb{R})$ is the space of continuous functions with the function space representation while $\mathcal{C}([0,1])$ has the same underlying set but in Cauchy representation with respect to the supremum norm and the rational polynomials as dense subset. More specifically the metric is given by $d(f,g) := \|f-g\|_\infty$, where $\|f\|_\infty := \sup\{|f(x)| \mid x \in [0,1]\}$ is the supremum norm.

Finally, for a represented space $\mathbf{X}$ we use the space $\mathbf{X}^*$ of finite lists over $\mathbf{X}$. A name of such a finite word takes the same kind of questions that one can ask about elements of $\mathbf{X}$ but returns a finite list of answers for each of the elements of the list.

2.2 Weihrauch reducibility and Weihrauch degrees

A reasonable notion of comparison of computational tasks f and g is to ask whether a solution to f can be specified if – in addition to computable operations – an arbitrary solution to g may be involved once. Weihrauch reductions make this idea formal. Recall that $\langle p,q\rangle$ is the pairing of elements p and q of Baire space.

▶ **Definition 5.** *Let f and g be multivalued functions between represented spaces. A **Weihrauch reduction** of f to g is a pair of computable functions $H,K\colon \mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}$ called **pre-** and **post-processor** such that for any realizer G of g the function $p \mapsto K(\langle G(H(p)),p\rangle)$ is a realizer of f.*

We write $f \leq_\mathrm{W} g$ if there exists a Weihrauch reduction and use $f \equiv_\mathrm{W} g$ as abbreviation for $f \leq_\mathrm{W} g \wedge g \leq_\mathrm{W} f$. The **Weihrauch degree** of f is its equivalence class under Weihrauch reductions.

For additional information on the theory of Weihrauch degrees we refer the reader to [8].

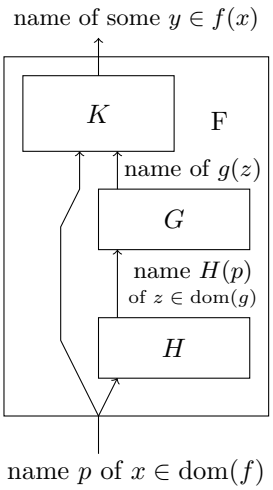

Figure 1 $f \leq_{\mathrm{W}} g$.

▶ **Example 6** (Weihrauch degrees)**.** The following Weihrauch degrees are important to us:

LPO: By $\mathrm{LPO}\colon \mathbb{N}^{\mathbb{N}} \to \mathbb{B}$ we denote the characteristic map of the singleton set containing the constant zero function and its Weihrauch degree. The name is short for 'lesser principle of omniscience' and originates from constructive mathematics, where this principle often serves as a Brouwerian counterexample to show that statements are not constructively provable. Other representatives of LPO include the mapping from $\mathbb{S}$ to $\mathbb{B}$ that sends $\top$ to 1 and $\bot$ to 0 and checking equality of real numbers.

lim and $\mathrm{C}_{\mathbb{N}}$: For a represented space $\mathbf{X}$ consider the multivalued function $\lim_{\mathbf{X}}\colon \mathbf{X}^{\mathbb{N}} \rightrightarrows \mathbf{X}$ that sends a sequence $(x_n)_{n\in\mathbb{N}}$ to the set $\{x \mid \lim_{n\to\infty} x_n = x\}$ of its limit points. It holds that $\lim_{\mathbb{R}} \equiv_{\mathrm{W}} \lim_{\mathbb{N}^{\mathbb{N}}} \equiv_{\mathrm{W}} \lim_{[0,1]}$, and we just write lim for this Weihrauch degree. We also use the degree of $\lim_{\mathbb{N}}$, which is equivalent to closed choice on the natural numbers, $\mathrm{C}_{\mathbb{N}}$, to be discussed in Section 2.3.

There are several ways to construct new Weihrauch degrees from known ones, let us go through some that particularly important for our causes. First consider the parallelization, where the new task is to solve the original task a countable number of times in parallel. For a given multivalued function $f\colon \mathbf{X} \rightrightarrows \mathbf{Y}$ let $\hat{f}\colon \mathbf{X}^{\mathbb{N}} \rightrightarrows \mathbf{Y}^{\mathbb{N}}$ be the operation of applying f pointwise, i.e. $\hat{f}((x_n)_{n\in\mathbb{N}}) := \{(y_n)_{n\in\mathbb{N}} \mid \forall n \in \mathbb{N}\colon y_n \in f(x_n)\}$. One verifies that $f \leq_{\mathrm{W}} g$ implies $\hat{f} \leq_{\mathrm{W}} \hat{g}$, and hence the operation lifts to Weihrauch degrees.

▶ **Lemma 7** (Parallelization of Weihrauch degrees [6])**.** *The following equivalences hold:*

$$\widehat{LPO} \equiv_W \widehat{\mathrm{C}_{\mathbb{N}}} \equiv_W \widehat{\lim} \equiv_W \lim.$$

There are some more complicated constructions that we also need, namely the sequential composition and most prominently the diamond operator where a task can be used sequentially as many times as needed. However, for a proper discussion of this operator we need register machines and we thus postpone it to Section 3. An additional operations is the finite parallel execution f^* where the task is given any integer n to solve the task corresponding to f in parallel n-times.

2.3 Spaces of sets and choice principles

Closed choice on the natural numbers, denoted by $\mathrm{C}_{\mathbb{N}}$, is the task of selecting an element of a closed set of natural numbers. To make this formal let us first recall how to introduce spaces of open and closed subsets of represented spaces.

Recall from Example 2 that the functions from a topological space to Sierpiński space $\mathbb{S}$ are exactly the characteristic functions of the open sets. In analogy to this one may generalize from topological spaces to an arbitrary represented space $\mathbf{X}$ and consider $\mathcal{O}(\mathbf{X}) := \mathcal{C}(\mathbf{X}, \mathbb{S})$ the space of open subsets of $\mathbf{X}$. This means that for an arbitrary represented space $\mathbf{X}$ we consider a subset $A \subseteq \mathbf{X}$ to be open if its characteristic function $\chi_A \colon \mathbf{X} \to \mathbb{S}$ is continuously realizable. Similarly, the represented space $\mathcal{A}(\mathbf{X})$ of closed sets consists of the complements of open sets and names them accordingly.

▶ **Lemma 8** ([12]). *If $\mathbf{X}$ is a metric space in the Cauchy representation, then the information that a name of $A \in \mathcal{O}(\mathbf{X})$ provides is exactly a sequence of balls of rational radius around elements of the dense sequence that exhausts A.*

In particular, for a computable metric space the computable elements of $\mathcal{O}(\mathbf{X})$ are exactly those open sets that are computably enumerable in the usual sense. As an example and for later use recall that $\mathbb{R}_<$ denotes the lower reals from Example 3.

▶ **Example 9** (Lower approximations to the volume of open sets)**.** The Lebesgue measure is computable as function $\lambda \colon \mathcal{O}([0,1]) \to \mathbb{R}_<$. This is because according to the previous lemma a name of an open set is a sequence of rational intervals that exhaust the set.

Closed choice for a space $\mathbf{X}$ is the multivalued function $\mathrm{C}_{\mathbf{X}} \colon \mathcal{A}(\mathbf{X}) \rightrightarrows \mathbf{X}$ where the eligible return values are the elements of the closed set provided as input. The domain of $\mathrm{C}_{\mathbf{X}}$ consists of the non-empty closed sets. Closed sets are encoded as positive information about their complement, thus $\mathrm{C}_{\mathbf{X}}$ is usually discontinuous and in particular incomputable. For instance $\mathrm{C}_{\mathbb{N}}$ can be reformulated as finding a number not occurring in an enumeration.

We also need the next higher level of complexity, namely the spaces of Σ^0_2 and Π^0_2 sets. The jump in complexity can be done using $\lim_{\mathbb{N}^{\mathbb{N}}}$ as a jump operator [15] respectively computable endofunctor [34, 32]. Namely, given a represented space $\mathbf{X} = (X, \delta_{\mathbf{X}})$ define the lim-jump of $\mathbf{X}$ to be the represented space that has the same underlying set but with the representation $\delta'_{\mathbf{X}} := \delta \circ \lim_{\mathbb{N}^{\mathbb{N}}} \circ \delta_{(\mathbb{N}^{\mathbb{N}})^{\mathbb{N}}}$. A function $f \colon \mathbf{X} \to \mathbf{Y}$ is called lim-computable resp. lim-continuous if it is computable resp. continuous as a function from $\mathbf{X}$ to the lim-jump of $\mathbf{Y}$. If $\mathbf{Y}$ is $\mathbb{R}$ or $\mathbb{R}^*$, this is the same as being (effectively) Baire class 1. The notion of lim-computability can also be expressed through Weihrauch degrees: A function $f \colon \mathbf{X} \to \mathbf{Y}$ is lim-computable if and only if $f \leq_{\mathrm{W}} \lim$.

We plan on replacing Sierpiński space $\mathbb{S}$ in the definitions of open and closed sets above by its lim-jump. Before we state these definitions let us specify a space $\mathbb{S}'$ that is isomorphic to the lim-jump of $\mathbb{S}$ but has a more approachable representation and convenient naming of its elements. Equip the set $\{\top', \bot'\}$ with the total representation $\delta_{\mathbb{S}'}$ that assigns $p \colon \mathbb{N} \to \mathbb{B}$ as name to $\top'$ if $p(n) = 0$ only for finitely many n.

Now that we have fixed $\mathbb{S}'$ we can set $\Sigma^0_2(\mathbf{X}) := \mathcal{C}(\mathbf{X}, \mathbb{S}')$ and let the names of an element of $\Pi^0_2(\mathbf{X})$ be the $\Sigma^0_2(\mathbf{X})$ names of its complement. As $\mathbb{S}'$ is isomorphic to the lim-jump of $\mathbb{S}$ this means that for an arbitrary represented space $\mathbf{X}$ we consider a subset $A \subseteq \mathbf{X}$ to be a Σ^0_2-set if its characteristic function $\chi_A \colon \mathbf{X} \to \mathbb{S}$ is lim-continuous. Here we follow the approach of synthetic descriptive set theory suggested in [34]. These definitions recreate more familiar ones that are only available in special cases. For instance: recall that the Π^0_2-subsets of a metric space are the countable intersections of open subsets. Following Brattka [5] one would define the names of a Π^0_2-set A to be the union of all name sets of sequences $(U_n)_{n\in\mathbb{N}} \in \mathcal{O}(\mathbf{X})^{\mathbb{N}}$ such that $A = \bigcap_{n\in\mathbb{N}} U_n$.

