δ-approximable Functions*

Charles Meyssonnier[1], Paolo Boldi[2], and Sebastiano Vigna[2]

[1] École Normale Supérieure de Lyon, France
[2] Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Italy

Abstract In this paper we study several notions of approximability of functions in the framework of the BSS model. Denoting with φ_M^δ the function computed by a BSS machine M when its comparisons are against $-\delta$ rather than 0, we study classes of functions f for which $\varphi_M^\delta \to f$ in some sense (pointwise, uniformly, etc.). The main equivalence results show that this notion coincides with Type 2 computability when the convergence speed is recursively bounded. Finally, we study the possibility of extending these results to computations over Archimedean fields.

1 Introduction

The problem of extending classical recursion theory to the non-discrete world of real numbers has given rise to two complementary approaches: following the tradition of Turing, one can extend the notion of Turing machine by allowing input and output tape to contain (infinite) representations of real numbers; this approach gave rise to *Type 2 Theory of Effectivity (TTE)* [16]. On the other hand, it is possible to consider the reals as basic atomic entities, on which exact computations and tests are permitted, as in the *BSS model* [1].

These models arise in several generalized theories of computability: Tucker and Zucker are able to obtain both BSS computability and TTE as special instances of *many-sorted topological partial algebras* [13]. In turn, an instance of *distributive computability*, introduced by R.F.C. Walters by means of a programming language based on distributive categories [15], gives back the BSS model [14]. Moreover, TTE can be further characterized by means of domain theory [7]. In [9], two notions of function computability (algebraic and approximate) are studied by means of *finite algorithmic procedures*, and they respectively correspond to BSS and TTE computability.

In a previous paper [2], the last two authors have introduced a version of the BSS model of computability [1] in which exact tests are not allowed. Essentially, a BSS machine is δ-uniform iff its halting set and computed function do not change when the test for equality with 0 is replaced with a test for membership to an arbitrary ball around 0.

There is a strict relation between δ-uniform computability and TTE, that is, recursive analysis; indeed, for any Archimedean field the halting sets of δ-uniform BSS machines with coefficients in **T** (the field of Turing computable reals) or **Q** are exactly

* The last two authors have been partially supported by the ESPRIT Working Group EP 27150, Neural and Computational Learning II (NeuroCOLT II).

the halting sets of Type 2 Turing machines [2]. Thus, the restriction of δ-uniformity reduces the full power of the BSS model, making it closer to Turing machines.

However, the notion of δ-uniformity is not expressive in terms of computed functions: indeed, in [3] it was shown that, essentially, δ-uniformly computable functions are just the rational functions, whereas Type 2 computable functions form a much wider class.

The main reason behind this difference is that the definition of computable function in TTE carries inherently a notion of continuous approximation, whereas BSS computable functions (even δ-uniformly computable functions) are computed in a finite number of steps with infinite precision.

In this paper we investigate several notions of approximability of functions in the BSS model. Given a BSS machine M, we consider, as in [2], the function φ_M^δ computed by M when its comparisons with 0 are replaced by comparisons with $-\delta$, and study under which conditions φ_M^δ converges, in some sense, to a function f when $\delta \to 0$, in which case we say that f is δ-approximable. Of course there are several notions of δ-approximability, depending on the particular notion of convergence under examination.

It turns out that the functions f that are computably δ-approximable, that is, $\|\varphi_M^\delta - f\|_\infty \to 0$ in a "computably controlled" way, are exactly the Type 2 computable functions. Moreover, the functions that are pointwise δ-approximable, that is, for all $\boldsymbol{x}$ we have $\varphi_M^\delta(\boldsymbol{x}) \to f(\boldsymbol{x})$ as $\delta \to 0$, are exactly the functions computed by *weak Type 2 machines*, which we introduce following Freund's works on functions and fields defined by Turing machines [8]. In weak Type 2 machines the output tape is two-way, and the content of a cell can change for an arbitrary, but finite, number of times.

We then proceed to study some properties of various kinds of δ-approximable functions, obtaining a number of continuity and separation results. Finally, we highlight some possible extensions to suitable Archimedean fields, and list some open problems.

Other authors have studied related notions of "thresholding" of real random access machines: in particular, *feasible real random access machines* [5] were introduced to study the efficient emulation of real computations in a discrete fashion, whereas δ-**Q**-*analytic machines* [6] use infinite converging computations on more and more precise rational roundings of their inputs.

Suitable types of analytic machines compute classes of functions identified by older computational models, such as BSS computable, Type 2 computable, and weakly (in the sense of Freund) computable functions. Since two of the main classes defined in this paper turn out to coincide with the latter two, we obtain correspondingly an equivalence with classes of functions computed by suitable analytic machines. Note however that the accent in our treatment is on *equality*, whereas in analytic machines also operations are approximated. It is particularly interesting to note that the resulting classes are the same, as this shows once again that the main problem in making a real computation feasible is testing for exact equality, and not computing exact operations.

