ELSEVIER

Theoretical Computer Science 219 (1999) 185–223

Theoretical Computer Science

www.elsevier.com/locate/tcs

On approximate and algebraic computability over the real numbers

Armin Hemmerling *

Ernst-Moritz-Arndt–Universität Greifswald, Institut für Mathematik und Informatik, F.-L.-Jahn Str. 15a, D-17487 Greifswald, Germany

Abstract

We consider algebraic and approximate computations of (partial) real functions $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$. Algebraic computability is defined by means of (parameter-free) finite algorithmic procedures. The notion of approximate computability is a straightforward generalization of the Ko–Friedman approach, based on oracle Turing machines, to functions with not necessarily recursively open domains.

The main results of the paper give characterizations of approximate computability by means of the passing sets of finite algorithmic procedures, i.e., characterizations from the algebraic point of view. Some consequences and also modifications of the concepts are discussed. Finally, two variants of arithmetical hierarchies over the reals are considered and used to classify and mutually compare the domains, graphs and ranges of algebraically resp. approximately computable real functions.

Keywords: Computable real function; Computable real number; Abstract (algebraic) computability; Approximate computability; Arithmetical hierarchies over the reals

1991 MSC: 03D65, 03F60, 03D10, 03D55, 26E40, 68Q05

1. Introduction

The intuitive notion of computability over the natural numbers and other discrete sets of objects has a well-defined and generally accepted meaning. This is usually expressed by Church's Thesis. On this basis, classical theory of computability (as well as of computational complexity) deals with fundamental abilities and limits of real world computers applied to discrete problems.

With respect to the ordered field of real numbers, "perhaps mathematics' most basic structure" [6], the situation is different. There is a variety of approaches, each with its own motivations, methods and results, which are partially incomparable among each

* E-mail: hemmerli@rz.uni-greifswald.de.

0304-3975/99/$ - see front matter
PII: S0304-3975(98)00289-8

other. In this paper, we consider two fundamental points of view which lead to different variants of computability over the reals. It should be noticed that some features of these two positions once more reflect the old conflict between the view of the infinite as only potential or constructive and its treatment as actual or existential. The former position goes back to Aristotle, the latter one was taken by Cantor, cf. [13, 19].

The first point of view is based on the postulate that only discrete, finitary entities can be stored in and processed by physical devices. Therefore, to apply computations to continuous objects like the real numbers, these objects have to be represented by discrete ones. Of course, such representations can only be approximate, with a certain degree of inaccuracy. Nevertheless, real numbers can be named by (infinite) sequences of discrete objects which converge to them, and classical discrete computation devices can be used to perform calculations over the reals by means of such approximating sequences. There are several implementations of this basic idea, starting with Turing's seminal contribution [37]. Throughout the present paper, by *approximate computability*, in the narrower sense we understand that concept which has more recently been defined and investigated by Ko–Friedman, Weihrauch and others, see [21, 20, 38, 39] for detailed presentations of the origins and the state of the art.

For the second group of approaches, we imagine a programmable calculator able to store and to process arbitrary real numbers with absolute precision. Its basic operations consist in assigning, to some destination register, either a base constant (0 or 1) or the result of a base operation $(+, -, \cdot, /)$ applied to the contents of source registers. The execution of jumps will depend on the validity of an order relation ($<$ or $\leqslant$) between the contents of two registers. Moreover, a halt instruction is needed. The "finite algorithmic procedures" [9] built in this way are already sufficient to perform all intuitively imaginable algebraic computations in the ordered field of real numbers.

Obviously, this is only a special case of a straightforward concept of computability over an arbitrary algebraic structure given by a universe of objects and by base costants, operations and relations. In order to enable a program to deal with arbitrarily many registers, some kind of indirect addressing is additionally needed. These and more or less related notions of *algebraic computability* were developed and investigated by several authors, first explicit presentations are by Janov [17] and Engeler [7]. For further discussions, we refer to [1, 8, 9, 11, 18, 31, 36].

From the algebraic (or formalistic) point of view, it is less interesting if there exist physical devices that perform operations or decisions between real numbers with exact precision. There are rich theories of general program schemes [23] and of computability over general data structures [34], computational geometry [28] is widely based on the model of real RAM, and the BSS theory of computability and complexity over the reals [2, 1] produced many interesting results and put new questions. On the other hand, digital computers surely can process only discrete objects. Thus, one could conclude that only the approximate point of view can lead to a genuine theory of computability and complexity over the reals. So either of the two approaches has its own justification, advantages and also disadvantages.

At first glance, the two settings seem to be rather contrary to each other. They use different tools and methods, and the results and problems of one of the both are often not meaningful or even not expressible within the framework of the other one. Moreover, the communities of the followers were nearly disjoint for a time. Up to few years ago, Friedman [9, 21] and Shepherdson [30–32] were the only authors who had worked in both directions.

More recently, Weihrauch, Hertling and Brattka [12, 4] made some attempts in studying the real RAM model from the approximate point of view or to modify it in order to get a closer relationship to approximate computability, also with respect to time-complexity measures. Meyer auf der Heide and Wiedermann [24] proposed a still more realistic RAM model which processes floating point representations of some precision instead of real numbers. Hotz et al. [16, 15] used infinite converging computations on BSS-like machines with rounding operations in order to approximate real functions. Boldi and Vigna [3] studied semi-decidable subsets of the reals by using both approximately working Turing machines and special algebraically working BSS machines. The recent contribution by Zhong [40] considers semi-decidability from the two points of view, too. For a general, alternative approach based on partial algebras, the reader is referred to Tucker–Zucker [35].

In the present paper, we show how approximate computability can be defined and investigated by means of algebraic machines or programs. It will even turn out that the approximate and the algebraic computability are directly complementary to each other. They simply represent the two sides of the same coin: whereas algebraic computability is defined by means of the finite, halting computations, approximate computability is related to the infinite, never halting computations.

More precisely, we consider parameter-free algebraic programs (finite algorithmic procedures, briefly: FAPs) over the ordered field of real numbers. If we restrict the partial real functions $f:\mathbb{R}^d \rightarrowtail \mathbb{R}$ which are computable by such programs to inputs of rational numbers, we obtain just those functions $f_0:\mathbb{Q}^d \rightarrowtail \mathbb{Q}$ which are classically computable (via some standard encoding of the rational numbers). Thus, we deal with a rather natural concept of algebraic computability. Of course, it violates a fundamental thesis of approximate computability which says that all computable functions are continuous.

In order to characterize approximate computability, however, we only have to consider the infinite, never halting computations of algebraic programs. Indeed, any approximately computable function f can straightforwardly be obtained from the $(d+1)$-dimensional *passing set* of a (robust) FAP. This is the set of all real $(d+1)$-tuples which do not belong to the program's halting set. It will turn out that f is approximately computable if and only if its graph, i.e., the set $\{(r_1,\ldots,r_d,r_0)\in\mathbb{R}^{d+1}: f(r_1,\ldots,r_d)=r_0\}$, coincides with such a $(d+1)$-dimensional passing set considered as a relation from $\mathbb{R}^d$ to $\mathbb{R}$ and restricted to those d-tuples on which it is unique. So we obtain a characterization of approximate computability from the algebraic point of view. In particular, it does not depend on any naming system or representation of the reals by (sequences of) finitary objects.

The paper is organized as follows. In Sections 2 and 3, we recall and modify suitably the fundamentals concerning algebraic and approximate computability, respectively, and introduce basic notations and techniques used in the sequel. Approximate computability is here defined by generalizing the Ko–Friedman approach to functions with not necessarily recursively open domains. Section 4 presents a first characterization of approximate computability by means of FAPs. Section 5 discusses the concepts and results and shows some consequences and further relationships. In Section 6, we introduce two variants of arithmetical hierarchy over the reals and apply them to classify and compare mutually the domains, graphs and ranges of computable functions. We close with some final remarks.

2. Algebraic computability

The reader is supposed to be familiar with the basic notions and results of classical recursion theory. This is the theory of computability concerning functions in the natural numbers, $f:\mathbb{N}\rightarrowtail\mathbb{N}$, word functions $f:A^*\rightarrowtail A^*$, for (always finite) alphabets A, or functions of more complicated and even of mixed types like $f:\mathbb{N}^{d_1}\times(A^*)^{d_2}\rightarrowtail\mathbb{N}^{d_3}$, for some $d_1,d_2,d_3\in\mathbb{N}_+$, and so on. In the sequel, these fundamentals will be applied in a rather informal manner. Moreover, we shall deal with several objects, like formulas, tuples of rational numbers, rational intervals, etc., which are straightforwardly encodable by natural numbers or words. Thus, the classical theory of computability can be applied to those domains via the corresponding encodings. To improve the readability of this paper, we then often do not explicitly distinguish between the objects and their codes.

We want to study computability in the ordered field of real numbers,

$$\mathscr{R}=\langle\mathbb{R};\ 0,1;+,-,\cdot,/;\leqslant\rangle,$$

where the division is simply thought to be a total operation: let $r/0=0$.

Formulas of the first-order language with equality over $\mathscr{R}$ (briefly: FO-formulas) will be denoted by Greek letters like $\varphi,\psi,\ldots$. If φ contains (at most) the free variables $x_1,\ldots,x_d$, we also write $\varphi(x_1,\ldots,x_d)$ or $\varphi=\varphi(x_1,\ldots,x_d)$. By replacing simultaneously variables x_i by numbers $r_i\in\mathbb{R}$, for $1\leqslant i\leqslant d'$ with some $d'\leqslant d$, from φ the *quasi-formula* $\varphi(r_1,\ldots,r_{d'},x_{d'+1},\ldots,x_d)$ is obtained. This can also be considered to be an FO-formula over the extended structure $\overline{\mathscr{R}}=\langle\mathbb{R};\mathbb{R};+,-,\cdot,/;\leqslant\rangle$. Notice, however, that $\overline{\mathscr{R}}$ is a structure of infinite signature. For $\mathscr{R}$-terms $t=t(x_1,\ldots,x_d)$, we use the corresponding symbolism. $\mathscr{R}\models\varphi(r_1,\ldots,r_d)$ means that $\varphi(r_1,\ldots,r_d)$ is valid in $\mathscr{R}$. We write $\mathscr{R}\models\varphi(x_1,\ldots,x_d)$ if $\varphi(r_1,\ldots,r_d)$ is valid in $\mathscr{R}$, for any tuple $(r_1,\ldots,r_d)\in\mathbb{R}^d$. Formulas $\varphi(x_1,\ldots,x_d)$ and $\psi(x_1,\ldots,x_d)$ are said to be ($\mathscr{R}$–) equivalent if $\mathscr{R}\models\varphi(x_1,\ldots,x_d)\leftrightarrow\psi(x_1,\ldots,x_d)$.