▶ **Lemma 10** (proven in [24]). *The representation of the Π^0_2-sets in the sense of Brattka as presented above can be computably translated to the representation of $\Pi^0_2(\mathbf{X})$. If $\mathbf{X}$ is a computable metric space, then a translation in the inverse direction is also possible.*

Using $\Pi^0_2(\mathbf{X})$ we can introduce another important Weihrauch degree:

▶ **Definition 11** (Π^0_2-choice). *Let $\Pi^0_2\mathrm{C}_{\mathbf{X}} : \Pi^0_2(\mathbf{X}) \rightrightarrows \mathbf{X}$ return the elements of A on input A.*

The domain of the multifunction $\Pi^0_2\mathrm{C}_{\mathbf{X}}$ are the non-empty Π^0_2-subsets of $\mathbf{X}$. Some other representatives of the Weihrauch degree $\Pi^0_2\mathrm{C}_{\mathbb{N}}$ can for instance be found in [7].

3 Algebraic models of computation and the diamond operator

Computation on algebraic structures has been studied in various ways, e.g. [26, 39]. The most influential approach is by Blum, Shub and Smale [2] introducing the algebraic model of computation on the reals that came to be named BSS-machines after the authors. We particularly wish to highlight Moschovakis' [29] as initial inspiration for our considerations. We use the notion of a register machine over some algebraic structure, as defined in [30] by following Gaßner [19, 20, 21] and Tavana and Weihrauch [38].

For us, an algebraic structure is a tuple $\mathfrak{A} = (A, f_1, f_2, \ldots, T_1, T_2, \ldots)$, where A is a set, each f_i is a (partial) function of type $f_i : \subseteq A^{k_i} \to A$, and each T_i is a relation of type $T_i \subseteq A^{l_i}$. A $\mathfrak{A}$-register machine computes functions of type $g : \subseteq A^* \to A^*$. It has registers $(Z_i)_{i\in\mathbb{N}}$ holding elements of A, and index registers $(I_n)_{n\in\mathbb{N}}$ holding natural numbers. Programs are finite lists of commands, consisting of:

- standard register machine operations on the index registers
- copying the value of the register Z_{I_1} indexed by I_1 into Z_{I_0}
- applying some f_i to the values contained in $Z_1, \ldots, Z_{k_i}$ and writing the result into Z_0
- jumping to a line in the program if the content of $Z_0, \ldots, Z_{l_i-1}$ is an element of T_i
- HALT, in which case the values in the registers $Z_0, \ldots, Z_{I_0}$ constitute the return value

At the start of the computation the register I_0 contains the length of the input, all other I_i start at 0. The input is in $Z_1, \ldots, Z_{I_0}$, all other Z_i contain some fixed value $a_0 \in A$. If the program either fails to halt on some input – which in particular happens if it invokes a partial function on some values outside its domain – the computed function is undefined on these values.

▶ **Definition 12.** *A BSS-machine is a $(\mathbb{R}; 0, 1, (q)_{q\in\mathbb{Q}}, +, \times, <)$-register machine.*

We can add any constant function $c : \mathbb{R}^0 \to \mathbb{R}$ taking a computable value without it making a difference anywhere in our paper, although this obviously increases the computational power whenever we add irrational constants. For a more detailed introduction, see [22].

As long as our signatures are finite, with the potential exception of constants, we can code $\mathfrak{A}$-register machine programs as inputs for an $\mathfrak{A}$-register machine using the length of the input for the discrete part, and elements of A for the constants. This in particular ensures the existence of universal machines, as these codes can be effectively decoded.

In computable analysis register machines whose signature only contains computable operations are a popular tool for making proofs more accessible [38]. Weihrauch reducibility has lead to a widespread use of register machines with incomputable signature:

▶ **Example 13** (The diamond operator)**.** The Weihrauch degree $f^\diamond$ captures all tasks that we can solve in a finite computation with oracle access to f. A representative of $f^\diamond$ can then be given as a multifunction that takes as input an index of a register machine M together with an input x for M, and outputs whatever M would output on x. An equivalent definition in terms of reduction games was given by Jockusch and Hirschfeldt [27].

A proof that $\mathrm{C}_{\mathbb{N}} \equiv_{\mathrm{W}} \mathrm{LPO}^\diamond$ was given in [30]. Recall that LPO is the Weihrauch degree of deciding equality of real numbers. As BSS-machines are register machines with the capability to do branching over equalities of reals this equivalence provides a close link between closed choice on the natural numbers to the power of BSS-machines.

The composition of Weihrauch degrees is an operation related to the diamond operator that we need only in the passing. The Weihrauch degree $f \star g$ essentially means "do something computable, then invoke g, do some more computation, invoke f and after some more computation return an answer". We have $f \star g \equiv_{\mathrm{W}} \max_{\leq_{\mathrm{W}}}\{F \circ G \mid F \leq_{\mathrm{W}} f \wedge G \leq_{\mathrm{W}} g\}$, where the maximum is only taken over f and g that are composable. It is not obvious that the maximum exists, but a concrete construction of a representative can be found in [11]. The diamond operation corresponds to the closure under composition in the sense that $\mathrm{id}_{\mathbb{N}^{\mathbb{N}}} \leq_{\mathrm{W}} f = f \star f$ is equivalent to $f = f^\diamond$ as proven by Westrick [41].

3.1 Enhancing BSS-machines and statement of our results

In this section we consider functions $f\colon \mathbb{R}^* \to \mathbb{R}^*$, where $\mathbb{R}^*$ is the space of finite sequences of real numbers as introduced at the very end of Section 2.1. We compare several models of computation by register machines that may produce incomputable functions. The models we consider fall into two classes: One adapts BSS-machines with primitive operations for computing measures and the other adds capabilities of computing limits.

Starting from a BSS-machine, we add the ability to compute the Lebesgue measure of a given BSS-decidable subset of $[0,1]^d$ to obtain BSS+λ-machines. More specifically the run of a BSS-machine proceed as follows in evaluating the Lebesgue measure (for more details see the appendix):

▶ **Definition 14.** *If a BSS+λ-machine reaches a λ-command, the content of I_0 is interpreted as a Gödel-number of a BSS-machine M using k constants. The contents $a_1, \ldots, a_k$ of $Z_1, \ldots, Z_k$ are used as the values for the k constants and the content of I_1 as the dimension n of the input. If $M(a_1, \ldots, a_k)$ halts for every input $x \in [0,1]^n$ and outputs either 0 or 1, then we replace the content of Z_0 with the Lebesgue measure of the set $\{x \in [0,1]^n \mid M(a_1, \ldots, a_k)(x) = 1\}$ and let the computation continue. If M is not as desired we do not modify the state so that the computation loops on this command.*

As it is also meaningful to talk about indices of BSS+λ-machines, we may iterate this process once. That is, we use BSS+λ+λ-machines that have an additional primitive operator for computing the Lebesgue measure of a set decidable by a BSS+λ-machine. One may produce a more formal definition by replacement of the type of index used in the previous definition.

Motivated by our proofs we also consider machines with access to arbitrary (partial) computable functions. We call such machines BSS+Comp-machines and as most of the signature of BSS-machines is computable these machines can just use tests for strict inequality of real numbers in addition to the computable functions on the reals. We add to these machines the capability of computing measure very similar to how this was done for BSS- and BSS+λ-machines. There is a small issue to address here as the infinite signature of BSS+Comp-machines makes talking about programs slightly more complicated. However,

we may replace the infinite signature with a finite one by use of a universal computable function $u\colon\subseteq\mathbb{R}\to\mathbb{R}$. This accounts for all unary computable functions and by the effective Kolmogorov superposition theorem [4], together with addition this already suffices to construct all computable functions of arbitrary finite arity. By adding a primitive for the measure of a set decidable in this setting (which is just a Δ^0_2-set), we obtain the BSS+Comp+λ-machines.

Our second class of models adds abilities to compute certain limits. The operator c-lim (for controlled limit) maps a program for a BSS-machine that computes a sequence of real numbers x_i with $|x_i - x_j| < 2^{-\min\{i,j\}}$ to the limit of this sequence. The operator u-lim (uncontrolled limit) accepts a program computing an arbitrary converging sequence of real numbers, and also outputs the limit. These operators correspond to the strongly and weakly analytic machines going back to Chadzelek and Hotz [14]. However, in our model the c-lim- and u-lim-commands can be used multiple times throughout the computation. Analytic machines only allow one application at the end of the computation. Thus, we consider the closure of what strongly/weakly analytic machines compute under composition here. We denote the respective machine models BSS+c-lim and BSS+u-lim and the details of what the commands do are similar to what was laid out in Definition 14.

In [30] it was shown that BSS-machines and strongly analytic machines can be characterized by a complete Weihrauch degree: every function computable in that model is Weihrauch reducible to the complete degree and the complete degree has a representative computable in the model. For the models we consider here we obtain the stronger characterization that the computed functions are exactly those functions from a lower cone in the Weihrauch degrees that are of suitable type. All of the models mentioned above coalesce into just two equivalence classes:

▶ **Theorem 15.** *For a function $f\colon\subseteq\mathbb{R}^*\to\mathbb{R}^*$ the following are equivalent:*

1. $f\leq_W \Pi^0_2\mathrm{C}_{\mathbb{N}}$
2. *f is computable by a BSS+λ-machine.*
3. *f is computable by a BSS+c-lim-machine.*
4. *f is computable by a BSS+Comp-machine with oracle access to the BSS-Halting problem*[2].
5. *There is a uniform sequence of Π^0_2-sets $(A_n)_{n\in\mathbb{N}}$ such that $\mathrm{dom}(f)=\bigcup_{n\in\mathbb{N}}A_n$ and each $f|_{A_n}$ is computable.*

Anticipating the notion of being piece-wise continuous as introduced in Section 4.3 the last item may be reformulated as f being a computable element of the Π^0_2-piece-wise continuous functions. Another equivalent model that we omit here is to be computable by a weakly non-deterministic Type-2 machine with advice space $\mathbb{N}$ as introduced by Ziegler [42].