To make the paper as self-contained as possible, we introduced all models used and we give the basic definitions of degree theory we shall need to use.

2 The Computation Models

2.1 BSS Machines

A finite dimensional (normalized) BSS machine [1] is just a non-discrete version of a Random Access Machine: it takes inputs from $\mathbf{R}^n$ and produces outputs in $\mathbf{R}^m$, using a state space whose registers contain elements of $\mathbf{R}$. Informally, the program is described by a finite flowchart, where each non-final node is either a computation node or a branching node. Computation nodes have just one successor, and they are associated with a rational function of the state space into itself. Branching nodes have two successors, and the decision about which branch to take depends on whether the first coordinate x_1 of the state space is negative or not. The *constants* of a BSS machine are the coefficients of the rational functions appearing in its definition.

2.2 (Weak) Type 2 Turing Machines

The tape of an ordinary Turing machine is nonblank only on a finite number of cells, at any moment of a computation. Thus, in order to allow elements of $\mathbf{R}$ to be taken into consideration, one slightly generalizes the notion of machine. A (deterministic Type 2 [16]) Turing machine consists of

1. a finite number of read-only one-way input tapes (possibly none), each containing at start an infinite string belonging to $\{\bar{1}, 0, 1, .\}^\omega$ and describing an element of $\mathbf{R}$ *via* its signed binary digit representation;
2. a finite number of write-only one-way output tapes (possibly none), on which the machine is supposed to write representations of elements of $\mathbf{R}$;
3. some other work tapes, initially blank.

The finite control is defined as usual via a finite set of states and a transition function. The only differences with a standard Turing machine are the possibility of filling completely the input tapes, and of considering nonstopping machines as machines outputting elements of $\mathbf{R}$. Every Type 2 machine computes a function, which is defined for all inputs for which all cells of the output tapes are eventually written.

Finally, a *weak* Type 2 machine is defined by making the output tape two-way: however, we ask that in every computation the content of an output tape cell is changed a *finite* number of times, so every finite prefix of the output tape eventually stabilizes.

These machines have been studied by Freund [8], who showed that they are more powerful than standard Type 2 machines. To be true, the model used in the abovementioned reference has a single tape: nonetheless, it is not difficult to see that the single-tape restriction does not change the power of the model. Moreover, in [8] the function computed by weak Type 2 machines are defined using a positive notation. This could in fact make a difference, and this issue is discussed in Section 10.

It should be noted that we obtained the notion of weak computability using a modified output tape; however, exactly the same notion can be recovered as a particular instance of TTE, just by using a different representation for the output: instead of the standard Cauchy representation, given by sequences of fast converging rationals (which is equivalent to the binary digit representation), one uses the *naïve* Cauchy representation, given by sequences of converging rationals, but without any convergence speed guarantee. The resulting class of functions is exactly the class of functions computable by a weak Type 2 machine.

One property of weak Type 2 machines we shall use is that they can, in a sense, compute the limit of a (however slowly) converging sequence of dyadics:

Proposition 1. *There is a weak Type 2 machine*[1] *that receives in input a sequence* $\{d_k\}$ *of dyadics converging to* $\alpha \in \mathbf{R}$ *and outputs a signed binary representation of* α*.*

Proof. At round k, the machine produces a signed binary representation r_k of d_k in such a way that the length of the common prefix of r_k and r_{k-1} is maximized (r_{-1} is the empty string) and writes it on the output tape (of course the common digits need not be rewritten). The stabilisation property follows from the observation that given two dyadics d and d' such that $|d - d'| < 2^{-k}$ and a finite signed binary representation r of d there is always a finite representation of d' that shares at least k fractional digits with r. □

In the following we denote with $\mathscr{T}_2$ the set of Type 2 computable functions, and with $\mathscr{W}_2$ the set of weakly Type 2 computable functions. Clearly $\mathscr{T}_2 \subseteq \mathscr{W}_2$.

3 Degrees of Real Numbers and Jumps

In this section we recall some basic definitions from degree theory, in particular about degrees of real numbers.

A set $A \subseteq \mathbf{N}$ is recursive in $B \subseteq \mathbf{N}$ iff there is an oracle Turing machine that decides membership to A using B as an oracle; this relation is a preorder on the subsets of $\mathbf{N}$, and the equivalence classes induced by this preorder are called *(Turing) degrees of unsolvability* [11]; they are of course a partially ordered set $\mathscr{D}$ (the order relation being denoted by "$\leq$"), which possesses finite suprema denoted by "$\vee$"; the bottom element (corresponding to decidable sets) is denoted by $\mathbf{0}$. We write $\operatorname{dg} A$ for the degree of a subset A of $\mathbf{N}$.