Quantifier-free formulas can be translated into equivalent disjunctions of conjunctions (the latter are also called systems) of rational polynomial inequations, each of the form $p(x_1,\ldots,x_d)>0$ or $p(x_1,\ldots,x_d)\leqslant 0$, where $p(x_1,\ldots,x_d)$ denotes a polynomial in the variables $x_1,\ldots,x_d$ with rational coefficients. Notice that we do not distinguish between

base constants, operations and relations of structure $\mathscr{R}$ and the symbols denoting them; $>$ denotes the negation of $\leqslant$. Rational numbers stand for variable-free $\mathscr{R}$-terms having them as values. FO-formulas and terms are considered to be words over a suitable finite alphabet. To this purpose, the indices of variables are binarily encoded. The translation just mentioned can effectively be performed: it is executable by a recursive word function.

A formula $\varphi = \varphi(x_1,\ldots,x_d)$ *represents* the following set of d-tuples,

$$\mathrm{set}(\varphi) = \{(r_1,\ldots,r_d) \in \mathbb{R}^d : \mathscr{R} \models \varphi(r_1,\ldots,r_d)\}.$$

A set $S \subseteq \mathbb{R}^d$ is called *FO-representable* if there is an FO-formula representing it.

As a tool which will be used throughout the paper, we recall Tarski's fundamental result that $\mathscr{R}$ (like every real closed field) admits effective quantifier elimination, cf. [33, 6].

Tool 1 (EQE: Effective Quantifier Elimination). *There is a recursive function Φ such that, for every FO-formula $\varphi = \varphi(x_1,\ldots,x_d)$, $\Phi(\varphi)$ is an $\mathscr{R}$-equivalent quantifier-free FO-formula in the same variables $x_1,\ldots,x_d$.*

In particular, FO-representable sets can always be represented by quantifier-free formulas. The corresponding result follows for quasi-formulas which may contain arbitrary real numbers, so-called parameters. The sets representable by means of quasi-formulas are just the semialgebraic sets considered in real algebraic geometry. We here are mainly interested in the parameter-free case, however.

By a *finite algorithmic procedure* (briefly: FAP), we mean a sequence $\mathscr{A} = (B_0; B_1; \cdots; B_l)$ of *instructions* B_λ of the following types.

Assignments:	$x_j := C$	$(C \in \{0,1\})$ and
	$x_j := x_{j_1} \omega x_{j_2}$	$(\omega \in \{+,-,\cdot,/\})$;
branchings:	*if* $x_{j_1} \leqslant x_{j_2}$ *then goto* λ_1	$(\lambda_1 \in \{0,1,\ldots,l\})$;
stop instructions:	*halt*.	

The indices of variables, j, j_1, j_2, have to be constant positive natural numbers. Thus, a FAP acts only on finitely many variables explicitly given in it. Moreover, we suppose that the last instruction is the only stop instruction: $B_l =$ "*halt*", and $B_\lambda \neq$ "*halt*", for $0 \leqslant \lambda < l$. Computations always start with the first instruction B_0.

The meaning of the instructions and the step-by-step working of an FAP are straightforward, cf. also the definition of the computation tree below. Let an FAP $\mathscr{A}$ act on the variables $x_1,\ldots,x_d,x_{d+1}$ and, possibly, some further auxiliary variables. Then, with respect to dimension d, it *computes* the function $F_{\mathscr{A},d}$ defined as follows.

For all $(r_1,\ldots,r_d) \in \mathbb{R}^d$, let $F_{\mathscr{A},d}(r_1,\ldots,r_d)$ be defined if and only if procedure $\mathscr{A}$, starting with the values r_i in the variables x_i, for $1 \leqslant i \leqslant d$, and with value 0 in all its other variables, stops after finitely many steps of working. In this case, let $F_{\mathscr{A},d}(r_1,\ldots,r_d)$ be equal to the last current value of variable x_{d+1}.

A real function $f:\mathbb{R}^d \rightarrowtail \mathbb{R}$ is said to be *algebraically computable* if there is a FAP $\mathscr{A}$ such that $f = F_{\mathscr{A},d}$.

This concept is already strong enough to include all real functions of fixed arities which are intuitively computable in $\mathscr{R}$ from the algebraic point of view. Even if we would allow the use of more comfortable storages like stacks of real numbers or if we would consider the parameter-free variant of Friedman's effective definitional schemes [9] or parameter-free BSS machines [2], we would obtain the same set of computable functions from some $\mathbb{R}^d$ into $\mathbb{R}$. This was pointed out by Friedman and Mansfield [10]. Of course, in order to compute string functions over $\mathscr{R}$, i.e., to process arbitrarily long input sequences of reals, one would need some implementation of indirect addressing like in the BSS machines, cf. [2, 11]. This aspect, however, is not the subject of the present paper.

The naturalness of our concept is also stressed by the fact that the restrictions of algebraically computable real functions to input tuples of rational numbers yield just those functions $g:\mathbb{Q}^d \rightarrowtail \mathbb{Q}$ which are classically computable via some standard encoding of the rational numbers, p.e., by pairs of integers. This fact easily follows from the definition of FAPs, and also from the proposition below in this section.

To analyse the actions of an FAP $\mathscr{A} = (B_0; B_1; \cdots; B_l)$ in more detail, one considers its *computation tree* $\mathscr{T}_{\mathscr{A}}$. This usually is a connected directed binary tree whose paths correspond to the possible computations of $\mathscr{A}$. As it has been done in [11] for a more general setting, here the vertices v of $\mathscr{T}_{\mathscr{A}}$ are identified with the non-empty finite sequences of indices of the instructions that are performed up to reaching the corresponding situation: $v \in \{0,1,\ldots,l\}^+$. So the computation tree can inductively be defined as follows.

The word $v_0 = 0$ is the root of $\mathscr{T}_{\mathscr{A}}$.

A vertex $v\lambda, v \in \{0,1,\ldots,l-1\}^*, \lambda \in \{0,1,\ldots,l\}$,

- has no son iff $B_\lambda =$ "*halt*";
- it has a son $v\lambda\lambda'$ iff
  - B_λ is an assignment and $\lambda' = \lambda + 1$, or
  - B_λ is a branching instruction "*if* $x_{j_1} \leqslant x_{j_2}$ *then goto* λ_1" and $\lambda' = \lambda + 1$ or $\lambda' = \lambda_1$.

Obviously, the set of vertices of $\mathscr{T}_{\mathscr{A}}$ is a recursive set of words in the alphabet $\{0,1,\ldots,l\}$.

To every $v = \lambda_1 \cdots \lambda_m \in \{0,1,\ldots,l\}^+$ and any dimension d, we assign the set of all input tuples which lead to vertex v in the computation tree of $\mathscr{A}$. More precisely,

- if v is a vertex of $\mathscr{T}_{\mathscr{A}}$, then

$$W_d(v) = \{(r_1,\ldots,r_d) \in \mathbb{R}^d : \text{starting with the input tuple } (r_1,\ldots,r_d), \text{ procedure } \mathscr{A} \text{ performs the sequence of instructions } B_{\lambda_1},\ldots,B_{\lambda_m}, \text{ in the first } m \text{ steps}\};$$

- if v does not occur in $\mathscr{T}_{\mathscr{A}}$, then $W_d(v) = \emptyset$.

For example, if v_1 and v_2 are vertices of $\mathscr{T}_{\mathscr{A}}$, $W_d(v_1) \subseteq W_d(v_2)$ iff the word v_2 is an initial segment of v_1.

It immediately follows from the inductive definition that the computation tree of a FAP can effectively be generated, successively up to any finite level. This corresponds to the symbolic simulation of the actions of the procedure. So one successively obtains formulas $\varphi_{d,v} = \varphi_{d,v}(x_1,\ldots,x_d)$ representing the sets $W_d(v)$. Moreover, the current values of the variables x_j used by the FAP, at the situation corresponding to vertex v, can be represented by a term $t_{d,v,j} = t_{d,v,j}(x_1,\ldots,x_d)$ only depending on the input variables $x_1,\ldots,x_d$. More precisely, let

$\varphi_{d,0}(x_1,\ldots,x_d) =$ "$x_1 \leqslant x_1$" (an always true formula), and

$$t_{d,0,j}(x_1,\ldots,x_d) = \begin{cases} x_j & \text{if } 1 \leqslant j \leqslant d, \\ 0 & \text{otherwise.} \end{cases}$$

Given $\varphi_{d,v\lambda}$ and $t_{d,v\lambda,j}$, we define

- if $B_\lambda =$ "$x_j := C$", for some $C \in \{0,1\}$, and $\lambda' = \lambda + 1$, then

$$\varphi_{d,v\lambda\lambda'}(x_1,\ldots,x_d) = \varphi_{d,v\lambda}(x_1,\ldots,x_d), \text{ and}$$

$$t_{d,v\lambda\lambda',j}(x_1,\ldots,x_d) = \begin{cases} C & \text{if } j = j_0, \\ t_{d,v\lambda,j}(x_1,\ldots,x_d) & \text{if } j \neq j_0; \end{cases}$$

- if $B_\lambda =$ "$x_{j_0} := x_{j_1} \omega x_{j_2}$", for some operation ω, and $\lambda' = \lambda + 1$, then

$$\varphi_{d,v\lambda\lambda'}(x_1,\ldots,x_d) = \varphi_{d,v\lambda}(x_1,\ldots,x_d), \text{ and}$$

$$t_{d,v\lambda\lambda',j}(x_1,\ldots,x_d) = \begin{cases} t_{d,v\lambda,j_1}(x_1,\ldots,x_d)\omega t_{d,v\lambda,j_2}(x_1,\ldots,x_d) & \text{if } j = j_0, \\ t_{d,v\lambda,j}(x_1,\ldots,x_d) & \text{if } j \neq j_0; \end{cases}$$

- if $B_\lambda =$ "*if* $x_{j_1} \leqslant x_{j_2}$ *then goto* λ_1", then

$$\varphi_{d,v\lambda\lambda'}(x_1,\ldots,x_d) = \begin{cases} \varphi_{d,v\lambda}(x_1,\ldots,x_d) \wedge (t_{d,v\lambda,j_1}(x_1,\ldots,x_d) \leqslant t_{d,v\lambda,j_2}(x_1,\ldots,x_d)) & \text{if } \lambda' = \lambda_1, \\ \varphi_{d,v\lambda}(x_1,\ldots,x_d) \wedge (t_{d,v\lambda,j_1}(x_1,\ldots,x_d) > t_{d,v\lambda,j_2}(x_1,\ldots,x_d)) & \text{if } \lambda' = \lambda + 1; \end{cases}$$

$$t_{d,v\lambda\lambda',j}(x_1,\ldots,x_d) = t_{d,v\lambda,j}(x_1,\ldots,x_d).$$

Thus, we have a second tool well-known from literature, cf. [31, 11, 36]:

Tool 2 (CTA: Computation tree analysis). *For any dimension d and any FAP $\mathscr{A} = (B_0; B_1; \cdots; B_l)$ which uses the variables $x_1,\ldots,x_d,x_{d+1},\ldots,x_m$, there are recursive functions Φ_d and Ψ_d such that, for all vertices v of $\mathscr{T}_\mathscr{A}$,*

- *$\Phi_d(v) = \varphi_{d,v}(x_1,\ldots,x_d)$ is a system of rational polynomial inequations which represents the set $W_d(v)$;*
- *$\Psi_d(v,j) = t_{d,v,j}(x_1,\ldots,x_d)$ is an $\mathscr{R}$-term representing the current value of variable x_j at the situations corresponding to vertex v $(1 \leqslant j \leqslant m)$.*

We remark that both the functions Φ_d and Ψ_d are uniformly effective with respect to dimension d and procedure $\mathscr{A}$, the latter considered as a word over a suitable alphabet.