▶ **Theorem 16.** *For a function $f\colon\subseteq\mathbb{R}^*\to\mathbb{R}^*$ the following are equivalent:*

1. $f\leq_W \lim^{\diamond}$
2. *f is computable by a BSS+λ+λ-machine.*
3. *f is computable by a BSS+u-lim-machine.*
4. *f is computable by a BSS+Comp+λ-machine.*
5. *There is a uniform cover $\bigcup_{n\in\mathbb{N}}A_n=\mathrm{dom}(f)$ where A_n is Π^0_n, and $f|_{A_n}$ is of effective Baire class n.*

From known properties of the Weihrauch degrees we can now for instance draw conclusions about how far computability is preserved point-wise by functions computed in these models:

[2] Note that it does not matter whether we use the Halting problem for additive machines or the full strength, or even just $\mathbb{Q}$ as an oracle.

▶ **Corollary 17.** *A BSS+λ-machine with computable constants on computable input either diverges or produces a computable value. A BSS+λ+λ-machine with computable constants can, on computable input, produce outputs in any finite level of the arithmetical hierarchy.*

Proof. To see the first claim, we observe that $\Pi^0_2\mathrm{C}_{\mathbb{N}}$ returns natural numbers. Thus $f \leq_{\mathrm{W}} \Pi^0_2\mathrm{C}_{\mathbb{N}}$ implies that f can only take computable values on computable inputs. For the second claim, observe that one the one hand if $p \in \mathbb{B}^{\mathbb{N}}$ is arithmetical, then the constant function $x \mapsto p\colon \mathbb{R}^* \to \mathbb{R}^*$ (where we identify Cantor space $\mathbb{B}^{\mathbb{N}}$ and the Cantor middle third set) is Weihrauch reducible to $\lim^\diamond$. Conversely, since every computation of $\lim^\diamond$-machine involves only finitely many limits, its output on a computable input is always arithmetical. ◀

3.2 Measure, controlled limits and the Weihrauch degree of sorting

First, we show how a controlled limit can be reduced to the measure of a decidable set. Note that strongly analytic machines can compute all computable functions $f\colon \mathbb{R}^* \to \mathbb{R}^*$, so the following in particular implies that BSS+λ-machines can do the same.

▶ **Lemma 18.** *From a strongly analytic machine with parameters $(a_1, \ldots, a_d)$ that computes a function $f\colon \mathbb{R}^d \to [0,1]$ we can compute a BSS-machine (also using $(a_1, \ldots, a_d)$) that decides some subset $A_{a_1,\ldots,a_d} \subseteq [0,1]$ such that $f(a_1, \ldots, a_d) = \lambda(A_{a_1,\ldots,a_d})$.*

Proof. We partition the unit interval into $\{0\}$ and $\left((2/3)^{i+1}, (2/3)^i\right]$ for $i \in \mathbb{N}$. The set $A_{a_1,\ldots,a_d}$ will be of the form $A_{a_1,\ldots,a_d} = \bigcup_{i \in B_{a_1,\ldots,a_d}} \left((2/3)^{i+1}, (2/3)^i\right]$ for some $B_{a_1,\ldots,a_d} \subseteq \mathbb{N}$. A BSS-machine can decide $A_{a_1,\ldots,a_d}$ if and only if it can decide $B_{a_1,\ldots,a_d}$.

A BSS-machine can simulate the strongly analytic machine for any finite amount of time. At some point, it can pick a true case out of $f(a_1, \ldots, a_d) < 2/3$ and $1/3 < f(a_1, \ldots, a_d)$, and decides $0 \notin B_{a_1,\ldots,a_d}$ and $c_0 = 0$ in the former, and $0 \in B_{a_1,\ldots,a_d}$ and $c_0 = 1/3$ in the latter case.

In the next step, $f(a_1, \ldots, a_d) - c_0 \in [0, 2/3]$, and the BSS-machine can simulate the strongly analytic machine until it determines a true case amongst $f(a_1, \ldots, a_d) - c_0 < (2/3)^2$ and $\frac{2}{3^2} < f(a_1, \ldots, a_d) - c_0$, and then sets $1 \notin B_{a_1,\ldots,a_d}$ and $c_1 = c_0$ in the former, and $1 \in B_{a_1,\ldots,a_d}$ and $c_1 = c_0 + 2/9$. The BSS-machine keeps iterating this process until it determines whether or not $i \in B_{a_1,\ldots,a_d}$ for the i it needs to know about to decide membership of the input in $A_{a_1,\ldots,a_d}$. It is straight-forward calculation that $f(a_1, \ldots, a_d) = \lambda(A_{a_1,\ldots,a_d})$. ◀

In the next step, we obtain a Weihrauch upper bound for computing measures of BSS-decidable sets. This involves the Weihrauch degree Sort of the task of sorting infinite binary sequences. Here, sorting means to return the infinite binary sequence 0^n1^ω if the input sequence has exactly n zeros and 0^ω if it has infinitely many zeros. It was shown in [30] that Sort^* characterizes the strength of strongly analytic machines.

▶ **Lemma 19.** *The task **given a BSS-machine M with constants $(a_1, \ldots, a_n)$ that decides a set $A \subseteq [0,1]^d$, compute** $\lambda(A)$ is Weihrauch-reducible to LPO$\star\,\mathrm{Sort}^n$.*

Proof. Using LPO$\star\,\mathrm{Sort}^n$ we can decide whether or not the $(a_1, \ldots, a_n)$ are algebraically independent, and if they are dependent, compute their minimal polynomial (as shown in [30]). Any test M makes on input $(x_1, \ldots, x_d)$ can be seen as asking whether a polynomial $P(a_1, \ldots, a_n, x_1, \ldots, x_d)$ is negative, positive, or 0. Knowing the minimal polynomial of $(a_1, \ldots, a_n)$ (or that they are algebraically independent) lets us decide whether P is 0 independent of the values of the x_i. If yes, we can eliminate the test for P from M. If no, then the set of $(x_1, \ldots, x_d)$ causing P to be 0 is of measure 0, hence can be safely ignored for determining $\lambda(A)$.

Once we do this procedure for all tests in M, we obtain two open sets U_1, U_2 (each as union of the open sets linked to a computation path where tests are yielding positive or negative answers) such that $U_1 \subseteq A$, $U_2 \subseteq [0,1]^d \setminus A$ and $\lambda([0,1]^d \setminus (U_1 \cup U_2)) = 0$. As the measure of open sets is lower-semicomputable, this allows us to compute $\lambda(A)$. ◀

The two preceding lemmas, together with the classification of strongly analytic machines from [30] already tell us that when allowing arbitrary use of the principle in a finite computation, then computing measures of BSS-computable sets, limits of fast converging BSS-computable sequences or solving Sort all yields the computational strength of $\mathrm{Sort}^\diamond$.

▶ **Proposition 20.** $\mathrm{Sort}^\diamond \equiv_W \Pi^0_2\mathrm{C}_\mathbb{N}$.

Proof. We find that isInfinite $\leq_\mathrm{W}$ Sort $\star$ Sort $\leq_\mathrm{W}$ $\mathrm{Sort}^\diamond$, and by results from [7], also $\mathrm{isInfinite}^\diamond \equiv_\mathrm{W} \Pi^0_2\mathrm{C}_\mathbb{N} \equiv_\mathrm{W} \Pi^0_2\mathrm{C}^\diamond_\mathbb{N}$. This shows that $\Pi^0_2\mathrm{C}_\mathbb{N} \leq_\mathrm{W} \mathrm{Sort}^\diamond$. For the other direction, we just point out that $\mathrm{Sort} \leq_\mathrm{W} \Pi^0_2\mathrm{C}_\mathbb{N}$ is immediate. ◀

We have now gathered all auxiliary results we need for proving the first of our theorems.

Proof of Theorem 15. **1. ⇒ 3.** It was shown in [30] that a strongly analytic machine can compute a representative of the Weihrauch degree Sort. It follows that a BSS+c-lim-machine can simulate a $\mathrm{Sort}^\diamond$-computation on any valid types. Now, $\mathrm{Sort}^\diamond \equiv_\mathrm{W} \Pi^0_2\mathrm{C}_\mathbb{N}$ by Proposition 20, so that our claim follows.

3. ⇒ 2. By Lemma 18, we can replace each c-lim-command by a λ-command.

2. ⇒ 1. By Lemma 19 a $\mathrm{Sort}^\diamond$-machine can simulate a BSS+λ-machine. Proposition 20 tells us that $\mathrm{Sort}^\diamond \equiv_\mathrm{W} \Pi^0_2\mathrm{C}_\mathbb{N}$.

1. ⇔ 4. This follows from [30, Theorem 21] and $\Pi^0_2\mathrm{C}_\mathbb{N} \equiv_\mathrm{W} \mathrm{isInfinite}^\diamond$.

1. ⇔ 5. It was observed by Nobrega [17] that being Weihrauch reducible to $\Pi^0_2\mathrm{C}_\mathbb{N}$ correspondents to a Π^0_2-cover of the domain such that all restrictions of the function to a piece are computable. Essentially, the pieces just are the inputs that lead to some number $n \in \mathbb{N}$ being a valid output of $\Pi^0_2\mathrm{C}_\mathbb{N}$ on the instance it is queried on. There is a minor issue here regarding whether we have a cover of the set of names of inputs, or directly of the inputs. For $\mathbb{R}^*$ as domain this makes no difference, as explained in [33]. ◀

3.3 Iterating measure and computable functions as supplement

We can now ask what happens if we allow to nest the λ-operator once.

▶ **Proposition 21.** *A function $f\colon \mathbb{R}^* \to \mathbb{R}^*$ is computable by a BSS+λ+λ-machine if and only if $f \leq_W \lim^\diamond$.*

Proof. A representative of the degree of lim is $\mathrm{id}\colon [0,1]_< \to [0,1]$, the identity from the lower reals in the unit interval to the unit interval. From $b \in [0,1]_<$ we can compute $[0,b) \in \mathcal{O}([0,1])$, and the characteristic function of an open set is computable relative to LPO, hence in particular relative to $\Pi^0_2\mathrm{C}_\mathbb{N}$. This establishes together with Theorem 15 that a BSS+λ+λ-machine can compute everything Weihrauch-reducible to lim, and subsequently $\lim^\diamond$.