Now consider a set $A \subseteq \mathbf{N}$; let μ_A be the least positive integer included in A (1 if A does not contain any positive integer), and let σ_A be either 1 or -1, depending on whether $0 \in A$ or not. Define

$$\rho(A) = \sigma_A \cdot \left(\mu_A - 1 + \sum_{\mu_A < i \in A} 2^{\mu_A - i} \right).$$

It should be clear that, for any nondyadic real number α, there exists exactly one set in $\rho^{-1}(\alpha)$ (which is neither finite nor cofinite), and we define the degree of α, denoted by $\operatorname{dg}\alpha$, as the degree of unsolvability of $\rho^{-1}(\alpha)$ [12,10,3]; moreover, we let $\operatorname{dg}\alpha = \mathbf{0}$ for every dyadic rational α. When the distinction is irrelevant, we shall confuse real numbers, subsets of $\mathbf{N}$ and degrees, omitting the map ρ; this will happen in particular

[1] The reader should observe that the input of this machine is not a sequence of signed binary digits, but rather a coded sequence of dyadics.

when using real numbers as oracles, or when specifying an arbitrary real number of given degree; note that in particular it is equivalent to think of a Turing machine as using oracles $\alpha_1, \ldots, \alpha_r$ or the single oracle $\alpha_1 \vee \cdots \vee \alpha_r$.

The last concept we need from degree theory is the notion of a *jump*. Given a degree $\boldsymbol{d} \in \mathscr{D}$, we can consider the set B that encodes the halting of the universal Turing machine relativized to (any set belonging to) $\boldsymbol{d}$; one defines $\boldsymbol{d}' = \operatorname{dg} B$, where $\boldsymbol{d}'$ is called the *jump of* $\boldsymbol{d}$. Note that one has $\boldsymbol{d}' > \boldsymbol{d}$ for all $\boldsymbol{d} \in \mathscr{D}$.

4 δ-approximable Functions

Given a BSS machine M and a $\delta \geq 0$ (called a *threshold*), we define the δ-computing endomorphism much as in the classical case [1], but substituting the test case as follows [2]:

$$\langle q, \boldsymbol{x} \rangle \mapsto \begin{cases} \langle \beta^-(q), \boldsymbol{x} \rangle & \text{if } x_1 < -\delta \\ \langle \beta^+(q), \boldsymbol{x} \rangle & \text{if } x_1 \geq -\delta \end{cases} \qquad \text{if } q \text{ is a branching node.}$$

This induces a δ-halting set (denoted by Ω_M^δ) and a δ-computed function φ_M^δ.

Definition 1. *Given a BSS machine M with input space $\mathbf{R}^n$ and output space $\mathbf{R}^m$, its* δ-convergence set $\Omega_M^\downarrow$ *is defined as*

$$\Omega_M^\downarrow = \left\{ \boldsymbol{x} \in \mathbf{R}^n \;\middle|\; \varphi_M^\delta(\boldsymbol{x}) \textit{ has a limit in } \mathbf{R}^m \textit{ as } \delta \to 0 \right\}.$$

Intuitively, the convergence set is the set of inputs for which better precision (i.e., smaller thresholds) provides better outputs. We are now ready to state our main

Definition 2. *Given a BSS machine M with input space $\mathbf{R}^n$ and output space $\mathbf{R}^m$ and a function $f : \Omega_M^\downarrow \subseteq \mathbf{R}^n \to \mathbf{R}^m$, we say that M*

- pointwise δ-*approximates f iff for every $\boldsymbol{x} \in \Omega_M^\downarrow$*

$$\left\| \varphi_M^\delta(\boldsymbol{x}) - f(\boldsymbol{x}) \right\| \to 0 \qquad \textit{as } \delta \to 0;$$

- uniformly δ-*approximates f iff*

$$\sup_{\boldsymbol{x} \in \Omega_M^\downarrow} \left\| \varphi_M^\delta(\boldsymbol{x}) - f(\boldsymbol{x}) \right\| \to 0 \qquad \textit{as } \delta \to 0;$$

- computably δ-*approximates f iff there exists a recursive monotonic positive function $\varepsilon : \mathbf{Q} \to \mathbf{Q}$ such that $\varepsilon(\delta) \to 0$ as $\delta \to 0$, and for all $\delta \in \mathbf{Q} \cap (0, 1)$*

$$\sup_{\boldsymbol{x} \in \Omega_M^\downarrow} \left\| \varphi_M^\delta(\boldsymbol{x}) - f(\boldsymbol{x}) \right\| \leq \varepsilon(\delta).$$

We denote with $\mathscr{A}_p$ ($\mathscr{A}_u$, $\mathscr{A}_c$) the class of pointwise (uniformly, computably, resp.) δ-approximable functions.

Some remarks are in order. The first definition is the weakest possible, as it just requires that for every input the computation converges, but we have no hints on the global behaviour of the approximation process. The second definition makes convergence uniform, but we have in principle no computable guarantee of the rate of convergence. Finally, in the last case we have very precise guarantee—given a rational threshold δ, we can recursively compute a guaranteed precision for the output. Clearly, $\mathscr{A}_c \subseteq \mathscr{A}_u \subseteq \mathscr{A}_p$.