By CTA, one obtains characterizations of the halting set and the function computed by a FAP. Vertex v of $\mathcal{T}_{\mathcal{A}}$ is said to be a *halting vertex* of the procedure if it terminates with the index l; remember that B_l is just the only stop instruction. Let $\mathrm{HV}(\mathcal{A})$ denote the set of all halting vertices of $\mathcal{A}$. The d-dimensional *halting set* of the procedure is defined as

$$\mathrm{Halt}_d(\mathcal{A}) = \{(r_1,\ldots,r_d) \in \mathbb{R}^d : \text{starting with the input tuple } (r_1,\ldots,r_d), \text{ procedure } \mathcal{A} \text{ reaches the stop instruction after finitely many work steps}\}.$$

Thus,

$$\mathrm{Halt}_d(\mathcal{A}) = \bigcup_{v \in \mathrm{HV}(\mathcal{A})} W_d(v).$$

Proposition 2.1. *A set $S \subseteq \mathbb{R}^d$ is the d-dimensional halting set of an FAP iff there is a recursive function Φ of $\mathbb{N}$ into the set of FO-formulas (or the systems of rational polynomial inequations) in the variables $x_1,\ldots,x_d$ such that* $\mathrm{set}(\Phi(n_1)) \cap \mathrm{set}(\Phi(n_2)) = \emptyset$ *if* $n_1 \neq n_2$, *and*

$$S = \bigcup_{n \in \mathbb{N}} \mathrm{set}(\Phi(n)).$$

A function $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$ is algebraically computable iff there are a recursive function Φ like described above, for $S = \mathrm{dom}(f)$, and, moreover, a recursive function Ψ of $\mathbb{N}$ into the set of $\mathcal{R}$-terms in the variables $x_1,\ldots,x_d$ such that

$$\text{if } (r_1,\ldots,r_d) \in \mathrm{set}(\Phi(n)), \text{ then } f(r_1,\ldots,r_d) = \Psi(n)(r_1,\ldots,r_d),$$

for all $(r_1,\ldots,r_d) \in \mathbb{R}^d, n \in \mathbb{N}$.

Proof. This representation of algebraically computable functions corresponds to Friedman's [9] notion of effective definitional schemes, applied to $\mathcal{R}$ and without parameters. So the proof is due to Friedman, cf. [9, 10]. We only give a sketch (for the second part of the proposition).

The direction "$\rightarrow$" can be shown by a straightforward application of CTA. Remark that $\mathrm{HV}(\mathcal{A})$ is a recursive set and that $W_d(v_1) \cap W_d(v_2) = \emptyset$, for any two different $v_1, v_2 \in \mathrm{HV}(\mathcal{A})$. Moreover, $F_{\mathcal{A},d}(r_1,\ldots,r_d) = t_{d,v,d+1}(r_1,\ldots,r_d)$ if $(r_1,\ldots,r_d) \in W_d(v)$.

For direction "$\leftarrow$", one has to observe that every classically computable arithmetical function can also be computed by a FAP. Thus, given some input tuple $(r_1,\ldots,r_d) \in \mathbb{R}^d$, (gödelizations of) the formulas $\Phi(n)$ can be put here, for $n = 0, 1, 2, \ldots$, up to realizing that $(r_1,\ldots,r_d) \in \mathrm{set}(\Phi(n))$. This is effectively decidable, and when such an n has been found, the value of the term $\Psi(n)(r_1,\ldots,r_d)$ is the corresponding value of function f. □

The content of the proposition is disappointing in some sense. Roughly speaking, algebraic computability in the real numbers is completely characterized by classical computability only enriched by the use of FO-formulas and $\mathcal{R}$-terms.

In particular, constant and total algebraically computable real functions yield only rational values. On the other hand, the proposition shows that algebraically computable functions are not necessarily continuous and that their domains are not necessarily recursively open, cf. the definition in the next section. Thus, their class is incomparable with that of approximately computable functions in the sense of Ko–Friedman [21].

For a better illustration, now we are going to discuss some consequences of the proposition with respect to the 1-dimensional case. For $d=1$, the formulas $\Phi(n)$ are systems (i.e., conjunctions) of rational polynomial inequations in the only variable x_1 which is now simply written as x. Hence the sets set($\Phi(n)$) are unions of finitely many open, closed or semiclosed real intervals whose endpoints are either from $\{-\infty,+\infty\}$, or they are algebraic numbers, namely zeroes of polynomials occurring as terms within $\Phi(n)$. The terms $\Psi(n)$ give rational functions (i.e., quotients of polynomials) depending on variable x.

Algebraic numbers can straightforwardly be encoded by pairs consisting of rational polynomials and positive integers. Let

$$\alpha(p(x),k)=r \quad \text{iff} \quad p(r)=0 \text{ and there are exactly } k-1 \text{ real numbers } y \text{ such that } y<r \text{ and } p(y)=0;$$

this means that r is the kth zero of polynomial $p(x)$, $k\in\mathbb{N}_+$. α is a partial function from $A^*\times\mathbb{N}_+$ onto the set of all algebraic numbers, for some suitable alphabet A. The equality and order of algebraic numbers are recursively decidable with respect to the encoding α, i.e., the set

$$\{((p_1(x),k_1),(p_2(x),k_2)):\alpha(p_1(x),k_1)\leqslant\alpha(p_2(x),k_2)\}$$

is classically recursive. This holds since

$$\alpha(p_1(x),k_1)\leqslant\alpha(p_2(x),k_2) \quad \text{iff} \quad \text{the following FO-formula is true:}$$

$$\forall z_1\forall z_2[p_1(z_1)=0\wedge p_2(z_2)=0 \wedge\exists^{=(k_1-1)}y(y<z_1\wedge p_1(y)=0)\wedge\exists^{=(k_2-1)}y(y<z_2\wedge p_2(y)=0)\rightarrow z_1\leqslant z_2].$$

This formula does not contain a free variable. By EQE, it can effectively be translated into an equivalent quantifier-free formula without variables. Thus, its validity is recursively decidable. By the way, we would obtain related results by encoding every algebraic number r by a rational polynomial $p(x)$ and a rational interval in which r is the only zero of $p(x)$.

Again by EQE, from the formulas $\Phi(n)$, one can effectively obtain a representation of set($\Phi(n)$) by finitely many mutually disjoint intervals whose endpoints are $-\infty$ or $+\infty$ or algebraic numbers given by codes according to the encoding α. Moreover, it is recursively decidable which of the endpoints belong to set($\Phi(n)$). Thus, we have

Corollary 2.1. *A real function $f:\mathbb{R}\rightarrowtail\mathbb{R}$ is algebraically computable iff there are*
- *a recursive sequence of mutually disjoint real intervals $(I_n)_{n\in\mathbb{N}}$, each with algebraic endpoints given by their α-codes, and*

– *a recursive sequence of* $\mathscr{R}$*-terms* $(t_n)_{n\in\mathbb{N}}$ *in the variable* x
such that
– $\mathrm{dom}(f)=\bigcup_{n\in\mathbb{N}} I_n$ *and*
– $f(r)=t_n(r)$ *if* $r\in I_n$ $(n\in\mathbb{N})$.

Notice that the intervals I_n are herein allowed to be empty. In particular, the empty function $f=\emptyset$ is algebraically computable.

3. Approximate computability

In the sequel, d-tuples of real numbers or variables are also denoted by the corresponding bold-face letters, p.e., $\boldsymbol{r}=(r_1,\ldots,r_d)$ or $\boldsymbol{x}=(x_1,\ldots,x_d)$. Topological notations always refer to the natural topology in the Euclidean space $\mathbb{R}^d$, which is induced by the standard norm, $|\boldsymbol{r}|=(\sum_{i=1}^{d} r_i^2)^{1/2}$, or by the maximum norm, $|\boldsymbol{r}|_{\max}=\max\{|r_i|: 1\leqslant i\leqslant d\}$.

We shall often have to deal with open and closed *d-dimensional intervals*, i.e., sets of the form

$$\mathrm{int}^d(\boldsymbol{a},\boldsymbol{b})=\{r\in\mathbb{R}^d: a_i<r_i<b_i \text{ for } 1\leqslant i\leqslant d\} \text{ and}$$

$$\mathrm{int}^d[\boldsymbol{a},\boldsymbol{b}]\{r\in\mathbb{R}^d: a_i\leqslant r_i\leqslant b_i \text{ for } 1\leqslant i\leqslant d\},$$

respectively. Correspondingly, we also prefer to work with the d-dimensional open *cubes*,

$$C_a^d(\boldsymbol{r})=\mathrm{int}^d((r_1-a/2,\ldots,r_d-a/2),(r_1+a/2,\ldots,r_d+a/2))$$
$$=\{\boldsymbol{r}'\in\mathbb{R}^d: |\boldsymbol{r}-\boldsymbol{r}'|_{\max}<a/2\},\quad a\in\mathbb{R},$$

instead of the customarily taken spheres. Like the spheres, the cubes build a basis of the natural topology of $\mathbb{R}^d$, even if we restrict ourselves to the recursively enumerable class of rational cubes. Here, both intervals and cubes are called *rational* if they are defined by rational numbers, $\boldsymbol{a},\boldsymbol{b},\boldsymbol{r}\in\mathbb{Q}^d$, $a\in\mathbb{Q}$, respectively. Then they can straightforwardly be encoded by finitary objects, like tuples of rationals, and we usually do not distinguish between them and their codes. The 1-dimensional intervals are denoted as (a,b) and $[a,b]$.

To define approximate computability, tuples of real numbers are named and handled by means of special sequences of multidimensional intervals. A sequence, $\Omega=(\Omega_n)_{n\in\mathbb{N}}$, of d-dimensional closed, nonempty intervals, $\Omega_n=\mathrm{int}^d[\boldsymbol{a}_n,\boldsymbol{b}_n]$, is called *nested* if $\Omega_{n+1}\subseteq\Omega_n$, for all $n\in\mathbb{N}$. It is called *regular* if, moreover, the intervals are rational, $\boldsymbol{a}_n,\boldsymbol{b}_n\in\mathbb{Q}^d$, and $|\boldsymbol{a}_n-\boldsymbol{b}_n|_{\max}<2^{-n}$, for all $n\in\mathbb{N}$. Then there is just one $\boldsymbol{r}\in\mathbb{R}^d$ such that $\{\boldsymbol{r}\}=\bigcap_{n\in\mathbb{N}}\Omega_n$, and we shall say that the regular sequence Ω *converges to* $\boldsymbol{r}$, or we shall call $\boldsymbol{r}$ the *target* of Ω, briefly: $\boldsymbol{r}=\mathrm{tar}(\Omega)$.