For the other direction, assume that the characteristic function χ_A of $A \subseteq [0,1]^d$ is computable by a BSS+λ-machine. By Theorem 15 it follows that $\chi_A \leq_\mathrm{W} \Pi^0_2\mathrm{C}_\mathbb{N}$. This in turn tells us that there are Π^0_2-sets $(B_n)_{n\in\mathbb{N}}$ and $(C_n)_{n\in\mathbb{N}}$ such that $A = \bigcup_{n\in\mathbb{N}} B_n$ and $[0,1]^d \setminus A = \bigcup_{n\in\mathbb{N}} C_n$. In particular, A is a Δ^0_3-set. Using $\lim \star \lim \star \lim$ we can compute the measure of a Σ^0_3-set as a real number, and thus the claim follows. ◀

Note that for the first direction in the theorem above we only used that BSS+λ-machines can compute all computable functions, and can decide all computable open sets. This can already be facilitated by BSS+Comp-machines that can be simulated by BSS+λ-machines:

▶ **Corollary 22.** *BSS+Comp+λ- and BSS+λ+λ-machines can compute the same functions.*

The following is somewhat more general than we need for our theorem.

▶ **Proposition 23.** *The following are equivalent for $f\colon \mathbf{X} \rightrightarrows \mathbf{Y}$, where $\mathbf{X}$ is a effectively countably based space:*

1. *$f \leq_W \lim^\diamond$.*
2. *There is a uniform cover $\bigcup_{n\in\mathbb{N}} A_n$ of $\mathbf{X}$ where A_n is Π^0_n, and $f|_{A_n}$ is of effective Baire class n.*

Proof. Let $f \leq_{\mathrm{W}} \lim^\diamond$. We point out that using lim, we can chose a canonic name for each point in an effectively countably-based space. Thus, in the $\lim^\diamond$-computation, we can assume that all names of the same point proceed in the same way through the computation tree generated by the generalized register machine in the definition of $^\diamond$. The computation tree has countably many leaves where the computation can terminate and provide an output. If the path to a particular leaf is using n invocations of lim, then the set of inputs leading to that leaf is a Π^0_{n+1}-set, and we can obtain this uniformly. Every content of a register at that moment (thus in particular the output) can be obtained as an effectively Baire class n function. By padding with empty sets if necessary, we obtain the desired sequence from an enumeration of the leaves.

Conversely, if we have a uniform cover $\bigcup_{n\in\mathbb{N}} A_n$ of $\mathbf{X}$ where A_n is Π^0_n, then a $\lim^\diamond$-computation can on input $x \in \mathbf{X}$ find some n such that $x \in A_n$. Subsequently, a $\lim^\diamond$-computation can simulate any effective Baire class n function. ◀

Note that in particular the pre-computable Quasi-Polish space from [16] are effectively countably-based. Finally, we prove our second theorem.

Proof of Theorem 16.

1. $\Leftrightarrow$ 2., 2. $\Leftrightarrow$ 4. and 1. $\Leftrightarrow$ 5. These follow from the Propositions 21 and 23 and Corollary 22 respectively.

1. $\Leftrightarrow$ 3. As a BSS+u-lim-machine can compute all computable partial functions on $\mathbb{R}^*$, the only difference between a BSS+u-lim and a $\lim^\diamond$-machine is what are appropriate input and output types. ◀

4 Measurability, Integrability and Weihrauch degrees

Classically, any measurable function $f\colon [0,1] \to [0,1]$ is integrable. Computably, this fails to be true. Rather than dealing with all Borel measurable functions, we restrict our attention to the lowest non-trivial complexity and explore the Δ^0_2-measurable functions. For metric spaces $\mathbf{X}$ and $\mathbf{Y}$, the Jayne-Rogers Theorem gives an alternate description of the Δ^0_2-measurable functions as piece-wise continuous functions [33]. In our setting the piece-wise continuous functions can be identified with the space $\mathcal{C}(\mathbf{X}, \mathbf{Y}^\nabla)$ of $\lim_\Delta$-continuous functions. Below, we go into more detail about these spaces and also about the space $\mathrm{L}_1([0,1])$ of integrable functions. Before we do this recall that $\mathbb{R}_<$ is the space of real numbers represented as suprema of sequences of rationals and let us state the main results that we prove.

▶ **Theorem 24.** *The following maps are Weihrauch equivalent to* lim*:*

1. *Evaluating a continuous function from its description as an integrable function. That is, the inverse of the inclusion of* $\mathcal{C}([0,1],[0,1])$ *into* $\mathrm{L}_1([0,1])$.
2. *The Lebesgue measure as function on* $\Delta^0_2([0,1])$ *with values either in* $\mathbb{R}$ *or in* $\mathbb{R}_<$*, i.e.*

$$\lambda\colon \Delta^0_2([0,1]) \to \mathbb{R} \quad \text{and} \quad \lambda\colon \Delta^0_2([0,1]) \to \mathbb{R}_<.$$

3. *Integration taking a piece-wise continuous function on the unit interval and returning an element of either* $\mathbb{R}$ *or* $\mathbb{R}_<$*, namely the functions*

$$\int\colon \mathcal{C}([0,1],[0,1]^\nabla) \to \mathbb{R} \quad \text{and} \quad \int\colon \mathcal{C}([0,1],[0,1]^\nabla) \to \mathbb{R}_<.$$

4. *Translating from piece-wise continuous functions to integrable functions, i.e. the inclusion of* $\mathcal{C}([0,1],[0,1]^\nabla)$ *into* $\mathrm{L}_1([0,1])$.

4.1 Integrable functions, $\mathrm{L}_1([0,1])$ and continuous functions

Let us first discuss what the represented space of integrable functions looks like. Recall that strictly speaking $\mathrm{L}_1([0,1])$ is not a space of functions but instead its elements are equivalence classes of such. For $f\colon [0,1] \to \mathbb{R}$ integrable in the sense that $\int_0^1 |f| \mathrm{d}\lambda < \infty$, the collection

$$[f] := \{g\colon [0,1] \to \mathbb{R} \mid \int_0^1 |f-g| \mathrm{d}\lambda = 0\}$$

of functions that coincide with f almost everywhere is an element of $\mathrm{L}_1([0,1])$. The vector-space operations factor through the equivalence class and $\|[f]\|_1 := \int_0^1 |f| \mathrm{d}\lambda$ defines a Norm on $\mathrm{L}_1([0,1])$. For this paper equip $\mathrm{L}_1([0,1])$ with the Cauchy representation with respect to the metric induced by this norm and choose as dense set the equivalence classes of piece-wise constant functions with a finite number of rational values and breakpoints.

For more on representing spaces of integrable functions, see [36] and [37]. Integration is computable as a mapping that takes endpoints of an interval and a function and integrates it over the interval.

Let us give a name to the restriction of the assignment $f \mapsto [f]$ to continuous functions:

$$\iota\colon \mathcal{C}([0,1],[0,1]) \to \mathrm{L}_1([0,1]), \quad f \mapsto [f].$$

As continuous functions that are equal almost everywhere are already equal, ι is an injection. However, a name of $\iota(f)$ provides strictly less information than one of f. Our first lemma follows from results by Brattka [3] but we provide an elementary direct proof in the appendix.

▶ **Lemma 25** ($\mathrm{lim} \leq_\mathrm{W} \iota^{-1}$)**.** *Evaluating an integrable function that happens to be continuous is enough to compute the limit of a sequence in Baire space.*

▶ **Lemma 26** ($\iota^{-1} \leq_\mathrm{W} \mathrm{lim}$)**.** *The inverse of the inclusion of* $\mathcal{C}([0,1],[0,1])$ *into* $\mathrm{L}_1([0,1])$ *is Weihrauch reducible to* lim*.*

Proof. [3] Recall that $\mathrm{lim} \equiv_\mathrm{W} \mathrm{lim}^\mathbb{N}_{[0,1]}$, so it suffices to produce a Weihrauch reduction of ι^{-1} to $\mathrm{lim}^\mathbb{N}_{[0,1]}$. Let us assume we are given $\iota(f) \in \mathrm{L}_1([0,1])$. Fix some enumeration $(I_k)_{k\in\mathbb{N}}$ of the rational intervals with endpoints in $[0,1]$. Assume we are given $\varepsilon \in \mathbb{Q}^+$ and we want to

[3] We are grateful to a referee for sketching this simplified proof for us.

produce some rational polynomial p such that $\|f-p\|_\infty \leq \varepsilon$. Let $(p_m)_{m\in\mathbb{N}}$ be an enumeration of the rational polynomials. Note that one easily obtains $[p_m]$ as an integrable function and as we can compute integrals of integrable functions, for each m the sequence $(x_n)_{n\in\mathbb{N}}$

$$x_n := \frac{1}{\lambda(I_n)}\Big|\int_{I_n} f - p_m \mathrm{d}\lambda\Big|$$

is computable. Note that $x_n \leq \|f-p_m\|_\infty$ and that we can get x_n arbitrary close to $\|f-p_m\|_\infty$: As f is continuous there exists an interval where f is almost constantly almost the value whose absolute value is maximal. Thus, one instance of $\lim_{[0,1]}$ can compute $\|f-p_m\|_\infty = \sup\{x_n \mid n\in\mathbb{N}\} = \lim_{[0,1]}\big((\max\{x_k \mid k\leq n\})_{n\in\mathbb{N}}\big)$. Moreover, using $\lim^{\mathbb{N}}_{[0,1]}$ produces $(\|f-p_m\|_\infty)_{m\in\mathbb{N}}$ as a sequence. Since f is continuous, we know that there exists some p_m such that $\|f-p_m\|_\infty < \varepsilon$. As a test for strict inequality is computable when its values are taken to be from $\mathbb{S}$, we may find such p_m by dovetailing and return it. ◀

4.2 The Lebesgue measure on the Δ^0_2-subsets of the unit interval

Let us now introduce the space $\Delta^0_2(\mathbf{X})$ of Δ^0_2-subsets of a represented space $\mathbf{X}$. First recall that $\Delta^0_1(\mathbf{X}) := \mathcal{C}(\mathbf{X},\mathbb{B})$ is the space of clopen subsets and can alternatively be represented a pair of names of the characteristic function of both a set and its complement as continuous functions to Sierpiński space. The space $\Delta^0_2(\mathbf{X})$ can be introduced the same way but with Sierpiński space replaced by its lim-jump as introduced in Section 2.3. As the product of lim-jumps of two spaces is isomorphic to the lim-jump of the product, from $\Delta^0_1(\mathbf{X}) = \mathcal{C}(\mathbf{X},\mathbb{B})$ we get $\Delta^0_2(\mathbf{X}) \simeq \mathcal{C}(\mathbf{X},\mathbb{B}')$, where $\mathbb{B}'$ is the lim-jump of $\mathbb{B}$. As every convergent sequence in $\mathbb{B}$ is already eventually constant, $\mathbb{B}'$ allows for a simpler description using another jump operator. Let $\lim_\Delta(p_n) := \{p\in\mathbb{N}^\mathbb{N} \mid \exists N, \forall n\geq N, p_n = p\}$ be the limit operator with respect to the discrete topology on Baire space.