5 δ-approximability vs. Turing Computability

Our first lemma records that for a fixed input there are arbitrarily small thresholds that give rise to an acceptance path in which all tests result in a strict inequality.

Lemma 1. *Let M be a BSS machine, $\boldsymbol{x} \in \Omega_M^{\downarrow}$, and $\alpha > 0$. Then there exists a dyadic $\delta \in (0, \alpha)$ such that $\boldsymbol{x} \in \Omega_M^{\delta}$ and all tests along the accepting path of $\boldsymbol{x}$ for threshold δ are strict inequalities.*

Proof. First, since $\boldsymbol{x} \in \Omega_M^{\downarrow}$, there is $\delta_0 \in (0, \alpha)$ such that $\boldsymbol{x} \in \Omega_M^{\delta_0}$. Let $\mathscr{P}^-$ be the (finite) set of polynomials evaluated negatively in the acceptance path of $\boldsymbol{x}$ for threshold δ_0, and let $\delta_1 = \min\{-P(\boldsymbol{x}) \mid P \in \mathscr{P}^-\}$. Since $P(\boldsymbol{x}) < -\delta_0$ holds for all $P \in \mathscr{P}^-$, we have $\delta_1 > \delta_0$, and for any threshold in (δ_0, δ_1) all tests along the accepting path of $\boldsymbol{x}$ are strict inequalities. $\square$

We are now ready to prove the first equivalence result between BSS approximation and Turing machines (clearly, all such results must be relativised so as to be dependent on the constants appearing in the BSS machines, which must be available as oracles to the corresponding Type 2 machines). The following theorem highlights the analogy between unrestricted pointwise convergence and the possibility given to a weak Type 2 machine to "change its mind" a finite number of times about the content of an output cell:

Theorem 1. *A function is weakly Type 2 computable iff it is pointwise δ-approximable, that is, $\mathscr{W}_2 = \mathscr{A}_p$.*

Proof. For the left-to-right implication, let f be a function computed by a weak Type 2 machine T. Recalling from [2] that BSS machines can δ-uniformly perform discrete computations[2], we can build a BSS machine M that pointwise δ-approximates f by first computing an integer k such that $2^{-k} < \delta$ (by finding the first k such that -2^{-k} does not appear to be negative), and then emulating k steps of the computation performed by T on a δ-uniformly generated signed binary expansion of the input. Since, for a given i, there is a k such that the first i digits written by T after k steps of computation are correct, we have $\varphi_M^{\delta}(\boldsymbol{x}) \to f(\boldsymbol{x})$ as $\delta \to 0$.

Conversely, let f be a function pointwise δ-approximated by a BSS machine M. It is possible for a Type 2 machine, given an integer k, to compute $\varphi_M^{\delta}(\boldsymbol{x})$ up to the first k fractional digits, for some $\delta \in (0, 2^{-k})$. Indeed, such a machine can clearly compute

[2] Note that the resulting machine is not necessarily δ-uniform: we are just exploiting δ-uniformity of the emulation to make it independent of the threshold.

$\varphi_M^\delta(\boldsymbol{x})$ with any given precision, provided that all tests along the accepting path of $\boldsymbol{x}$ for threshold δ are strict inequalities, so all we have to do is dovetail these computations for all dyadic thresholds in $(0, 2^{-k})$ and, due to Lemma 1, one of the computations will necessarily halt. We thus obtain a sequence of dyadics (indexed by k) converging to $f(\boldsymbol{x})$ that can be fed into the machine of Proposition 1. This completes the proof. $\square$

If we want to obtain true, strong Type 2 computability we must ensure that the rate of convergence is computably controlled:

Theorem 2. *A function is Type 2 computable iff it is computably δ-approximable, that is, $\mathscr{T}_2 = \mathscr{A}_c$.*

Proof. For the left-to-right implication, let f be a function computed by a Type 2 machine T. Let M be the BSS machine that computes a k such that $2^{-k} < \delta$ (see proof of Theorem 1), and then emulates the computation of T so as to compute the first k fractional digits of $f(\boldsymbol{x})$. We then have $\left\|\varphi_M^\delta(\boldsymbol{x}) - f(\boldsymbol{x})\right\| \leq 2^{-k} < \delta$.