The definition of approximate computability we shall use here is based upon the concept of (function-) *oracle Turing machine* (briefly: OTM) in the sense of Ko–Friedman [21]. Such an OTM $\mathscr{M}$ takes a natural number n as input and a regular

sequence Ω as oracle, and it yields a 1-dimensional rational interval $[a', b'] \subseteq \mathbb{R}$ as output if it halts. The oracle queries are of the form "m?", for $m \in \mathbb{N}$, and they are answered by providing the machine with the mth oracle interval, $\Omega_m = \mathrm{int}^d[\boldsymbol{a}_m, \boldsymbol{b}_m]$. For further details, cf. [21, 20].

If the machine $\mathscr{M}$ with oracle Ω on input n halts with some output $[a', b']$, we write:

$$\mathscr{M}^{\Omega}(n) = [a', b'].$$

A function $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$ is said to be *approximately KF-computable* (this means: computable in the Ko–Friedman sense) if there is an OTM $\mathscr{M}$ such that

1. for every $\boldsymbol{r} \in \mathrm{dom}(f)$ and any regular sequence Ω that converges to $\boldsymbol{r}$, the results $\mathscr{M}^{\Omega}(n)$ always exist and yield a regular sequence of one-dimensional intervals, $(\mathscr{M}^{\Omega}(n))_{n\in\mathbb{N}}$, which converges to the real number $f(\boldsymbol{r})$;
2. for all $\boldsymbol{r} \notin \mathrm{dom}(f)$, every regular sequence Ω with $\mathrm{tar}(\Omega) = \boldsymbol{r}$ and all $n \in \mathbb{N}$, $\mathscr{M}^{\Omega}(n)$ is undefined (i.e., the machine does not reach a halt).

It is easily seen that this definition is only a slight, equivalent modification of the original one by Ko–Friedman who used binarily converging sequences of dyadic numbers to represent real numbers. Thus, it is known that this concept of computability, coincides with a variety of related notions defined by several authors, see [21, 20, 38, 39]. More precisely, these equivalences hold at least for total functions or with respect to the restrictions of computable functions to closed intervals. Computability of partial functions with more complicated domains seems to be dealt with at the first time by Ko and Friedman [21] in the just described way, cf. also [32]. It turned out that the domains of approximately KF-computable functions are exactly the recursively open sets.

A set $S \subseteq \mathbb{R}^d$ is called *recursively open* if there is a recursive function Φ of $\mathbb{N}$ into the class of d-dimensional rational open cubes such that

$$S = \bigcup_{n\in\mathbb{N}} \Phi(n).$$

S is said to be *recursively closed* if the complement, $\mathbb{R}^d \setminus S$, is recursively open.

Instead of the cubes, one could also allow arbitrary open FO-representable sets to exhaust the recursively open sets.

Proposition 3.1. *A set $S \subseteq \mathbb{R}^d$ is recursively open iff there is a recursive function Ψ of $\mathbb{N}$ into the set of FO-formulas such that* $\mathrm{set}(\Psi(n))$ *is open, for all $n \in \mathbb{N}$, and*

$$S = \bigcup_{n\in\mathbb{N}} \mathrm{set}(\Psi(n)).$$

Proof. The first direction of the proof is trivial, since from any rational cube a representing FO-formula is easily obtained.

Conversely, if $S = \bigcup_{n\in\mathbb{N}} \mathrm{set}(\Psi(n))$ as specified above, then

$$\begin{aligned} S &= \bigcup\{C_a^d(\boldsymbol{q}) : a \in \mathbb{Q},\ a \geqslant 0,\ \boldsymbol{q} \in \mathbb{Q}^d,\ C_a^d(\boldsymbol{q}) \subseteq S\} \\ &= \bigcup\{C_a^d(\boldsymbol{q}) : a \in \mathbb{Q},\ a \geqslant 0,\ \boldsymbol{q} \in \mathbb{Q}^d,\ C_a^d(\boldsymbol{q}) \subseteq \mathrm{set}(\Psi(n)),\ \text{for some } n \in \mathbb{N}\}. \end{aligned}$$

The latter set of rational cubes is recursively enumerable, since

$$C_a^d(\boldsymbol{q}) \subseteq \mathrm{set}(\Psi(n)) \quad \text{iff} \quad \forall \boldsymbol{x}[(|\boldsymbol{x}-\boldsymbol{q}|_{\max} < a/2) \to \Psi(n)(\boldsymbol{x})],$$

and, by EQE, one effectively obtains an equivalent quantifier-free FO-formula whose validity is recursively decidable, with respect to the arguments $a, \boldsymbol{q}$. □

Lemma 3.1. *If a function $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$ is approximately KF-computable, then* $\mathrm{dom}(f)$ *is recursively open. If a set $S \subseteq \mathbb{R}^d$ is recursively open, then the function $f = S \times \{0\}$ is approximately KF-computable.*

Proof. The proof is analogous to that one given by Ko–Friedman, see [21, 20]. □

Unfortunately, there are rather simple functions which do not have recursively open domains. For example, the trivial sets $\emptyset$ and $\mathbb{R}^d$ are the only subsets of $\mathbb{R}^d$ which are both open and closed. Thus, if S is a nontrivial closed subset of $\mathbb{R}^d$, the constant function on S, $f_S = S \times \{0\}$, cannot be approximately KF-computable. Moreover, the empty set is the only subset of $\mathbb{N}^d$ which is open in $\mathbb{R}^d$. Therefore, the empty function $\emptyset$ is the only function $f : \mathbb{N}^d \rightarrowtail \mathbb{N}$ that is approximately KF-computable.

Now we define a generalization of approximate KF-computability which was already introduced by Kreitz and Weihrauch [22]. It includes the classical computability over $\mathbb{N}$ and also allows, for example, the computability of functions with closed domains.

A function $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$ is called *approximately computable* if there is an OTM $\mathcal{M}$ such that

1. for every $\boldsymbol{r} \in \mathrm{dom}(f)$ and any regular sequence Ω with $\mathrm{tar}(\Omega) = \boldsymbol{r}$, the results $\mathcal{M}^{\Omega}(n)$ always exist and $\mathrm{tar}((\mathcal{M}^{\Omega}(n))_{n\in\mathbb{N}}) = f(\boldsymbol{r})$
   (this coincides with the first condition of approximate KF-computability);
2. for all $\boldsymbol{r} \notin \mathrm{dom}(f)$ and every regular sequence Ω with $\mathrm{tar}(\Omega) = \boldsymbol{r}$, there is an input $n \in \mathbb{N}$ such that $\mathcal{M}^{\Omega}(n)$ remains undefined (i.e., the machine does not halt).

In our opinion, this definition corresponds well to the idea of approximate computability. It includes also functions over domains which are not recursively open. On recursively open domains, however, our notion of computability coincides with the KF version, as we are going to show now.

Obviously, every approximately KF-computable function is also approximately computable. On the other hand, we have

Proposition 3.2. *Let $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$ be an approximately computable function and S be a recursively open set such that $S \subseteq \mathrm{dom}(f)$. Then the restriction of function f to the set S, i.e., the function $f_{|S}$, is approximately KF-computable.*

Proof. Suppose that there is a representation $S = \bigcup_{n\in\mathbb{N}} \Phi(n)$, with some recursive function Φ yielding rational open cubes $\Phi(n)$, and that there is an OTM $\mathcal{M}$ that approximately computes the function f. For every input $n \in \mathbb{N}$ and any regular sequence Ω, let the OTM $\mathcal{M}'$ first, via Cantor's recursive pairing function, search for

a pair (m,k) such that $\Omega_m \subseteq \Phi(k)$. This condition is expressible in the form

$$\forall x(x \in \Omega_m \to x \in \Phi(k)).$$

By EQE and using the oracle Ω and the computation of function Φ, an equivalent quantifier-free FO-formula $\varphi_{(m,k)}$ can effectively be obtained from (m,k). Moreover, since $\varphi_{(m,k)}$ does not depend on free variables, its validity is recursively decidable. Thus, such a pair (m,k) can be found iff it exists, and this holds iff $\mathrm{tar}(\Omega) \in S$. Then let $\mathscr{M}'$ halt with the result of machine $\mathscr{M}$, i.e., with $\mathscr{M}^{\Omega}(n)$ if it exists. Otherwise, if $\mathscr{M}^{\Omega}(n)$ does not exist or $\mathrm{tar}(\Omega) \notin S$, $\mathscr{M}'$ does never halt.

This OTM $\mathscr{M}'$ computes approximately the restriction $f_{|S}$ in the KF sense. □

Immediately we now have

Corollary 3.1. *Let $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$ and $\mathrm{dom}(f)$ be recursively open. Then f is approximately computable iff it is approximately KF-computable.*

The corollary implies that our concept of approximate computability owns all the properties well-known from the other equivalent approaches, at least if it is applied to total functions or to the restrictions of computable functions with recursively open domains to closed intervals. We here formulate only two examples which even apply to our general notion of computability: it is closed under compositions of functions, and approximately computable functions are continuous.

Lemma 3.2. *Let $f_1 : \mathbb{R}^d \rightarrowtail \mathbb{R}, f_2 : \mathbb{R} \rightarrowtail \mathbb{R}$ be approximately computable functions. Then $f_2 \circ f_1$ is approximately computable, too.*

Proof. The proof is by standard arguments. □

The analogous result would hold for functions $f_1 : \mathbb{R}^{d_1} \rightarrowtail \mathbb{R}^{d_2}, f_2 : \mathbb{R}^{d_2} \rightarrowtail \mathbb{R}^{d_3}, d_1, d_2, d_3 \in \mathbb{N}_+$, if the approximate computability of such vector-valued functions would be defined in the straightforward way.

The following proposition shows that our concept satisfies the fundamental thesis of approximate computability.

Proposition 3.3. *Every approximately computable function $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$ is continuous on its domain.*

Proof. For the proof, we have to show that from $r, r_n \in \mathrm{dom}(f)$, $\lim_{n\to\infty} r_n = r$, it follows $\lim_{n\to\infty} f(r_n) = f(r)$. Let M be an OTM that the function f computes approximately.

We consider a regular sequence Ω with $\mathrm{tar}(\Omega) = r$ such that r belongs to the interiors of all intervals Ω_m, $m \in \mathbb{N}$. For any $k \in \mathbb{N}$, there is an index m_k such that, in computing $\mathscr{M}^{\Omega}(k)$, machine $\mathscr{M}$ puts only oracle queries concerning indices $m \leqslant m_k$. Thus, for almost all elements r_n, there are regular sequences $\Omega^{(n)}$ with $\mathrm{tar}(\Omega^{(n)}) = r_n$

and $\Omega_m^{(n)} = \Omega_m$ for all $m \leqslant m_k$. It follows that almost all $f(\boldsymbol{r}_n)$, as well as $f(\boldsymbol{r})$, belong to the interval $\mathscr{M}^{\Omega}(k)$, i.e., $|f(\boldsymbol{r}_n) - f(\boldsymbol{r})| < 2^{-k}$. Therefore, $\lim_{n\to\infty} f(\boldsymbol{r}_n) = f(\boldsymbol{r})$. □

Remark that we did not suppose or conclude in the proof that $\mathrm{dom}(f)$ has to be closed.