▶ **Definition 27** (The $\lim_\Delta$-jump $\mathbf{X}^\nabla$ of a space $\mathbf{X}$)**.** *Define the* $\lim_\Delta$*-jump of a represented space* $\mathbf{X} = (X,\delta_\mathbf{X})$ *as* $\mathbf{X}^\nabla := (X, \delta_\mathbf{X}\circ\lim_\Delta\circ\delta_{(\mathbb{N}^\mathbb{N})^\mathbb{N}})$.

Note that the definition of the lim-jump of in Section 2.3 is identical, except that $\lim_\Delta$ replaces $\lim_{\mathbb{N}^\mathbb{N}}$. Now, $\mathbb{B}^\nabla$ is isomorphic to the lim-jump of $\mathbb{B}$ and $\Delta^0_2(\mathbf{X}) \simeq \mathcal{C}(\mathbf{X},\mathbb{B}^\nabla)$.

▶ **Lemma 28.** *There exists a computable function* $D\colon \mathrm{dom}(\lim_{[0,1]}) \to \Delta^0_2([0,1])$ *that maps each converging* $\mathbf{x}\in[0,1]^\mathbb{N}$ *to a set* $D(\mathbf{x})$ *with Lebesgue measure* $\lambda(D(\mathbf{x})) = \lim_{[0,1]}(\mathbf{x})$.

Proof. Our construction here has some similarities to the proof of Lemma 18. To argue that we can compute a Δ^0_2-set A from the sequence $(a_i)_{i\in\mathbb{N}}$ means that we can compute its characteristic function as $\chi_A\colon [0,1]\to\mathbb{B}_\Delta$ with access to $(a_i)_{i\in\mathbb{N}}$, which in turn is equivalent to a $\mathrm{LPO}^\diamond$-machine being able to compute $\chi_A\colon[0,1]\to\mathbb{B}$. We give the algorithm for the latter as Algorithm 1 in the appendix. ◀

From this Lemma we can draw the conclusion that we need.

▶ **Corollary 29** ($(\lambda\colon \Delta^0_2([0,1])\to\mathbb{R}) \equiv_\mathrm{W} \lim$)**.** *The Lebesgue measure as a function from* $\Delta^0_2([0,1])$ *to* $\mathbb{R}$ *is a representative of the Weihrauch degree* lim.

Proof. It suffices to prove $\lambda \leq_\mathrm{W} \lim_{\mathbb{N}^\mathbb{N}}$ and $\lim_{[0,1]} \leq_\mathrm{W} \lambda$. The latter of these follows directly from the previous Lemma 28.

To see that also $\lambda \leq_\mathrm{W} \lim_{\mathbb{N}^\mathbb{N}}$ assume that we are given $A\in\Delta^0_2([0,1])$ as input. Recall that a Δ^0_2-set is specified as a pair names of itself and its complement as Π^0_2-subsets of $[0,1]$. Now $[0,1]$ is a computable metric space and according to Lemma 10 we can thus

computably obtain names of sequences $(U_n)_{n\in\mathbb{N}}, (V_n)_{n\in\mathbb{N}} \in \mathcal{O}([0,1])^{\mathbb{N}}$ so that $\bigcap_{n\in\mathbb{N}} U_n = A$ and $\bigcap_{n\in\mathbb{N}} V_n = A^c$. Thus the sequences $x_n := \lambda(\bigcap_{k\leq n} U_k)$ converges to $\lambda(A)$ from above and the sequence $y_n := 1 - \lambda(\bigcap_{k\leq n} V_k)$ from below. Note that the Lebesgue measure is computable from open sets to $\mathbb{R}_<$. However, it is known that $(\mathrm{id}\colon \mathbb{R}_< \to \mathbb{R}) \equiv_{\mathrm{W}} \lim$ and we know that $\widehat{\lim} \equiv_{\mathrm{W}} \lim$ from Lemma 7. Therefore, $\lim_{\mathbb{N}^{\mathbb{N}}}$ is sufficient to lift both of the sequences (x_n) and (y_n) from sequences in $\mathbb{R}_<$ to sequences of real numbers. These sequences approximate $\lambda(A)$ from above and below so that we can compute $\lambda(A) \in \mathbb{R}$. ◀

4.3 Piece-wise continuous functions and Δ^0_2-measurable functions

Before we talk about Δ^0_2-measurable functions let us get back to admissibility and continuous functions for illustration. For any $f \in \mathcal{C}(\mathbf{X}, \mathbf{Y})$ we may consider the pre-image function $f^{-1}\colon \mathcal{O}(\mathbf{Y}) \to \mathcal{O}(\mathbf{X})$ defined by $f^{-1}(\chi) := \chi \circ f$. The above says that continuously realizable functions between represented spaces are such that the preimage of an open set is not only open but can continuously be obtained from it. Now, admissibility of the representation of $\mathbf{Y}$ guarantees that the notions coincide in that it assures that the assignment $f \mapsto f^{-1}$ can be inverted and in particular $\mathcal{C}(\mathbf{X}, \mathbf{Y})$ is isomorphic to its image in $\mathcal{C}(\mathcal{O}(\mathbf{Y}), \mathcal{O}(\mathbf{X}))$ under moving to the pre-image function. A function is called Δ^0_2-measurable if its preimages of open sets are Δ^0_2-sets. We may thus ask whether there is an assumption that allows reconstruction of a Δ^0_2-measurable function from its preimage function $\mathcal{O}(\mathbf{Y}) \to \Delta^0_2(\mathbf{X})$ just like admissibility did this for continuity. Indeed, there exists such a condition and it is called $\lim_\Delta$-admissibility. It implies that $\mathcal{C}(\mathbf{X}, \mathbf{Y}^\nabla)$ is isomorphic to the corresponding subspace of $\mathcal{C}(\mathcal{O}(\mathbf{Y}), \Delta^0_2(\mathbf{X}))$. Here $\mathbf{Y}^\nabla$ is the $\lim_\Delta$-jump of $\mathbf{Y}$ from Definition 27 and as its underlying set is identical to that of $\mathbf{Y}$, measurable functions are indeed functions from $\mathbf{X}$ to $\mathbf{Y}$. We point to [33, 32] for proofs that all spaces that appear as $\mathbf{Y}$ in this paper are $\lim_\Delta$-admissible.

The space $\mathcal{C}(\mathbf{X}, \mathbf{Y}^\nabla)$ can often be understood as the space of piece-wise continuous functions via the Jayne-Rogers theorem. A function $f\colon \mathbf{X} \to \mathbf{Y}$ is piece-wise continuous if there is a sequence $(A_n)_{n\in\mathbb{N}} \in \mathcal{A}(\mathbf{X})^{\mathbb{N}}$ that covers $\mathbf{X}$ and such that for each $n \in \mathbb{N}$ the function $f|_{A_n}$ is continuous from A_n as a subspace of $\mathbf{X}$ to $\mathbf{Y}$. Let $\mathcal{C}_{pw}(\mathbf{X}, \mathbf{Y})$ be the space of piece-wise continuous functions represented as expected.

▶ **Lemma 30** ($\mathcal{C}(\mathbf{X}, \mathbf{Y}^\nabla) \simeq \mathcal{C}_{pw}(\mathbf{X}, \mathbf{Y})$, proven in [33]). *Let $\mathbf{X}$ and $\mathbf{Y}$ be a computable metric spaces such that $\mathbf{Y}$ is complete. Then $\mathcal{C}(\mathbf{X}, \mathbf{Y}^\nabla)$ is isomorphic to the space of piece-wise continuous functions $\mathcal{C}_{pw}(\mathbf{X}, \mathbf{Y})$ as introduced above.*

Let us give a name to another restriction of the assignment $f \mapsto [f]$, namely set

$$\iota_\Delta\colon \mathcal{C}([0,1], [0,1]^\nabla) \to \mathrm{L}_1([0,1]), \quad f \mapsto [f].$$

This map is not injective: as piece-wise continuous functions characteristic functions of singletons are distinct from zero but they produce the same equivalence class in $\mathrm{L}_1([0,1])$.