Conversely, let f be a function computably δ-approximated by a BSS machine M. We use the same Type 2 machine as in the proof of Theorem 1, but for each integer k we choose a δ_k so that $\varepsilon(\delta_k) < 2^{-k}$ and dovetail the computation of $\varphi_M^\delta(\boldsymbol{x})$ up to k fractional digits for all dyadic $\delta \in (0, \delta_k)$. Let $\boldsymbol{y}$ be the first output obtained, say with threshold δ. Since $\left\|\boldsymbol{y} - \varphi_M^\delta(\boldsymbol{x})\right\| \leq 2^{-k}$ and

$$\left\|\varphi_M^\delta(\boldsymbol{x}) - f(\boldsymbol{x})\right\| \leq \varepsilon(\delta) \leq \varepsilon(\delta_k) < 2^{-k},$$

we have $\|\boldsymbol{y} - f(\boldsymbol{x})\| < 2^{-(k-1)}$, that is, for all greater k's the first $k-1$ signed digits can be taken to be the same, which implies that those digits can now be output by T. $\square$

6 Some Properties of δ-approximable Functions

The following theorem answers a most natural question: what is the relation between the functions *computed* and *pointwise δ-approximated* by a BSS machine?

Theorem 3. *Let M be a BSS machine that pointwise δ-approximates a function f. Then f extends φ_M.*

Proof. Indeed, for $\boldsymbol{x} \in \Omega_M$, let $\mathscr{P}^-$ be the set of polynomials evaluated negatively along the acceptance path of $\boldsymbol{x}$ for threshold 0, and $\delta_1 = \min\{-P(\boldsymbol{x}) \mid P \in \mathscr{P}^-\}$. Then, for any threshold $\delta \in (0, \delta_1)$ the acceptance path of $\boldsymbol{x}$ is unchanged, so $\varphi_M^\delta(\boldsymbol{x}) = \varphi_M(\boldsymbol{x})$. Therefore, we have $\varphi_M^\delta(\boldsymbol{x}) \to \varphi_M(\boldsymbol{x})$ as $\delta \to 0$. $\square$

Note that in general f will be defined on more points than φ_M. Indeed, for the BSS machines approximating Type 2 computable functions that we used in the previous proofs it is often the case that φ_M is everywhere undefined whereas f is defined on a significant set of inputs.

Another interesting property follows from uniformity:

Theorem 4. *A uniformly δ-approximable function is continuous.*

Proof. Let $f \in \mathscr{A}_u$ and let M be a BSS machine that δ-approximates it. Given an $\varepsilon > 0$, the uniform convergence of φ_M^δ to f ensures the existence of an α such that for all $\delta \in (0, \alpha)$ and for all $\boldsymbol{y} \in \operatorname{dom} f$ we have $\left\|\varphi_M^\delta(\boldsymbol{y}) - f(\boldsymbol{y})\right\| < \frac{\varepsilon}{4}$. For every $\boldsymbol{x} \in \operatorname{dom} f$, Lemma 1 allows us to choose δ so that all tests along the accepting path of $\boldsymbol{x}$ for threshold δ are strict inequalities. This, by continuity of the tested polynomials, implies that a whole open neighbourhood of $\boldsymbol{x}$ is accepted along the same accepting path. Over this neighbourhood, φ_M^δ is a rational function, so it is continuous, that is, there is an open neighbourhood U of $\boldsymbol{x}$ such that for all $\boldsymbol{y} \in U$ we have $\left\|\varphi_M^\delta(\boldsymbol{y}) - \varphi_M^\delta(\boldsymbol{x})\right\| < \frac{\varepsilon}{2}$. But then $\| f(\boldsymbol{x}) - f(\boldsymbol{y})\| < \varepsilon$ holds for all $\boldsymbol{y} \in U \cap \operatorname{dom} f$. ◻

Of course, this is *a fortiori* true of computably δ-approximable functions.

7 Separation Results

In this section we show that the δ-approximability classes considered so far are all (essentially) separated. In lights of our previous results, this also shows that $\mathscr{T}_2 \subset \mathscr{W}_2$.

Theorem 5. *There are pointwise δ-approximable functions that are not uniformly δ-approximable, that is, $\mathscr{A}_u \subset \mathscr{A}_p$.*

Proof. Let $f : \mathbf{R} \to \mathbf{R}$ be defined by

$$f(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

f is clearly BSS computable, and therefore pointwise δ-approximable by Theorem 3. However, since f is not continuous, by Theorem 4 it cannot be uniformly δ-approximable. ◻

A similar result for $\mathscr{A}_c$ and $\mathscr{A}_u$ is not possible in the same form: indeed, we can always add to a BSS machine a constant containing an oracle that makes it possible to computably control the convergence rate. However, if we fix the set of constants involved this is no longer true:

Theorem 6. *There are functions uniformly δ-approximable that are not computably δ-approximable using the same set of constants.*

Proof. We prove just the case without constants; it is easy to relativise the proof to any given set of constants. Let $H \subseteq \mathbf{N}$ be the set of numbers (seen as pairs by Cantor pairing) over which the universal recursive function is defined. Let $f : \mathbf{R} \to \mathbf{R}$ be the constant function $f(x) = h$, where

$$h = \sum_{i \in H} 2^{-(i+1)}.$$

We can δ-approximate f with a BSS machine as follows: let $a_1, \dots, a_k$ be the first k elements of a recursive enumeration of H, where k is the first integer such that $2^{-k} < \delta$, and output $\sum_{i=1}^k 2^{-(a_i+1)}$. When $\delta \to 0$, we have $k \to \infty$ and then it is easy to see that $\sum_{i=1}^{k} 2^{-(a_i+1)} \to h$. Furthermore, the convergence of φ_M^δ to f is uniform, since all functions involved are constant. Therefore f is uniformly δ-approximable (without constants).