Now we are going to show that our concept also includes the classical computability to a certain extend.

Proposition 3.4. *Let* $f : \mathbb{N}^d \rightarrowtail \mathbb{N}$, *and* $\mathrm{dom}(f)$ *be a recursively enumerable set. Then* f *is a (classically) recursive function iff it is approximately computable.*

Proof. Given a recursive function f, let the OTM $\mathscr{M}$ work as follows, on an input $n \in \mathbb{N}$ and a regular oracle Ω. First it checks if the interval Ω_n contains a tuple of natural numbers, $\boldsymbol{m} \in \mathbb{N}^d$. If yes, this tuple $\boldsymbol{m}$ is uniquely determined. Then $\mathscr{M}$ tries to compute $f(\boldsymbol{m})$ according to a classical Turing machine. When a result $k = f(\boldsymbol{m})$ is obtained, let $\mathscr{M}$ output the interval $[k - 2^{-(n+2)}, k + 2^{-(n+2)}]$. If $\boldsymbol{m} \notin \mathrm{dom}(f)$, $\mathscr{M}$ does not halt. Also if $\Omega_n \cap \mathbb{N}^d = \emptyset$, let $\mathscr{M}^{\Omega}(n)$ remain undefined. Then $\mathscr{M}$ computes f approximately.

Conversely, suppose that $\mathscr{M}$ approximately computes the function f and that $\mathrm{dom}(f)$ is recursively enumerable. A classical Turing machine computing f (with respect to some standard encoding of tuples of natural numbers) can work as follows, on a given input $\boldsymbol{m} \in \mathbb{N}^d$. First it checks if $\boldsymbol{m} \in \mathrm{dom}(f)$. If not, it does never halt. If yes, $f(\boldsymbol{m})$ can be obtained by simulating the computation of $\mathscr{M}$ on input $n = 0$, with the oracle sequence Ω defined by $\Omega_k = \mathrm{int}^d[\boldsymbol{m} - (2^{-(k+2)}, \ldots, 2^{-(k+2)}), \boldsymbol{m} + (2^{-(k+2)}, \ldots, 2^{-(k+2)})]$, $k \in \mathbb{N}$. Indeed, $\mathscr{M}^{\Omega}(0)$ is a closed real interval $[a', b']$ of a length $< 2^0 = 1$, and it contains the number $f(\boldsymbol{m})$. □

Notice that the supposition of recursive enumerability of $\mathrm{dom}(f)$ is essential for the second part of the proof. If the Turing machine would simply simulate the computation of $\mathscr{M}^{\Omega}(0)$, one would classically compute a function $\overline{f}$ such that $f \subseteq \overline{f}$. The domains of approximately computable functions, both generally and restricted to the natural numbers, will be characterized in Section 6.

A real number r is said to be (approximately) *computable* if there is a classically computable regular sequence Ω such that $r = \mathrm{tar}(\Omega)$. This is equivalent to the usual definitions, and by standard arguments one obtains

Lemma 3.3. *For any real number* r *are equivalent:*

(i) r *is (approximately) computable,*
(ii) *the constant total function* $f = \mathbb{R} \times \{r\}$ *is approximately computable,*
(iii) *the set* $\mathbb{R} \setminus \{r\}$ *is recursively open,*
(iv) *the partial function* $f = ((-\infty, r) \times \{-1\}) \cup ((r, +\infty) \times \{1\})$ *is approximately computable.*

In contrast to KF-computability, with respect to our concept, the function $f_r = \{r\} \times \{0\}$ is also approximately computable, for every computable number r. Indeed, let $\Omega^{(r)}$

be a classically computable regular sequence such that $\mathrm{tar}(\Omega^{(r)}) = r$. Without loss of generality, we suppose that r belongs to the interiors of all intervals $\Omega_n^{(r)}$, $n \in \mathbb{N}$. Let the OTM $\mathcal{M}$ work as follows, on input n and any regular oracle Ω.

If there is a $k \in \mathbb{N}$ with $\Omega_k \subseteq \Omega_n^{(r)}$, $\mathcal{M}$ outputs the (degenerated) interval $[0,0]$; otherwise $\mathcal{M}^{\Omega}(n)$ remains undefined. Thus, $\mathcal{M}^{\Omega}(n)$ is defined for all $n \in \mathbb{N}$ iff $\mathrm{tar}(\Omega) \subseteq \Omega_n^{(r)}$, for all $n \in \mathbb{N}$, and this holds iff $\mathrm{tar}(\Omega) = r$.

We want to stress again that the concepts of algebraic resp. approximate computability are mutually incomparable. On the one hand, a constant total function with an irrational but (approximately) computable real value is approximately but not algebraically computable. On the other hand, there are discontinuous total real functions which are algebraically computable. Now we are going to show that not even for continuous total functions algebraic computability implies the approximate computability.

Our example is based on a standard (Gödel) numbering of the FAPs $(\mathcal{A}_n : n \in \mathbb{N})$. Analogously, the classical Turing machines could be used. Without loss of generality, we can suppose that no FAP begins with the stop instruction. So at least one work step has always to be performed.

There is a universal FAP which simulates step-by-step the work of any $\mathcal{A}_n$ on a given input assignment. If the inputs are rational, this simulation can even recursively be performed. In particular, and only this is essential in the sequel, there is a recursive total function $\kappa : \mathbb{N}^2 \to \{0,1\}$ such that

$$\kappa(n,l) = \begin{cases} 1 & \text{if } \mathcal{A}_n \text{ reaches the stop instruction after exactly } l \text{ steps,} \\ & \text{on the empty input assignment (i.e., } x_i = 0, \text{ for all } i \in \mathbb{N}_+), \\ 0 & \text{otherwise.} \end{cases}$$

The *special halting problem*,

$$\mathrm{HP} = \{n : \text{there is an } l \in \mathbb{N} \text{ such that } \kappa(n,l) = 1\}$$

is not recursively decidable, since $(\mathcal{A}_n : n \in \mathbb{N})$ immediately yields a standard numbering of the partial recursive functions over $\mathbb{N}$.

We consider the function $\hat{f} : \mathbb{R} \to \mathbb{R}$ defined as follows.

Let $\hat{f}(r) = 0$, for all $r \in (-\infty, 0)$ and all $r \in \mathbb{N}$. On the intervals $(m, m+1)$, $\hat{f}$ is defined in dependence on the number of steps performed by $\mathcal{A}_m$ on the empty input assignment ($x_i = 0$, for all $i \in \mathbb{N}_+$). More precisely, if $m \in \mathbb{N}$ and $\mathcal{A}_m$ reaches the stop instruction after exactly l steps, then let

$$\hat{f}(r) = \begin{cases} 0 & \text{if } r \in (m + 1/l, m+1), \\ 1 & \text{if } r = m + 1/(2l), \\ s \cdot 2l & \text{if } r = m + s, \text{ for } s \in (0, 1/(2l)), \\ 1 - s \cdot 2l & \text{if } r = m + 1/(2l) + s, \text{ for } s \in (0, 1/(2l)); \end{cases} \qquad (*)$$

if $\mathcal{A}_m$ does never stop on the empty input, then let

$$\hat{f}(r) = 0 \quad \text{for all } r \in (m, m+1).$$

Obviously, $\hat{f}$ is a continuous total real function.

Lemma 3.4. *The just defined real function $\hat{f}$ is algebraically computable, but it is not approximately computable.*

Proof. Given an input $r \in (0,\infty)\backslash\mathbb{N}$, $\hat{f}(r)$ can be obtained by determining first the numbers $m \in \mathbb{N}$, $l \in \mathbb{N}_+$ such that $r \in (m + 1/(l+1), m + 1/l]$, by computing then the values $\kappa(m,l')$, for all $l' \in \{1,2,\ldots,l\}$, and halting finally with the output according to definition $(*)$ above. This can be performed by a FAP.

Assume now that an OTM M would approximately compute the function $\hat{f}$. Then it could be used to solve the halting problem HP in the following way: For $n \in \mathbb{N}$, simulate the computation of $\mathscr{M}^{\Omega^{(n)}}(2)$, where $\Omega^{(n)} = ([n - 1/2^{k+2}, n + 1/2^{k+2}])_{k\in\mathbb{N}}$. This computation finally halts with some output $[a',b']$, $b' < \frac{1}{4}$, and it uses only oracle intervals with indices $k \leqslant k_0$, for some $k_0 \in \mathbb{N}$. It follows that $\hat{f}(r) < \frac{1}{4}$, for all $r \in (n, n + 1/2^{k_0+2})$. Thus, by computing $\kappa(n,l)$ for the finitely many $l \in \{0,1,2,\ldots,2^{k_0+2}\}$, it could effectively be decided whether $n \in \mathrm{HP}$. □

4. Approximate computability by means of FAPs

Whereas algebraic computability refers to finite, halting computations of FAPs, approximate computability can be characterized by means of the infinite, never halting computations. They will also be referred to as *passing computations.*

Given a FAP $\mathscr{A} = (B_0; B_1; \cdots; B_l)$, its *passing set of depth t* (and dimension d) is defined to be the set of all input tuples on which $\mathscr{A}$ has not yet stopped after t steps of work, i.e.,

$$P_{d,t}(\mathscr{A}) = \bigcup\{W_d(v)\colon v \text{ is a vertex of } \mathscr{T}_{\mathscr{A}},\ v \notin \mathrm{HV}(\mathscr{A}),\ \mathrm{length}(v) = t+1\}.$$

This is complementary to the *halting set of depth t*,

$$H_{d,t}(\mathscr{A}) = \bigcup\{W_d(v)\colon v \in \mathrm{HV}(\mathscr{A}),\ \mathrm{length}(v) \leqslant t+1,\}.$$

It holds $P_{d,t}(\mathscr{A}) \cap H_{d,t}(\mathscr{A}) = \emptyset$, $P_{d,t}(\mathscr{A}) \cup H_{d,t}(\mathscr{A}) = \mathbb{R}^d$, $P_{d,t+1}(\mathscr{A}) \subseteq P_{d,t}(\mathscr{A})$, and $H_{d,t+1}(\mathscr{A}) \supseteq H_{d,t}(\mathscr{A})$.

Analogously to the first part of Proposition 2.1, by CTA we obtain an effective representation of the halting resp. passing set of any depth.