▶ **Lemma 31** ($\left(\int\colon \mathcal{C}([0,1], [0,1]^\nabla) \to \mathbb{R}\right) \leq_{\mathrm{W}} \lim$). *Integrating a piece-wise continuous function over the unit interval is Weihrauch reducible to $\lim_{\mathbb{R}}$.*

Proof. According to Lemma 30, the information that we get about f is a sequence A_n of closed sets together with the restrictions $f|_{A_n}$. By Lemma 8, the information contained in a name of $A_n \in \mathcal{A}([0,1])$ is an increasing sequence $(U^i_n)_{i\in\mathbb{N}}$ of finite unions of rational intervals such that $\bigcup_{i\in\mathbb{N}} U^i_n = A^c_n$. Construct a sequence $r_n \in [0,1]$ as follows: For each $i \leq 2^n$ set $x_i := i \cdot 2^{-n}$ and pick some canonical name of $x_i \in [0,1]$. For each $k \leq n$ check whether $x_i \in U^k_n$, and if this is not the case evaluate the universal used for function spaces for n steps

in attempt to obtain a 2^{-n} approximation to $f|_{A_k}(x_i)$. Let $g(x_i)$ be the first value for which this succeeds and returns something in $[-2^{-n}, 1+2^{-n}]$ and let $g(x_i)$ be zero if this never happens. Set $r_n := \sum_{i \leq 2^n} g(x_i) \cdot 2^{-n}$ and note that our description of this sequence provides a way to compute it from the input information. To see that the sequence r_n converges to the integral of f one uses the sigma-additivity of the Lebesgue measure and that, as the A_n cover $[0,1]$, it holds that $\lim_{\mathbb{R}}(\lambda(A_n))_{n\in\mathbb{N}} = 1$ and also $\lim_{\mathbb{R}}(\lambda(U_n^i))_{i\in\mathbb{N}} = 1 - \lambda(A_n)$. ◀

▶ **Corollary 32** ($\iota_\Delta \equiv_{\mathrm{W}} \lim$)**.** *The Weihrauch degree of the mapping from* $\mathcal{C}([0,1],[0,1]^\nabla)$ *to* $\mathrm{L}_1([0,1])$ *that takes* f *to its equivalence class is* lim*.*

Proof. Let us first argue that $\iota_\Delta \leq_{\mathrm{W}} \lim$ and thus assume that we are given an input for ι_Δ, i.e. some $f \in \mathcal{C}([0,1],[0,1]^\nabla)$. Let $(p_k)_{k\in\mathbb{N}}$ be an enumeration of rational piece-wise constant functions similar to the one we use for defining the Cauchy representation of $\mathrm{L}_1([0,1])$ but who map to $[0,1]$. Note that the sequence $(p_k)_{k\in\mathbb{N}}$ is computable as sequence in $\mathcal{C}([0,1],[0,1]^\nabla)$ and that taking differences and the absolute value are computable operations on the piece-wise continuous functions as they can be lifted in a pointwise manner from the continuous functions. Thus, for each fixed k we can compute $\|[f]-[p_k]\|_1 = \int_0^1 |f-p_k| \mathrm{d}\lambda$ using an invocation of lim by the previous Lemma. As $\widehat{\lim} \equiv_{\mathrm{W}} \lim$ this means that we can also compute $(\|[f]-[p_k]\|_1)_{k\in\mathbb{N}}$ as a sequence by one application of lim. Now given some $\varepsilon \in \mathbb{Q}^+$ we know that there exists some p_k such that $\|[f]-[p_k]\|_1 < \varepsilon$ and may thus search for it and return it.

To see that also $\lim \leq_{\mathrm{W}} \iota_\Delta$ note that $\mathbb{B}^\nabla$ is isomorphic to $\{0,1\}$ as subspace of $[0,1]^\nabla$. Thus we can computably obtain the characteristic function $\chi_A \in \mathcal{C}([0,1],[0,1]^\nabla)$ from a $A \in \Delta^0_2([0,1])$. As $\int_0^1 \chi_A \mathrm{d}\lambda = \lambda(A)$ we can thus use ι_Δ to compute the Lebesgue measure as function from $\Delta^0_2([0,1])$ to $\mathbb{R}$ and get the desired reduction from Lemma 29. ◀

References

1 Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and Real Computation.* Springer, 1998.

2 Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989. URL: `http://projecteuclid.org/euclid.bams/1183555121`.

3 Vasco Brattka. Computable invariance. *Theoretical Computer Science*, 210:3–20, 1999. `doi:10.1016/S0304-3975(98)00095-4`.

4 Vasco Brattka. A computable Kolmogorov superposition theorem. Informatik Berichte 272, FernUniversität Hagen, 2000.

5 Vasco Brattka. Effective Borel measurability and reducibility of functions. *Mathematical Logic Quarterly*, 51(1):19–44, 2005. `doi:10.1002/malq.200310125`.

6 Vasco Brattka, Matthew de Brecht, and Arno Pauly. Closed choice and a uniform low basis theorem. *Annals of Pure and Applied Logic*, 163(8):968–1008, 2012. `doi:10.1016/j.apal.2011.12.020`.

7 Vasco Brattka, Guido Gherardi, Rupert Hölzl, Hugo Nobrega, and Arno Pauly. Borel choice. in preparation.

8 Vasco Brattka, Guido Gherardi, and Arno Pauly. Weihrauch complexity in computable analysis, 2017. URL: `https://arxiv.org/abs/1707.03202`.

9 Vasco Brattka and Peter Hertling. Feasible real random acess machines. *Journal of Complexity*, 14:490–526, 1998. `doi:10.1006/jcom.1998.0488`.

10 Vasco Brattka, Peter Hertling, and Klaus Weihrauch. A tutorial on computable analysis. In Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *New Computational Paradigms:*

Changing Conceptions of What is Computable, pages 425–491. Springer, 2008. URL: https://link.springer.com/chapter/10.1007/978-0-387-68546-5_18.

11 Vasco Brattka and Arno Pauly. On the algebraic structure of Weihrauch degrees. *Logical Methods in Computer Science*, 14(4):1–36, 2018. doi:10.23638/LMCS-14(4:4)2018.

12 Vasco Brattka and Gero Presser. Computability on subsets of metric spaces. *Theoretical Computer Science*, 305(1-3):43–76, 2003. doi:10.1016/S0304-3975(02)00693-X.

13 Peter Bürgisser and Felipe Cucker. Counting complexity classes for numeric computations ii: Algebraic and semialgebraic sets. *Journal of Complexity*, 22(2):147–191, 2006. doi:10.1016/j.jco.2005.11.001.

14 Thomas Chadzelek and Günter Hotz. Analytic machines. *Theoretical Computer Science*, 219:151–167, 1999. doi:10.1016/S0304-3975(98)00287-4.

15 Matthew de Brecht. Levels of discontinuity, limit-computability, and jump operators. In Vasco Brattka, Hannes Diener, and Dieter Spreen, editors, *Logic, Computation, Hierarchies*, pages 79–108. de Gruyter, 2014. doi:10.1515/9781614518044.79.

16 Matthew de Brecht, Arno Pauly, and Matthias Schröder. Overt choice. *Computability*, 2020. available at https://arxiv.org/abs/1902.05926. doi:10.3233/COM-190253.

17 Hugo de Holanda Cunha Nobrega. Game characterizations of function classes and Weihrauch degrees. M.Sc. thesis, University of Amsterdam, 2013. URL: https://eprints.illc.uva.nl/905/1/MoL-2013-16.text.pdf.

18 Tobias Gärtner and Martin Ziegler. Real analytic machines and degrees. *Logical Methods in Computer Science*, 7:1–20, 2011. doi:10.2168/LMCS-7(3:11)2011.

19 Christine Gaßner. On NP-completeness for linear machines. *Journal of Complexity*, 13:259–271, 1997. doi:10.1006/jcom.1997.0444.

20 Christine Gaßner. The P-DNP problem for infinite abelian groups. *Journal of Complexity*, 17:574–583, 2001. doi:10.1006/jcom.2001.0583.

21 Christine Gaßner. A hierarchy below the halting problem for additive machines. *Theory Computing Systems*, 17:574–583, 2008. doi:10.1007/s00224-007-9020-y.

22 Christine Gaßner. An introduction to a model of abstract computation: the BSS-RAM model. In Adrian Rezus, editor, *Contemporary Logic and Computing*, volume 1 of *Landscapes in Logic*, pages 574–603. College Publications, 2020.

23 Christine Gaßner and Pedro F. Valencia Vizcaíno. Operators for BSS RAM's. In Martin Ziegler and Akitoshi Kawamura, editors, *The Twelfth International Conference on Computability and Complexity in Analysis*, pages 24–26, 2015.

24 Vassilios Gregoriades, Tamás Kispéter, and Arno Pauly. A comparison of concepts from computable analysis and effective descriptive set theory. *Mathematical Structures in Computer Science*, 27(8):1414–1436, 2017. 2015. doi:10.1017/S0960129516000128.

25 Anders C. Hansen. On the solvability complexity index, the n-pseudospectrum and approximations of spectra of operators. *Journal of the AMS*, 24:81–124, 2011.

26 Armin Hemmerling. Computability of string functions over algebraic structures. *Mathematical Logic Quarterly*, 44(1):1–44, 1998. doi:10.1002/malq.19980440102.

27 Denis R. Hirschfeldt and Carl G. Jockusch. On notions of computability-theoretic reduction between Π^1_2-principles. *Journal of Mathematical Logic*, 16(1), 2016. 1650002:1-1650002:59. doi:10.1142/S0219061316500021.

28 Klaus Meer. Counting problems over the reals. *Theoretical Computer Science*, 242:41–58, 2000. doi:10.1016/S0304-3975(98)00190-X.

29 Yiannis N. Moschovakis. Abstract first order computability. I. *Transactions of the American Mathematical Society*, 138:427–464, 1969. doi:10.2307/1994926.

30 Eike Neumann and Arno Pauly. A topological view on algebraic computations models. *Journal of Complexity*, 44:1–22, 2018. doi:10.1016/j.jco.2017.08.003.

31 Arno Pauly. On the topological aspects of the theory of represented spaces. *Computability*, 5(2):159–180, 2016. doi:10.3233/COM-150049.

32 Arno Pauly and Matthew de Brecht. Towards synthetic descriptive set theory: An instantiation with represented spaces. http://arxiv.org/abs/1307.1850, 2013.

33 Arno Pauly and Matthew de Brecht. Non-deterministic computation and the Jayne Rogers theorem. *Electronic Proceedings in Theoretical Computer Science*, 143:87–96, 2014. DCM 2012. doi:10.4204/EPTCS.143.8.

34 Arno Pauly and Matthew de Brecht. Descriptive set theory in the category of represented spaces. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 438–449, 2015. doi:10.1109/LICS.2015.48.

35 Arno Pauly and Florian Steinberg. Comparing representations for function spaces in computable analysis. *Theory of Computing Systems*, 62(3):557–582, 2018. doi:10.1007/s00224-016-9745-6.

36 Marian Pour-El and Ian Richards. *Computability in analysis and physics*. Perspectives in Mathematical Logic. Springer, 1989.

37 Florian Steinberg. Complexity theory for spaces of integrable functions. *Logical Methods in Computer Science*, 13(3):1–39, 2017. doi:10.23638/LMCS-13(3:21)2017.