Suppose now by contradiction that f is computably δ-approximable (without constants). According to Theorem 1, there is a Type 2 machine M that computes f. Since f is constant, we can derive from M a classical Turing machine M', which on input i returns the i-th fractional digit of the positive binary representation of h (note that this can be done because there is a Type 2 machine that, for any non-dyadic input, outputs the corresponding positive representation [2]). Thus, M' decides the halting problem, which is impossible. $\square$

8 Generalizations: (Weak) Turing Closures

In generalizing the results above to an arbitrary Archimedean field R (such fields can be identified with subfields of the reals), one is immediately confronted with the possibility that $\varphi_M^\delta(\boldsymbol{x})$ has the Cauchy property but does not converge in R (i.e., it really converges to an element of $\mathbf{R} \setminus R$) as $\delta \to 0$ for some $\boldsymbol{x}$.

This is an unpleasant situation, as it would make the δ-approximated function undefined on some values over which, in any reasonable sense, the function is really converging.

The solution to this problem is to restrict the Archimedean fields suitably, so as to guarantee that being Cauchy (i.e., converging in $\mathbf{R}$) always implies convergence in R. The relevant notion to give sufficient conditions in this sense has been introduced in [3]:

Definition 3. *Let R be an Archimedean field, and $R \subseteq F$ a field extension with F Archimedean. An element $x \in F$ is said to be* Turing over *R iff there are $n \in \mathbf{N}$, $\boldsymbol{y} \in R^n$ and a Turing (Type 2) machine M (with n input tapes) such that $M(\boldsymbol{y}) = x$. If every $a \in F$ is Turing over R, then $R \subseteq F$ is said to be a* Turing extension *of R. A field R is* Turing closed *iff it does not have any proper Turing extension. The* Turing closure *of a field R is the intersection of all Turing closed fields containing R, and will be denoted*[3] *by $T(R)$.*

Of course, one can restate the previous definition using *weak* Type 2 machines. In this case we will talk of the *weak Turing closure* of R, and denote it with $W(R)$. Clearly, the smallest weakly Turing closed field is $\mathbf{W} = W(\mathbf{Q})$, the weak Turing closure of the rationals.

We shall prove that weak Turing closures can be reduced to suitably iterated Turing closures. To this purpose, we first prove a limitation on the degree of the numbers produced by a weak Turing machine:

Proposition 2. *Let α be a real number output by a weak Type 2 machine M on inputs $\alpha_1, \dots, \alpha_r$. Then* $\operatorname{dg} \alpha \leq (\alpha_1 \vee \cdots \vee \alpha_r)'$.

Proof. For each $j, k \in \mathbf{N}$ let S_{jk} be the Turing machine (with oracles $\alpha_1, \dots, \alpha_r$) that emulates M until the j-th output cell has been changed k times and then stops. Of

[3] We use also $T(S)$ when S is an arbitrary subset of $\mathbf{R}$, with the same meaning.

course, for each j there is a $k > 1$ such that S_{jk} does not stop. If we have $(\alpha_1 \vee \cdots \vee \alpha_r)'$ as an oracle, for each j we can clearly compute the least such k, emulate M up to the $(k-1)$-th cell change and output the resulting value. This gives as a result a (non-weak) Type 2 machine with oracle $(\alpha_1 \vee \cdots \vee \alpha_r)'$ that outputs α, hence the thesis. $\square$

We can finally prove our characterization:

Theorem 7. *Let us define $\tau(R) = T(R')$ for every Archimedean field R, where $R' = \{\alpha' \mid \alpha \in R\}$ is the set of reals that are one jump over those in R. Then*

$$W(R) = \bigcup_{k \in \mathbf{N}} \tau^k(R).$$

Proof. Since there is a weak Type 2 machine that outputs $(\alpha_1 \vee \cdots \vee \alpha_r)'$ on inputs $\alpha_1, \ldots, \alpha_r$ (start writing out zeroes and change them to ones when the emulation of the corresponding machine stops), every weakly Turing closed extension of R is to contain $\bigcup_{k \in \mathbf{N}} \tau^k(R)$. On the other hand, we show that the last field is weakly Turing closed, thus proving the statement. Indeed, if α is the output of a weak Type 2 machine on inputs $\alpha_1, \ldots, \alpha_r \in \bigcup_{k \in \mathbf{N}} \tau^k(R)$, then there is a j such that $\alpha_1, \ldots, \alpha_r \in \tau^j(R)$, and by Proposition 2 we have $\operatorname{dg} \alpha \leq (\alpha_1 \vee \cdots \vee \alpha_r)'$, so $\alpha \in \tau^{j+1}(R)$. $\square$

We can finally state the main theorem of this section; we assume to have restated Definition 1 and 2 replacing $\mathbf{R}$ with R.