Proposition 4.1. *For every FAP $\mathscr{A}$ acting on variables $x_1,\ldots,x_d,\ldots$, there is a recursive function Φ of $\mathbb{N}$ into the set of (quantifier-free) FO-formulas in the variables $x_1,\ldots,x_d$ such that, for all $t \in \mathbb{N}$, $H_{d,t}(\mathscr{A}) = set(\Phi(t))$, and $P_{d,t}(\mathscr{A}) = set(\neg\Phi(t))$.*

By Section 2, the halting set (of dimension d) of procedure $\mathscr{A}$ is

$$\mathrm{Halt}_d(\mathscr{A}) = \bigcup_{t\in\mathbb{N}} H_{d,t}(\mathscr{A}).$$

Complementary, for every dimension d, we now define the *passing set* of $\mathscr{A}$,

$$\mathrm{Pass}_d(\mathscr{A}) = \bigcap_{t\in\mathbb{N}} P_{d,t}(\mathscr{A}).$$

It follows

$$\mathrm{Pass}_d(\mathscr{A}) \cap \mathrm{Halt}_d(\mathscr{A}) = \emptyset \quad \text{and} \quad \mathrm{Pass}_d(\mathscr{A}) \cup \mathrm{Halt}_d(\mathscr{A}) = \mathbb{R}^d.$$

Let $\mathscr{A}$ be a FAP acting on the variables $x_1, \ldots, x_d, x_{d+1}$ and, possibly, some further ones. With respect to dimension d, it *determines approximately* a function $G_{\mathscr{A},d} : \mathbb{R}^d \rightarrowtail \mathbb{R}$ defined as follows.

For $\boldsymbol{r} \in \mathbb{R}^d$, $G_{\mathscr{A},d}(\boldsymbol{r})$ is defined if and only if

(1) for all $m \in \mathbb{N}$, there are numbers $l, t \in \mathbb{N}$ such that,
for all $(\boldsymbol{r}_1, y_1), (\boldsymbol{r}_2, y_2) \in P_{d+1,t}(\mathscr{A})$, it holds
$\boldsymbol{r}_1, \boldsymbol{r}_2 \in C^d_{2^{-l}}(\boldsymbol{r}) \rightarrow |y_1 - y_2| < 2^{-m}$; and

(2) for all $l, t \in \mathbb{N}$,
$(C^d_{2^{-l}}(\boldsymbol{r}) \times \mathbb{R}) \cap P_{d+1,t}(\mathscr{A}) \neq \emptyset$.

If Conditions (1) and (2) hold for $\boldsymbol{r} \in \mathbb{R}^d$, there is exactly one $y_{\boldsymbol{r}} \in \mathbb{R}$ which satisfies

(3) $(\boldsymbol{r}, y_{\boldsymbol{r}}) \in \bigcap_{t \in \mathbb{N}} \mathrm{cl}(P_{d+1,t}(\mathscr{A}))$.

Then let $G_{\mathscr{A},d}(\boldsymbol{r}) = y_{\boldsymbol{r}}$.

Herein "cl" denotes the *closure operator* applicable to subsets of $\mathbb{R}^{d+1}$. To improve the readability, the pairs $(\boldsymbol{r}, y) \in \mathbb{R}^d \times \mathbb{R}$ are identified with the corresponding $(d+1)$-tuples $(r_1, \ldots, r_d, y)$.

To show the existence of a $y_{\boldsymbol{r}}$ satisfying Condition (3), let for $n \in \mathbb{N}$, $l_n, t_n \in \mathbb{N}$ be chosen such that $|y_1 - y_2| < 2^{-n}$ if $(\boldsymbol{r}_1, y_1), (\boldsymbol{r}_2, y_2) \in P_{d+1,t_n}(\mathscr{A})$ and $\boldsymbol{r}_1, \boldsymbol{r}_2 \in C^d_{2^{-l_n}}(\boldsymbol{r})$. Then we take $(\boldsymbol{r}_n, y_n) \in (C^d_{2^{-l_n}}(\boldsymbol{r}) \times \mathbb{R}) \cap P_{d+1,t_n}(\mathscr{A})$. This is possible by Conditions (1) and (2), respectively. Moreover, one can secure that $\lim_{n\to\infty} l_n = \infty$. It follows $\lim_{n\to\infty} \boldsymbol{r}_n = \boldsymbol{r}$. Since $(y_n)_{n\in\mathbb{N}}$ is a Cauchy sequence, there exists $y_{\boldsymbol{r}} = \lim_{n\to\infty} y_n$. We have

$$(\boldsymbol{r}, y_{\boldsymbol{r}}) \in \bigcap_{t \in \mathbb{N}} \mathrm{cl}(P_{d+1,t}(\mathscr{A})).$$

The uniqueness of $y_{\boldsymbol{r}}$ follows immediately from (1). Thus, the definition of function $G_{\mathscr{A},d}$ has been shown to be correct.

Obviously, Conditions (1) and (2) just ensure that $G_{\mathscr{A},d}(\boldsymbol{r}) = y_{\boldsymbol{r}}$ is uniquely defined by (3). In other words, $G_{\mathscr{A},d}$ is simply defined by the set $\bigcap_{t\in\mathbb{N}} \mathrm{cl}(P_{d+1,t}(\mathscr{A}))$ which is considered as a relation from $\mathbb{R}^d$ into $\mathbb{R}$ but restricted to those arguments, for which this relation is unique.

To say it more precisely, let

$$\mathrm{graph}(f) = \{(r_1, \ldots, r_d, y) \in \mathbb{R}^{d+1} : f(r_1, \ldots, r_d) = y\},$$

for any function $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$. Then it is easily shown

Proposition 4.2. *For every FAP $\mathscr{A}$ acting on variables $x_1, \ldots, x_d, x_{d+1}, \ldots$, it holds*

- $\mathrm{dom}(G_{\mathscr{A},d}) = \{\boldsymbol{r} \in \mathbb{R}^d : \exists^{=1} y[(\boldsymbol{r}, y) \in \bigcap_{t\in\mathbb{N}} \mathrm{cl}(P_{d+1,t}(\mathscr{A}))]\}$,
- $\mathrm{graph}(G_{\mathscr{A},d}) = \bigcap_{t\in\mathbb{N}} \mathrm{cl}(P_{d+1,t}(\mathscr{A})) \cap (\mathrm{dom}(G_{\mathscr{A},d}) \times \mathbb{R})$.

Conversely, by the equations given in the proposition, the function $G_{\mathscr{A},d}$ is uniquely determined. So it has been shown to be defined in a rather natural way. For some further discussion, we refer to the next section.

The remaining part of this section is devoted to the main theorem of the paper which also stresses the naturalness of the concept of approximate determination of functions by FAPs.

Theorem 4.1. *A function $f:\mathbb{R}^d \rightarrowtail \mathbb{R}$ is approximately computable iff there is a FAP $\mathscr{A}$ such that $f = G_{\mathscr{A},d}$.*

Proof. To start with the proof of direction "←", let $f = G_{\mathscr{A},d}$, for a FAP $\mathscr{A}$. According to Proposition 4.1, a representation of the $(d+1)$-dimensional halting set of $\mathscr{A}$ is supposed,

$$H_{d+1,t}(\mathscr{A}) = \mathrm{set}(\Phi(t)),$$

where $\Phi(t) = \Phi(t)(x_1,\ldots,x_d,x_{d+1})$ are quantifier-free FO-formulas.

Now we describe the work of an OTM $\mathscr{M}$, on an input $n \in \mathbb{N}$ and a regular oracle $\Omega = (\Omega_k)_{k\in\mathbb{N}}$ consisting of d-dimensional intervals Ω_k.

1. First let $\mathscr{M}$ check if $(\Omega_l \times \mathbb{R}) \cap P_{d+1,t}(\mathscr{A}) \neq \emptyset$, for all $l,t \leqslant n$. Since, for any pair (l,t), Ω_l is a constant rational interval and $P_{d+1,t}(\mathscr{A}) = \mathrm{set}(\neg\Phi(t))$, this test condition is easily FO-representable and then effectively decidable by means of EQE.
   If the answer is "no", for some $l,t \leqslant n$, let the computation of $\mathscr{M}^{\Omega}(n)$ never halt.
2. Otherwise, let $\mathscr{M}$, according to the order defined via Cantor's recursive pairing function, search for a pair $(l,t) \in \mathbb{N}^2$ such that for all $(r_1,y_1),(r_2,y_2) \in P_{d+1,t}(\mathscr{A})$

   $$r_1, r_2 \in \Omega_l \rightarrow |y_1 - y_2| < 2^{-(2n+4)}.$$

   Again, this condition is FO-expressible and effectively decidable by means of function Φ and via EQE.
3. When such a pair (l,t) has been found, $\mathscr{M}$ searches for a rational number q satisfying

   $$(\Omega_l \times (q - 2^{-(2n+4)}, q + 2^{-(2n+4)})) \cap P_{d+1,t}(\mathscr{A}) \neq \emptyset.$$

   This last test condition is effectively decidable, too, and a corresponding rational number q can be found if it exists, by means of a recursive enumeration of the set $\mathbb{Q}$.
   If this search terminates successfully, let

   $$\mathscr{M}^{\Omega}(n) = [q - 2^{-(2n+2)}, q + 2^{-(2n+2)}].$$

   Otherwise, $\mathscr{M}^{\Omega}(n)$ remains undefined.

Assume that $G_{\mathscr{A},d}(r)$ is defined and equal to some y_r. Let $\mathrm{tar}(\Omega) = r$. Then, for $n \in \mathbb{N}$, the check 1 always terminates with "yes", by Condition (2) of the definition of $G_{\mathscr{A},d}$. Moreover, by Condition (1) and since Ω is a regular sequence, the search 2 is successful. Finally, the search 3 yields some $q_n \in \mathbb{Q}$ such that $|y_r - q_n| \leqslant 2^{-(2n+3)}$.

We have $\mathcal{M}^{\Omega}(n)=[q_n-2^{-(2n+2)},q_n+2^{-(2n+2)}]$. Since $q_n-2^{-(2n+2)}\leqslant y_r-2^{-(2n+3)}\leqslant q_{n+1}-2^{-(2n+4)}\leqslant y_r\leqslant q_{n+1}+2^{-(2n+4)}\leqslant y_r+2^{-(2n+3)}\leqslant q_n+2^{-(2n+2)}$, it follows that $(\mathcal{M}^{\Omega}(n))_{n\in\mathbb{N}}$ is a regular sequence of intervals with the target y_r.

If $G_{\mathcal{A},d}(\boldsymbol{r})$ is not defined, either the check 1 or the search 3, or the search 2 cannot be successful, for some $n\in\mathbb{N}$, and $\mathcal{M}^{\Omega}(n)$ remains undefined.

So we have shown the approximate computability of function $G_{\mathcal{A},d}$. Notice that the approximate KF-computability cannot be obtained in general.

Now we are going to show direction "$\rightarrow$" of the theorem. Let f be approximately computed by an OTM $\mathcal{M}$.

Without loss of generality, we can suppose that, for all regular sequences Ω and all $n\in\mathbb{N}$, if $\mathcal{M}^{\Omega}(n)$ is defined,

(i) $\mathcal{M}^{\Omega}(n)=[a',b']$, for some $a',b'\in\mathbb{Q}$ with $a'<b'$,

(ii) $\mathcal{M}^{\Omega}(n_1)$ is defined too, for all $n_1<n$, and $\mathcal{M}^{\Omega}(n)\subseteq\mathcal{M}^{\Omega}(n_1)$.