38 Nazanin Tavana and Klaus Weihrauch. Turing machines on represented sets, a model of computation for analysis. *Logical Methods in Computer Science*, 7:1–21, 2011. doi:10.2168/LMCS-7(2:19)2011.

39 John V. Tucker and Jeffrey I. Zucker. Computable functions and semicomputable sets on many-sorted algebras. In T.S.E. Maybaum S. Abramsky, D.M. Gabbay, editor, *Handbook of Logic in Computer Science*, volume 5 of *Oxford Science Publications*, pages 317–523, 2000.

40 Klaus Weihrauch. *Computable Analysis*. Springer-Verlag, 2000.

41 Linda Westrick. A note on the diamond operator. *Computability*, 202X. to appear. doi:10.3233/COM-200295.

42 Martin Ziegler. Computability and continuity on the real arithmetic hierarchy and the power of type-2 nondeterminism. In Barry S. Cooper, Benedikt Löwe, and Leen Torenvliet, editors, *Proceedings of CiE 2005*, volume 3526 of *Lecture Notes in Computer Science*, pages 562–571. Springer, 2005. URL: https://link.springer.com/chapter/10.1007/11494645_68.

A Proof of Lemma 28

That we have access to LPO is relevant in Lines 5 and 7, where LPO lets us resolve the if-statements. As we can use arbitrary computable functions, the universal quantifiers $\forall i \geq n$ are unproblematic here. Since the sequence $(a_i)_{i\in\mathbb{N}}$ is guaranteed to converge, the while-loop will terminate. If $b = 0$, then for sure $\lim_{i\to\infty} a_i \in [0, 2/3]$, if $b = 1$ then $\lim_{i\to\infty} a_i \in [1/3, 1]$. In the latter case, Line 15 adds $1/3$ to the measure of A. By moving to the sequence $(a_i - b/3)_{i\in\mathbb{N}}$ we then get a sequence guaranteed to have a limit in $[0, 2/3]$, and still have to determine membership in A for $x \in (1/3, 1)$. We rescale this interval up to $(0, 1)$ again, and iterate the process.

B Proof of Lemma 25

Proof. As $\lim \equiv_{\mathrm{W}} \widehat{\mathrm{LPO}}$ by Lemma 7 we may solve a countable number of instances of $\mathrm{LPO} = \chi_{\{k\mapsto 0\}}$ instead of one of lim. Thus, our input is a sequence (p_n) of elements of Baire space. Let m_n be the least natural number such that $p_n(m_n) \neq 0$ if such a number exists and ∞ otherwise. Define a sequence of functions $f_n \in \mathcal{C}([0, 1])$ by

$$f_n(x) := \begin{cases} 0 & \text{if } m_n = \infty \\ \max\{0, 1 - 2^{m_n+2}|x - \frac{3}{4}|\} & \text{otherwise,} \end{cases}$$

Algorithm 1 Computing the characteristic function of A.

```
1  Function Charac is
       input  : x ∈ [0,1], converging sequence (a_i)_{i∈ℕ}
       output : Boolean indicating whether x ∈ A
2      if x = 0 then return "no";
3      b := −1; n = 0; while b = −1 do
4          if ∀i ≥ n a_i ≤ 2/3 then
5              b := 0;
6          else if ∀i ≥ n a_i ≥ 1/3 then
7              b := 1;
8          else
9              n := n + 1;
10         end if
11     end while
12     if x ≥ 2/3 then
13         if b = 0 then return "no";
14         if b = 1 then return "yes";
15     else
16         return Charac (3/2 x, (3/2 a_i − 1/2 b)_{i∈ℕ})
17     end if
18 end
```

and let f be defined by an infinite sum:

$$f(x) := \sum_{n\in\mathbb{N}} 2^{-n} f_n(2^n x).$$

This sum converges in supremum norm and f is a continuous function. To see that a sequence of piece-wise linear approximations to f in L_1-norm can be computed from p note that

$$\int 2^{-n} f_n(2^n x)dx = 2^{-2n} \int f_n(x)dx = 2^{-2(n+1)-m_n}.$$

Thus, the infinite sum may be approximated by the finite sums

$$\sum_{n \text{ s.t. } \max\{n,m_n\}\leq k} 2^{-n} f_n(2^n x)$$

up to any desired precision ε by choosing an appropriate k. Furthermore, these finite sums are computable from the input sequence p_n. An application of a solution of ι^{-1} provides with this function as a continuous function. In particular we may evaluate the values of the function in the peaks and can read the value of $\chi_{\{k\mapsto 0\}}(p_n)$ from these. ◀

C The BSS RAM model – some details

In the following, an algebraic structure $\mathcal{A}$ is a tuple $(U; c_1, c_2, \ldots; f_1, f_2, \ldots; r_1, r_2, \ldots)$ with universe $U_{\mathcal{A}} = U$, where U is any nonempty set, each c_i is an element in U, each f_i is a (partial) function of type $f_i :\subseteq U^{m_i} \to U$ $(m_i \geq 1)$, and each r_i is a relation of type $r_i \subseteq U^{k_i}$. Here, any $\mathcal{A}$-machine $\mathcal{M}$ computes a function of type $g :\subseteq U^* \to U^*$. It has

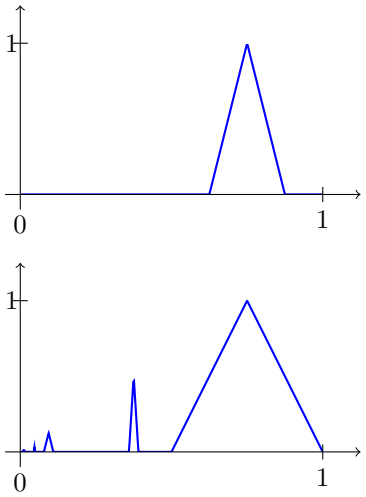

Figure 2 f_n for $m_n = 1$ and f for $m_n = (0, 3, \infty, 1, 6, \infty, \ldots)$.

registers $(Z_i)_{i\geq 1}$ and index registers $(I_j)_{j\in\{1,\ldots,k_{\mathcal{M}}\}}$ and, at any time, the content $c(Z_i)$ of any Z_i is an element of U and any I_j holds a natural number $c(I_j)$. Each program $\mathcal{P}_{\mathcal{M}}$ of any $\mathcal{M}$ is a finite list of labeled commands of the following forms: **computation instructions** $\ell : Z_j := f_i^{m_i}(Z_{j_1}, \ldots, Z_{j_{m_i}})$ and $\ell : Z_j := c_i^0$, **copy instructions** $\ell : Z_{I_j} := Z_{I_k}$, **branching instructions** $\ell :$ if $r_i^{k_i}(Z_{j_1}, \ldots, Z_{j_{k_i}})$ then goto ℓ_1 else goto ℓ_2, **index instructions** $\ell :$ if $I_j = I_k$ then goto ℓ_1 else goto ℓ_2, $\ell : I_j := 1$, and $\ell : I_j := I_j + 1$, a **stop instruction** $l :$ stop.

For explaining some details, let $(\vec{x}\,.\,\vec{y}) = (x_1, \ldots, x_n, y_1, \ldots, y_m) \in U^{n+m}$ and $(\vec{x}\,.\,\bar{z}) = (x_1, \ldots, x_n, z_1, z_2, \ldots) \in U^{\omega}$ for $n, m \geq 0$ and $\vec{x} = (x_1, \ldots, x_n) \in U^n$, $\vec{y} = (y_1, \ldots, y_m) \in U^m$, and $\bar{z} = (z_1, z_2, \ldots) \in U^{\omega}$. For any $\mathcal{A}$-machine $\mathcal{M}$, let $\{(\ell\,.\,\vec{\iota}\,.\,\bar{z}) \mid \ell \in \mathcal{L}_{\mathcal{M}}\ \&\ \vec{\iota} \in \mathbb{N}_+^{k_{\mathcal{M}}}\ \&\ \bar{z} \in U^{\omega}\}$ be the space of all possible configurations of $\mathcal{M}$ with the list $\mathcal{L}_{\mathcal{M}} =_{\mathrm{df}} \{1, \ldots, \ell_{\mathcal{M}}\}$ of labels. For computing partial functions of the form $f :\subseteq U^* \to U^*$, we use $\mathcal{A}$-machine $\mathcal{M}$ – so-called BSS RAM's over $\mathcal{A}$ – with at least one constant c_1 and $k_{\mathcal{M}} \geq 2$ and an input and an output procedure. Let the input procedure of $\mathcal{M}$ be determined by $\mathrm{Input}_{\mathcal{M}}(x_1, \ldots, x_n) = (1\,.\,\vec{\iota}\,.\,(x_1, \ldots, x_n, x_n, x_n, \ldots))$ with $\vec{\iota} = (n, 1, \ldots, 1) \in \mathbb{N}_+^{k_{\mathcal{M}}}$ and $\mathrm{Input}_{\mathcal{M}}(()) = (1\,.\,\vec{\iota}\,.\,(c_1, c_1, \ldots))$ with $\vec{\iota} = (1, 2, 1, \ldots, 1) \in \mathbb{N}_+^{k_{\mathcal{M}}}$. Let the output procedure be given by $\mathrm{Output}_{\mathcal{M}}(\ell\,.\,\vec{\iota}\,.\,\bar{z}) = (z_1, \ldots, z_{\iota_1})$ if $(\iota_1, \iota_2) \neq (1, 2)$ and $\mathrm{Output}_{\mathcal{M}}(\ell\,.\,\vec{\iota}\,.\,\bar{z}) = ()$ for $(\iota_1, \iota_2) = (1, 2)$. This means, at the start of the computation, the register I_1 contains the length of the input, all other I_j start at 1. The input is in $Z_1, \ldots, Z_n$, all other Z_i contain the constant $c_1 \in U_{\mathcal{A}}$. If the program fails to halt on some input, the computed function is undefined on these values.