Theorem 8. *Let R be a weakly Turing closed field and M be a BSS machine on R. Then every input $\boldsymbol{x}$ for which $\varphi_M^\delta(\boldsymbol{x})$ is Cauchy belongs to $\Omega_M^\downarrow$.*

Proof. Let $\boldsymbol{x}$ be an input such that $\varphi_M^\delta(\boldsymbol{x})$ is Cauchy as $\delta \to 0$. Then $\varphi_M^\delta(\boldsymbol{x}) \to \alpha \in \mathbf{R}$. By Theorem 1, there is a weak Type 2 machine with input $\boldsymbol{x}$ that outputs α, so $\alpha \in W(R) = R$. $\square$

The previous theorem is clearly the best possible: indeed, if R is not weakly Turing closed then there is a weak Type 2 machine that outputs an element not in R; hence, by Theorem 1 there is a BSS machine M such that for each input x we have that $\varphi_M^\delta(x)$ is Cauchy, but it does not converge in R.

A computable version of the previous theorem can be stated when the Cauchy property can be computably controlled:

Theorem 9. *Let R be a Turing closed field and M be a BSS machine on R. Then every input $\boldsymbol{x}$ for which $\varphi_M^\delta(\boldsymbol{x})$ is computably Cauchy*[4] *belongs to $\Omega_M^\downarrow$.*

In particular, this applies to computably δ-approximable functions: if φ_M^δ is computably Cauchy (in the obvious sense), then $\Omega_M^\downarrow$ coincides with the set of inputs $\boldsymbol{x}$ that make $\varphi_M^\delta(\boldsymbol{x})$ Cauchy.

[4] That is, there is a recursive function $\delta : \mathbf{Q} \to \mathbf{Q}$ such that for all rational $\varepsilon > 0$ and all $\delta_1, \delta_2 \leq \delta(\varepsilon)$ we have $\left\| \varphi_M^{\delta_1}(\boldsymbol{x}) - \varphi_M^{\delta_2}(\boldsymbol{x}) \right\| \leq \varepsilon$.

9 A remark on signedness

In this section we briefly discuss the impact on signed vs. unsigned representations in weak Type 2 computability of functions. The following proposition holds:

Proposition 3. *There is a weak Type 2 machine that converts from signed binary to positive binary notation, that is, a machine that computes the identity function and outputs only positive digits.*

Proof. We give a proof for the case of positive numbers; it can be easily generalized to arbitrary numbers. The machine works as follows: first of all, it converts and outputs the integral part. Thereafter, it behaves as follows:

1. on reading a nonnegative digit, it copies it;
2. on reading a negative digit, if the current output ends with a suffix of the form 10^k it changes it into 01^{k+1}. Otherwise, say if the fractional part is 0^k, it decrements the integral part and changes the fractional part to 1^{k+1}.

It is straightforward to see that the value of the current output always corresponds to the input read so far.

We now prove by induction that the first k digits after the fractional point eventually stabilize. For $k = 0$ this is true because the integral part cannot be changed more than twice (it can only get decremented). Assume that the first k digits stabilize after t steps. Then after step t the $(k + 1)$-th digit can only change from one to zero, as changing from zero to one would imply modifying the previous digits. Since only one change is possible, also the first $k + 1$ digits eventually stabilize. ◻

The previous proposition may suggest that weak Type 2 computability is independent of signedness of the representation used. However, it is not possible, in general, to compose weak Type 2 machines (this is implied by Proposition 2), and in particular Vasco Brattka [4] has shown that it is not true in this case:

Proposition 4 (Vasco Brattka). *There is no weak Type 2 machine that receives in input a sequence $\{d_k\}$ of dyadics converging to $\alpha \in \mathbf{R}$ and outputs a* positive *binary representation of α.*

Proof. Let M be such a machine, and $\{a_k\}$ be a sequence of rationals such that $a_{2i} < 1 < a_{2i+1}$ for all $i \in \mathbf{N}$ and $a_k \to 1$ as $k \to \infty$.

The machine M, when fed with the constant sequence $a_0, a_0, a_0, \ldots$, must eventually write `0.` on its output tape. Let t_0 be the number of input elements of the sequence read when this happens. Now, consider the sequence starting with t_0 repetitions of a_0 and then extended indefinitely with a_1. There is a $t_1 \in \mathbf{N}$ such that after t_1 copies of a_1 have been read, the machine changes its mind and replaces the start of the output tape with `1.`. Consider now the sequence starting with t_0 repetitions of a_0, t_1 repetitions of a_1 and extended indefinitely with a_2. Again, there is a $t_2 \in \mathbf{N}$ such that after t_2 copies of a_2 have been read, the machine will change again its mind and replace the start of the output tape with `0.`. Continuing this way, we obtain an input sequence of rationals converging to 1 that forces M to rewrite infinite times the content of the first cell, a contradiction. ◻

10 Open Problems

Theorem 3 highlights an interesting property: a BSS machine δ-approximates an extension of its computed function φ_M. When M δ-approximates *exactly* φ_M we say that φ_M is computable *with infinite precision*; the class of functions that are δ-approximable with infinite precision is denoted then by $\mathscr{A}_p^\infty$, and analogously we can define $\mathscr{A}_u^\infty$ and $\mathscr{A}_c^\infty$.