Indeed, if $\mathcal{M}$ does not yet satisfy these conditions, they can be achieved by slight modifications of $\mathcal{M}$ without changing the function which is approximately computed. Firstly, it can be ensured that $\mathcal{M}^{\Omega}(n)$ is defined only if $\mathcal{M}^{\Omega}(n_1)$ is also defined and $\mathcal{M}^{\Omega}(n)\subseteq\mathcal{M}^{\Omega}(n_1)$, for all $n_1\leqslant n$, $n\in\mathbb{N}$. Secondly, the degenerated intervals, when they occur at index $n'+1$ at the first time, i.e., $\mathcal{M}^{\Omega}(n')=[a',b']$, $a'<b'$, and $\mathcal{M}^{\Omega}(n'+1)=[c',c']$, can be avoided by taking $[2^{-m}(a'+c'),2^{-m}(b'+c')]$ instead of $\mathcal{M}^{\Omega}(n'+m)$, for $m\geqslant 1$. The case that already $\mathcal{M}^{\Omega}(0)=[c',c']$ can similarly be treated.

A regular sequence $\Omega=(\Omega_k)_{k\in\mathbb{N}}$ of d-dimensional intervals is called *distinguished* if, for all $k\in\mathbb{N}$, there is a tuple of integers, $\boldsymbol{z}\in\mathbb{Z}^d$, such that

$$\Omega_k=\mathrm{int}^d[2^{-(k+1)}\boldsymbol{z},2^{-(k+1)}(\boldsymbol{z}+\boldsymbol{1})],$$

where $\boldsymbol{1}=(1,\ldots,1)\in\mathbb{Z}^d$. This means that the interval Ω_k is a (closed) cell of the d-dimensional regular grid of mesh width $2^{-(k+1)}$. The interiors of these cells are mutually disjoint, neighbouring cells have common boundary elements however.

More precisely, a real d-tuple $\boldsymbol{r}$ belongs to exactly 2^c of the cells of width $2^{-(k+1)}$ if c is the number of components r_i of $\boldsymbol{r}$ which are representable as $r_i=z2^{-(k+1)}$, with integers $z\in\mathbb{Z}$. Thus, for any $\boldsymbol{r}\in\mathbb{R}^d$ there are just 2^c distinguished sequences with the target $\boldsymbol{r}$ if c is the number of components representable as $r_i=z_i2^{-(k_i+1)}$, with $k_i\in\mathbb{N}$, $z_i\in\mathbb{Z}$ (i.e., r_i is an integral multiple of reciprocal power of 2).

Now we are ready to describe the work of a FAP $\mathcal{A}$ starting with some $(d+1)$-tuple $(\boldsymbol{r},y)=(r_1,\ldots,r_d,y)$ whose components are the input values of the variables $x_1,\ldots,x_d,x_{d+1}$, respectively.

Procedure $\mathcal{A}$ works by Stages n, for $n=0,1,2,\ldots$.

In Stage n, it tries to simulate the computations of $\mathcal{M}^{\Omega}(n)$, for all distinguished sequences Ω with $\mathrm{tar}(\Omega)=\boldsymbol{r}$. Recall that there are at most 2^d of such sequences and, if $\mathcal{M}^{\Omega}(n)$ exists, this result depends only on a finite initial part of the sequence Ω. Moreover, the finite initial parts of arbitrary length of these sequences can effectively be put here by the FAP $\mathcal{A}$, depending on the input components of $\boldsymbol{r}$. Herein, the tuple $\boldsymbol{r}$ is only used for simple order tests implementing the conditions

"$\boldsymbol{r} \in \mathrm{int}^d[2^{-(k+1)}\boldsymbol{z}, 2^{-(k+1)}(\boldsymbol{z}+\boldsymbol{1})]$", for some $k \in \mathbb{N}$, $\boldsymbol{z} \in \mathbb{Z}^d$. Thus, the simulations can be performed as long as $\mathscr{M}^{\Omega}(n)$ exists, for all such sequences Ω.
If $\Omega^{(1)}, \ldots, \Omega^{(l)}$, $1 \leqslant l \leqslant 2^d$, are the distinguished sequences with target $\boldsymbol{r}$ and all the $\mathscr{M}^{\Omega^{(i)}}(n)$ exist, $\mathscr{A}$ finally obtains the set

$$R^{(\boldsymbol{r})}(n) = \bigcup_{1 \leqslant i \leqslant l} \mathscr{M}^{\Omega^{(i)}}(n).$$

The sets $\mathscr{M}^{\Omega^{(i)}}(n)$ are closed rational intervals, each with a length $< 2^{-n}$ and containing the value $f(\boldsymbol{r})$ if it exists.
Let procedure $\mathscr{A}$ halt at Stage n on input $(\boldsymbol{r}, y)$ if $y \notin R^{(\boldsymbol{r})}(n)$; otherwise, let it continue with Stage $n+1$.
If some $\mathscr{M}^{\Omega^{(i)}}(n)$ does not exist, Stage n does not terminate on the input $(\boldsymbol{r}, y)$.

Remark that the number l of sequences is not known a priori. If $\boldsymbol{r}$ has a component of form $r_i = z2^{-(k+1)}$, where k is minimal, the corresponding "splitting" of the distinguished sequences at level k is realized at Stage n only if, in the course of computing $\mathscr{M}^{\Omega}(n)$, an oracle query "m?" is put, for some $m \geqslant k$. Within the computation tree $\mathscr{T}_{\mathscr{A}}$, the path, which is determined by the input $(\boldsymbol{r}, y)$ up to Stage n has been performed, depends only on the *set* of the initial parts $(\Omega_k^{(i)})_{0 \leqslant k \leqslant K_n^{(i)}}$, $1 \leqslant i \leqslant l$, where $K_n^{(i)}$ is the maximal index of an element of $\Omega^{(i)}$ to which $\mathscr{M}$ queries in the course of computing $\mathscr{M}^{\Omega^{(i)}}(n)$.

If some result $\mathscr{M}^{\Omega^{(i)}}(n)$ does not exist, the corresponding simulation by $\mathscr{A}$ in Stage n does never halt, i.e., $\mathscr{A}$ remains within this stage ad infinitum and does not reject a further input. If this happens (for the first time) at Stage n, then

$$\{\boldsymbol{r}\} \times R^{(\boldsymbol{r})}(n-1) \quad \subseteq \mathrm{Pass}_{d+1}(\mathscr{A}) \quad \text{if } n > 0,$$
$$\{\boldsymbol{r}\} \times \mathbb{R} \quad \subseteq \mathrm{Pass}_{d+1}(\mathscr{A}) \quad \text{if } n = 0.$$

Remember that $R^{(\boldsymbol{r})}(0) \supseteq R^{(\boldsymbol{r})}(1) \supseteq \cdots \supseteq R^{(\boldsymbol{r})}(n)$ by Supposition (ii), as long as these sets are defined. Moreover, their diameters, $\mathrm{diam}(R^{(\boldsymbol{r})}(n)) = \sup\{|y_1 - y_2| : y_1, y_2 \in R^{(\boldsymbol{r})}(n)\}$, are properly greater than 0, because of Suppositon (i).

Thus, if $f(\boldsymbol{r})$ is not defined, then $\mathscr{M}^{\Omega^{(i)}}(n)$ does not exist, for some $n \in \mathbb{N}$ and a distinguished sequence $\Omega^{(i)}$ converging to $\boldsymbol{r}$, and it follows that $G_{\mathscr{A},d}(\boldsymbol{r})$ is undefined due to Condition (1).

Otherwise, if $f(\boldsymbol{r})$ exists, then all the results $\mathscr{M}^{\Omega^{(i)}}(n)$ are obtained by $\mathscr{A}$. So it successively runs through all the stages, computes the $R^{(\boldsymbol{r})}(n)$ and halts on input $(\boldsymbol{r}, y)$, for all $y \notin \bigcap_{n \in \mathbb{N}} R^{(\boldsymbol{r})}(n)$. Since $\mathscr{M}$ approximately computes function f, it follows

$$\{f(\boldsymbol{r})\} = \bigcap_{n \in \mathbb{N}} R^{(\boldsymbol{r})}(n) \quad \text{and} \quad (\boldsymbol{r}, f(\boldsymbol{r})) \in \mathrm{Pass}_{d+1}(\mathscr{A}) \subseteq \bigcap_{t \in \mathbb{N}} \mathrm{cl}(P_{d+1,t}(\mathscr{A})).$$

Moreover, since each $R^{(\boldsymbol{r})}(n)$ is a union of intervals of lengths $< 2^{-n}$ that contains $f(\boldsymbol{r})$, it holds $\mathrm{diam}(R^{(\boldsymbol{r})}(n)) < 2^{-(n-1)}$.

It remains to show that Condition (1) of the definition of $G_{\mathscr{A},d}(\boldsymbol{r})$ holds. To this purpose, let be given some $m \in \mathbb{N}$.

Case 1: $\boldsymbol{r}$ always belongs to the interior of its cells in the grids, for all mesh widths $2^{-(k+1)}$. Then there is just one distinguished sequence $\Omega^{(1)}$ with $\mathrm{tar}(\Omega^{(1)})=\boldsymbol{r}$. We consider a level t in the computation tree $\mathscr{T}_{\mathscr{A}}$ at which Stage $m+1$ has been performed, for the input $(\boldsymbol{r}, f(\boldsymbol{r}))$. Let K_m be the maximal index k such that the oracle query "k ?" was put in the course of performing Stages $0,\ldots,m,m+1$. Now we choose a natural number l in such a way that

$$C^d_{2^{-l}}(\boldsymbol{r}) \subseteq \Omega^{(1)}_{K_m}.$$

If $(\boldsymbol{r}_1, y_1), (\boldsymbol{r}_2, y_2) \in P_{d+1,t}(\mathscr{A})$ and $\boldsymbol{r}_1, \boldsymbol{r}_2 \in C^d_{2^{-l}}$, then $\mathscr{A}$ works, up to Stage $m+1$, on the inputs $(\boldsymbol{r}_1, y_1)$ and $(\boldsymbol{r}_2, y_2)$ like on $(\boldsymbol{r}, f(\boldsymbol{r}))$. Thus, $y_1, y_2 \in R^{(\boldsymbol{r})}(m+1)$, and $|y_1 - y_2| < 2^{-m}$.