For $\mathbb{R}_0 = (\mathbb{R}; 0, 1, q_1, q_2, \ldots; -, +, \times; <, =)$ with $\{q_1, q_2, \ldots\} = \mathbb{Q}$, let $\mathsf{M}_{\mathbb{R}_0}$ be the class of all BSS machines without irrational constants which can be considered to be a BSS RAM over $\mathbb{R}_0$. For a universal register machine, as inputs we use the first part of the code for encoding the program by means of Gödel numberings gn as given in [22] and the second part for the constants. For any $\mathbb{R}_0$-machine $\mathcal{M}$ with the constants in $\vec{c}^{\,(\mathcal{M})} = (c_{j_1}, \ldots, c_{j_{n_1}})$, let $\mathrm{code}(\mathcal{M}) = (\mathrm{code}(\mathcal{P}_{\mathcal{M}})\,.\,\vec{a}^{\,(\mathcal{M},1)})$ where $\mathrm{code}(\mathcal{P}_{\mathcal{M}}) \in \{1\}^{gn(\mathcal{P}_{\mathcal{M}})}$ and $\vec{a}^{\,(\mathcal{M},a)} = (a_1, \ldots, a_{\ell_{\mathcal{M}}})$ is defined as follows. For any $\ell \leq \ell_{\mathcal{M}}$, let a_ℓ be the i^{th} component c_{j_i} in $\vec{c}^{\,(\mathcal{M})}$ if the ℓ^{th} instruction of $\mathcal{P}_{\mathcal{M}}$ is the instruction $Z_j := c_i^0$ for some j and otherwise let $a_\ell = a$. We know that there is a universal BSS RAM $\mathcal{M}_0 \in \mathsf{M}_{\mathcal{A}}$ over $\mathbb{R}_0$ satisfying $\mathcal{M}_0(\mathrm{code}(\mathcal{M})\,.\,\vec{x}) = \mathcal{M}(\vec{x})$ for all $\vec{x} \in U_{\mathcal{A}}^{\infty}$ and any BSS RAM $\mathcal{M}$ over $\mathbb{R}_0$ ($U_{\mathcal{A}}^{\infty}$ contains all tuples/strings in $U_{\mathcal{A}}^*$

without the empty string). Any simple 1-tape Turing machine M computing a function $f :\subseteq \{0,1\}^* \to \{0,1\}^*$ or $f :\subseteq \mathbb{N} \to \mathbb{N}$ can be simulated by $\mathcal{A}_0$-machines $\mathcal{M}^{\mathrm{T}}(M)$ and $\mathcal{M}^{\mathrm{T}}_{\mathbb{N}}(M)$, respectively, for $\mathcal{A}_0 = (\{0,1\}; 0, 1; ; =)$ with suitable input and output procedures (for details see [22]) and thus be encoded by the Gödel number $\mathrm{code}(M) = gn(\mathcal{P}_{\mathcal{M}^{\mathrm{T}}(M)})$. By analogy, for computing a function $f :\subseteq \mathbb{R} \to \mathbb{R}$ by a type-2 machine M, we can take an $\mathcal{A}_0$-machine with modified input and output for simulating M. A **BSS+Comp machine** $\mathcal{M}$ is a generalization of a BSS machine that can additionally execute instructions of the form

$$Z_j := Comp(I_1, Z_1). \tag{1}$$

If $c(I_1)$ is the code of some type-2 machine M and M computes, on input $c(Z_1)$, the name of a real value, then, by (1), this value is assigned to Z_j and, otherwise, (1) causes that $\mathcal{M}$ does not halt.

Now, we define oracle instructions similar to instructions in Moschovakis' model using the ν-oparator (cf. [29]) and use an operator $\vec{\nu}$ in order to introduce the deterministic measure operator λ and other operators (cf. [23]). A consequence is that we can get – in the same way – a precise definition of oracle instructions that can be considered as a generalization of instructions introduced in [13, p. 156] for characterizing counting complexity classes for numeric computations by using classes such as $\sharp\mathrm{P}_{\mathbb{R}}$ introduced in [28, p. 44]. For $f :\subseteq \mathbb{R}^\infty \to \mathbb{R}^*$, let the $\vec{\nu}$-operator here provide a total function $\vec{\nu}[f]$ from $\mathbb{R}^*$ into the power set $\mathcal{P}(\mathbb{R}^\infty)$ of $\mathbb{R}^\infty$. For $\vec{x} \in \mathbb{R}^*$, let $\vec{\nu}[f](\vec{x}) = \{\vec{y} \in \mathbb{R}^\infty \mid f(\vec{x} \,.\, \vec{y}) = 0\}$. We are interested in measures of such sets for a universal function $f_{\mathrm{BSS-uni}} : \mathbb{R}^\infty \to \mathbb{R}^*$ with $f_{\mathrm{BSS-uni}}(\vec{z}) = \mathcal{M}(\vec{y})$ if if there are an $\mathcal{M} \in \mathsf{M}_{\mathbb{R}_0}$, a $k \geq 1$ and a $\vec{y} \in \mathbb{R}^k$ with $\vec{z} = (\mathrm{code}(\mathcal{M}) \,.\, k \,.\, \vec{y})$ and $f_{\mathrm{BSS-uni}}(\vec{z}) = 0$ otherwise. Let $\lambda(A)$ be the Lebesgue measure of a set $A \subseteq \mathbb{R}^k$ if A is in the considered σ-algebra, and undefined otherwise. The **measure operator** $\lambda[f_{\mathrm{BSS-uni}}]$ provides, for any k, a partial function from $\mathbb{R}^*$ into the interval $[0, \infty]$. Let

$$\ell :\ Z_j := \lambda[f_{\mathrm{BSS-uni}}](I_1, Z_1, \ldots, Z_{I_2}, I_3). \tag{2}$$

If there is an $\mathcal{M} \in \mathsf{M}_{\mathbb{R}_0}$ with Gödel number $gn(\mathcal{P}_{\mathcal{M}})$ stored in I_1 and the constants stored in $Z_1, \ldots, Z_{c(I_2)}$ that decides, for $k = c(I_3)$, $A =_{\mathrm{df}} \vec{\nu}[f_{\mathrm{BSS-uni}}](code(\mathcal{M}) \,.\, k) \cap [0,1]^k$ on the interval $[0,1]^k$ and $\lambda(A)$ is defined and finite (and, thus, in $[0, \infty[$), then by (2) $\lambda(A)$ is assigned to the register Z_j. Otherwise, an instruction of the form (2) causes that a machine trying to execute the instruction and to compute the corresponding measure does not halt.

For determining the limits of sequences $(y_i)_{i \in \mathbb{N}}$ with elements in $\mathbb{R}$, two further deterministic operators, the **limit operator** lim (resp. the **u-limit operator** u-lim for uncontrolled limits) and the **c-limit operator** c-lim (for controlled limits), are available and they can be used in executing instructions of the form

$$\ell :\ Z_j := [\text{c-}]\lim[f](I_1, Z_1, \ldots, Z_{I_2}), \tag{3}$$

These operators correspond to the strongly analytic and to the weakly analytic machines going back to Hotz [14]. The differences between both instructions are partially comparable to those of weakly analytic and strongly analytic machines over $(\mathbb{R}; \mathbb{R}; +, -, \cdot, /; =, <)$ that compute the limits by producing sequences $y_0, y_1, \ldots$ weakly analytically and strongly analytically, respectively; for details see the paper [18, p. 4] on relationships between the BSS machines over real numbers (cf. [1]) and the analytic machines (cf. [14]).

For any permitted function $f :\subseteq \mathbb{R}^\infty \to \mathbb{R}^*$, $[\text{c-}]\lim[f]$ is a function of the form $g :\subseteq \mathbb{R}^* \to [-\infty, +\infty]$. For $f :\subseteq \mathbb{R}^\infty \to \mathbb{R}^*$ and $\vec{x} \in \mathbb{R}^*$, $\lim[f](\vec{x})$ is defined if and only if there are a convergent sequence $y_0, y_1, \ldots \in \mathbb{R}$ and an $\mathcal{M} \in \mathsf{M}_{\mathbb{R}_0}$ whose Gödel number $gn(\mathcal{P}_{\mathcal{M}})$

is stored in I_1 and whose constants are stored in $Z_1, \ldots, Z_{c(I_2)}$ that computes y_i on i for all $i \in \mathbb{N}$ and then $\lim[f](\vec{x})$ is the limit of this sequence; c-$\lim[f](\vec{x})$ is equal to the limit if, moreover, $|y_i - y_j| < 2^{-\min\{i,j\}}$ is satisfied for all $i, j \in \mathbb{N}$, and undefined otherwise. Let $f_{\mathrm{BSS-enu}}$ be the function f satisfying $f(\vec{z}) = 0$ if there is a BSS machine $\mathcal{M}$ satisfying $(\forall i \in \mathbb{N})(\exists y_i \in \mathbb{R})(\vec{z} = (\mathrm{code}(\mathcal{M}) \,.\, y_i)$ & $\mathcal{M}(i) = \ y_i)$ and $f(\vec{z}) = 1$ otherwise. This implies that $\{y_i \mid i \in \mathbb{N}\} = \nu[f_{\mathrm{BSS-enu}}](\mathrm{code}(\mathcal{M}))$ for any BSS machine $\mathcal{M}$. Then, the instructions (3) with $f = f_{\mathrm{BSS-enu}}$ allow to compute the corresponding finite limit if the limit of the sequence enumerating by $\mathcal{M}$ exists and is in $\mathbb{R}$ and, otherwise, a machine trying to execute such an instruction does not halt.

A **BSS+$\lambda[f_{\mathrm{BSS-uni}}]$ machine** is a generalization of a BSS machine which can additionally execute instructions of the form (2). A function g is **BSS+λ-computable** if g is computable by a BSS+$\lambda[f_{\mathrm{BSS-uni}}]$ machine. A function g is **BSS+λ+λ-computable** if g is computable by a BSS+$\lambda[f_{\mathrm{BSS+\lambda-uni}}]$ machine. In a similar way, we can define the computability by means of other generalizations of BSS machines as follows. A **BSS+[c-]**$\lim[f_{\mathrm{BSS-enu}}]$ **machine** is a generalization of a BSS machine which can additionally execute the corresponding instructions of the form (3). A function g is a **BSS+[c-]**$\lim$**-computable** if g is computable by a BSS+[c-]$\lim[f_{\mathrm{BSS-enu}}]$ machine. A function g is **BSS+**$\lim$**+**c-$\lim$**-computable** if g is computable by a BSS+c-$\lim[f_{(\mathrm{BSS+lim})-\mathrm{enu}}]$ machine.