Several questions arise in this case: first of all, we would like to study separations for the inclusions $\mathscr{A}_c^\infty \subseteq \mathscr{A}_u^\infty \subseteq \mathscr{A}_p^\infty$, and for the inclusions $\mathscr{A}_{\_}^\infty \subseteq \mathscr{A}_{\_}$ (the case $\mathscr{A}_p$ is solved by Theorem 3). Of course in this case the question should be posed modulo extensions. However, in the following inclusion diagram

$$\begin{tikzcd}
\mathscr{A}_c^\infty \arrow[r] \arrow[d] & \mathscr{A}_u^\infty \arrow[r] \arrow[d] & \mathscr{A}_p^\infty \arrow[r] \arrow[d] & \mathrm{BSS} \\
\mathscr{T}_2 = \mathscr{A}_c \arrow[r] & \mathscr{A}_u \arrow[r] & \mathscr{A}_p = \mathscr{W}_2 &
\end{tikzcd}$$

all inclusions are strict except possibly for $\mathscr{A}_c^\infty \subseteq \mathscr{A}_u^\infty$ and $\mathscr{A}_p^\infty \subseteq \mathrm{BSS}$: indeed, all vertical inclusions are strict for otherwise $\mathscr{T}_2 \subseteq \mathrm{BSS}$, and the lower line separation results are Theorem 5 and 6; moreover, Theorem 5 can be adapted to show that $\mathscr{A}_u^\infty \subset \mathscr{A}_p^\infty$. Finally, one can ask whether it happens that $\mathscr{A}_{\_} \cap \mathrm{BSS} = \mathscr{A}_{\_}^\infty$ (Theorem 3 just gives a partial answer for the pointwise case).

Some knowledge should also be gained about **W**, the weak Turing closure of **Q**, which plays for weak Turing machines the same role played by **T**, the Turing closure of **Q**, in the standard case. Certainly **W** is countable, it contains **T**, and since it is Turing closed, it is also real closed, but little else is known. Note that **W** is much larger than the field of weakly Turing computable numbers defined in [8].

References

1. 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. *Bull. Amer. Math. Soc. (N.S.)*, 21:1–46, 1989.
2. Paolo Boldi and Sebastiano Vigna. δ-uniform BSS machines. *J. Complexity*, 14(2):234–256, 1998.
3. Paolo Boldi and Sebastiano Vigna. The Turing closure of an Archimedean field. *Theoret. Comput. Sci.*, 231:143–156, 2000.
4. Vasco Brattka. Personal electronic communication, 2001.
5. Vasco Brattka and Peter Hertling. Feasible real random access machines. *J. Complexity*, 14(4):490–526, 1998.
6. Thomas Chadzelek and Günter Hotz. Analytic machines. *Theoret. Comput. Sci.*, 219(1–2):151–167, 1999.
7. Abbas Edalat and Philipp Sünderhauf. A domain-theoretic approach to computability on the real line. *Theoret. Comput. Sci.*, 210(1):73–98, 1999.
8. Rudolf Freund. Real functions and numbers defined by Turing machines. *Theoret. Comput. Sci.*, 23(3):287–304, May 1983.

9. Armin Hemmerling. On approximate and algebraic computability over the real numbers. *Theoret. Comput. Sci.*, 219:185–223, 1999.
10. Ker-I Ko. Reducibilities on real numbers. *Theoret. Comput. Sci.*, 31(1–2):101–123, May 1984.
11. Joseph R. Shoenfield. *Degrees of Unsolvability*. North–Holland, Amsterdam, 1971.
12. Robert I. Soare. Recursion theory and Dedekind cuts. *Trans. Amer. Math. Soc.*, 140:271–294, 1969.
13. John V. Tucker and Jeffery I. Zucker. Computation by 'While' programs on topological partial algebras. *Theoret. Comput. Sci.*, 219(1–2):379–420, 1999.
14. Sebastiano Vigna. On the relations between distributive computability and the BSS model. *Theoret. Comput. Sci.*, 162:5–21, 1996.
15. Robert F.C. Walters. An imperative language based on distributive categories. *Math. Struct. Comp. Sci.*, 2:249–256, 1992.
16. Klaus Weihrauch. *Introduction to Computable Analysis*. Texts in Theoretical Computer Science. Springer–Verlag, 2000.