Case 2: $\boldsymbol{r}$ has components of form $r_i = z2^{-(k+1)}$, for some $z \in \mathbb{Z}$, $k \in \mathbb{N}$. Let $\Omega^{(1)},\ldots,\Omega^{(l)}$, $1 \leqslant l \leqslant 2^d$, be the distinguished sequences with target $\boldsymbol{r}$. Now we consider a level t in $\mathscr{T}_{\mathscr{A}}$, at which Stage $m+1$ has been performed both for the input $(\boldsymbol{r}, f(\boldsymbol{r}))$ and also for all the other inputs whose distinguished sequences coincide with some of the $\Omega^{(i)}$ up to the maximal query index used up to Stage $m+1$. Let K_m denote the maximal index of an oracle element used in the course of performing Stages $0,\ldots,m,m+1$. We choose an $l \in \mathbb{N}$ such that

$$C^d_{2^{-l}}(\boldsymbol{r}) \subseteq \bigcup_{1 \leqslant i \leqslant l} \Omega^{(i)}_{K_m}.$$

If $(\boldsymbol{r}_1, y_1), (\boldsymbol{r}_2, y_2) \in P_{d+1,t}(\mathscr{A})$ and $\boldsymbol{r}_1, \boldsymbol{r}_2 \in C^d_{2^{-l}}$, it follows again that $|y_1 - y_2| < 2^{-m}$.

This completes the proof of the theorem. □

5. Robustness and some discussion

It is possible to avoid the closure operator in Condition (3) of the definition of $G_{\mathscr{A},d}$ or in Proposition 4.2, simply by the requirement that the sets $P_{d+1,t}(\mathscr{A})$ have to be closed. This leads to an interesting special type of FAPs.

A FAP $\mathscr{A}$ acting on variables $x_1,\ldots,x_d,\ldots$, is called *robust* with respect to dimension d (briefly: d-robust) if $P_{d,t}(\mathscr{A})$ is closed, for every $t \in \mathbb{N}$. From this it follows that $\mathrm{Pass}_d(\mathscr{A})$ is closed, but not conversely (see Lemma 5.4 below).

FAP $\mathscr{A}$ is d-robust iff its halting sets $H_{d,t}(\mathscr{A})$ are open, on all depths t. So we see that robustness represents a certain kind of stability of the halting sets. Indeed, it means that any set $H_{d,t}(\mathscr{A})$ contains, with some tuple $\boldsymbol{r} \in \mathbb{R}^d$, always a certain neighbourhood $C^d_a(\boldsymbol{r}), a > 0$. By Proposition 5.1 below, the halting sets of robust procedures are just the recursively open sets which were originally defined as unions of effective sequences of rational open cubes. It follows that $\boldsymbol{r} \in \mathrm{Halt}_d(\mathscr{A})$ is recognizable by the validity of some proper inequalities of form $a_i < r_i < b_i$, with rational numbers a_i, b_i produced by a classical procedure working over $\mathbb{Q}$. In particular, any equality tests for irrational numbers can be avoided in the halting computations of robust FAPs.

The example at the end of Section 3 can be used to show that not every (continuous, total) algebraically computable function $f:\mathbb{R}^d \to \mathbb{R}$ can be computed by a d-robust FAP. If the function $\hat{f}:\mathbb{R}\to\mathbb{R}$ defined there would be algebraically computable by a 1-robust FAP $\mathscr{A}$, the special halting problem HP would be recursively decidable.

Indeed, by the Heine–Borel covering theorem , to every $n\in\mathbb{N}$, a level $t_n\in\mathbb{N}$ would exist such that $[n,n+1]\subseteq H_{1,t_n}(\mathscr{A})$. By means of EQE, this t_n is recursively computable from n. Now, $n\in\mathrm{HP}$ iff there is an $r\in[n,n+1]$ such that $\hat{f}(r)=1$. This could be decided by means of the term representations of the values of variable x_2, which are given by $\Psi_1(v,2)$, for $\mathrm{length}(v)\leqslant t_n+1$, according to CTA (Tool 2).

Proposition 5.1. *A set $S\subseteq\mathbb{R}^d$ is the halting set of a d-robust FAP iff it is recursively open. S is the passing set of a d-robust FAP iff it is recursively closed.*

Proof. The second assertion is only another formulation of the first one. For direction "$\to$" of the proof of the first assertion, we use a representation of the halting set of some FAP $\mathscr{A}$ according to Proposition 4.1,

$$H_{d,t}(\mathscr{A})=\mathrm{set}(\Phi(t)) \quad \text{for all } t\in\mathbb{N},$$

with a recursive function Φ of $\mathbb{N}$ into the set of FO-formulas. Then

$$\mathrm{Halt}_d(\mathscr{A})=\bigcup_{t\in\mathbb{N}}\mathrm{set}(\Phi(t)),$$

and if $\mathscr{A}$ is d-robust, by means of Proposition 3.1, we have that $\mathrm{Halt}_d(\mathscr{A})$ is recursively open.

Conversely, let $S=\bigcup_{n\in\mathbb{N}}\Phi(n)$, where Φ is a recursive function of $\mathbb{N}$ into the set of d-dimensional rational open cubes. Then there is a FAP $\mathscr{A}$ which, given any input tuple $\boldsymbol{r}\in\mathbb{R}^d$, successively for $n=0,1,2,\ldots$, generates the open cubes $\Phi(n)$ and halts when $\boldsymbol{r}\in\Phi(n)$. It follows that every $H_{d,t}(\mathscr{A})$ is some finite union of open cubes. □

We remark that the notion of robust FAP is closely related to the concept of locally time bounded BSS machine used by Boldi and Vigna [3]. In particular, Theorem 3 from [3] corresponds to our Proposition 5.1. A set $S\subseteq\mathbb{R}^d$ is recursively open iff it is open and semidecidable by a locally time bounded BSS machine with rational constants in the sense of Boldi and Vigna. Notice that the constants of BSS machines can usually be introduced into the parameter-free FAPs in place of additional variables.

By Proposition 4.2, we immediately have

Lemma 5.1. *For every $(d+1)$-robust* FAP $\mathscr{A}$,
- $\mathrm{dom}(G_{\mathscr{A},d})=\{\boldsymbol{r}\in\mathbb{R}^d\colon \exists^{=1}y[(\boldsymbol{r},y)\in\mathrm{Pass}_{d+1}(\mathscr{A})]\}$,
- $\mathrm{graph}(G_{\mathscr{A},d})=\mathrm{Pass}_{d+1}(\mathscr{A})\cap(\mathrm{dom}(G_{\mathscr{A},d})\times\mathbb{R})$.

Now we show that approximately computable functions can always be determined by (the passing sets of) robust procedures.

Theorem 5.1. *A function $f:\mathbb{R}^d \rightarrowtail \mathbb{R}$ is approximately computable iff $f = G_{\mathscr{A},d}$, for a $(d+1)$-robust FAP $\mathscr{A}$.*

Proof. To show this, for any FAP $\mathscr{A}$ acting on variables $x_1,\ldots,x_d,x_{d+1},\ldots$, we define its $(d+1)$-*robustification* $\mathscr{A}^0$ which works as follows, on inputs $(\boldsymbol{r},y)\in\mathbb{R}^{d+1}$.

For $t=0,1,2,\ldots$, let $\mathscr{A}^0$ check whether $(\boldsymbol{r},y)\in \mathrm{in}(H_{d+1,t}(\mathscr{A}))$
and halt in this case (otherwise, it continues with Stage $t+1$).

By "in", the *interior operator* applicable to subsets of $\mathbb{R}^{d+1}$ is denoted.

Remark that the required checks are effectively executable by a FAP, by means of EQE. Indeed, let Φ be a recursive function according to Proposition 4.1, i.e., $H_{d+1,t}(\mathscr{A}) = \mathrm{set}(\Phi(t))$, for all $t\in\mathbb{N}$. Then it holds

$$(\boldsymbol{r},y)\in \mathrm{in}(H_{d+1,t}(\mathscr{A})) \text{ iff}$$
$$\exists z[z>0 \wedge \forall x_1'\cdots\forall x_d'\forall y'((x_1',\ldots,x_d',y')\in C_z^{d+1}(\boldsymbol{r},y) \rightarrow \Phi(t)(x_1',\ldots,x_d',y'))].$$

From the definition of $\mathscr{A}^0$, it follows that there is a (recursive) strictly monotone sequence of natural numbers, $(n_t)_{t\in\mathbb{N}}$, such that

$$\emptyset = H_{d+1,0}(\mathscr{A}^0) = \cdots = H_{d+1,n_0-1}(\mathscr{A}^0) \quad \text{and}$$
$$\mathrm{in}\,(H_{d+1,t}(\mathscr{A})) = H_{d+1,n_t}(\mathscr{A}^0) = \cdots = H_{d+1,n_{t+1}-1}(\mathscr{A}^0) \quad \text{for all } t\in\mathbb{N}.$$

Thus, $\mathscr{A}^0$ is a $(d+1)$-robust FAP, and

$$\mathrm{Halt}_{d+1}(\mathscr{A}^0) = \bigcup_{t\in\mathbb{N}} \mathrm{in}(H_{d+1,t}(\mathscr{A})),$$
$$\mathrm{cl}(P_{d+1,t}(\mathscr{A})) = P_{d+1,n_t}(\mathscr{A}^0)) \quad \text{for all } t\in\mathbb{N},$$
$$\mathrm{Pass}_{d+1}(\mathscr{A}^0) = \bigcap_{t\in\mathbb{N}} \mathrm{cl}(P_{d+1,t}(\mathscr{A})).$$

By Proposition 4.2 and Lemma 5.1, it follows $G_{\mathscr{A}^0,d} = G_{\mathscr{A},d}$. □

As mentioned already in the previous section, Proposition 4.2 like Lemma 5.1 characterize the function $G_{\mathscr{A},d}$ by the sets $\bigcap_{t\in\mathbb{N}} \mathrm{cl}(P_{d+1,t}(\mathscr{A}))$ and $\mathrm{Pass}_{d+1}(\mathscr{A})$, respectively, considered as relations from $\mathbb{R}^d$ to $\mathbb{R}$ and restricted then to those arguments for which they are unique. It naturally arises the question if the approximate determination of a function f can always be performed by a FAP $\mathscr{A}$ such that

$$\mathrm{graph}(f) = \bigcap_{t\in\mathbb{N}} \mathrm{cl}(P_{d+1,t}(\mathscr{A})).$$

If this equation holds, for the just defined robustification $\mathscr{A}^0$, we obtain

$$\mathrm{graph}(f) = \mathrm{Pass}_{d+1}(\mathscr{A}^0).$$

Thus, we can equivalently ask for the existence of a $(d+1)$-robust FAP $\mathscr{A}^0$ which satisfies this equation. From Proposition 5.1, it follows that this is equivalent to the condition that $\mathrm{graph}(f)$ is recursively closed. Therefore, we have

Corollary 5.1. *For a function* $f:\mathbb{R}^d \rightarrowtail \mathbb{R}$, *there is a FAP* $\mathscr{A}$ *satisfying* $\mathrm{graph}(f) = \bigcap_{t\in\mathbb{N}} \mathrm{cl}(P_{d+1,t}(\mathscr{A}))$ *or, equivalently, a* $(d+1)$*-robust FAP* $\mathscr{A}^0$ *satisfying* $\mathrm{graph}(f) = \mathrm{Pass}_{d+1}(\mathscr{A}^0)$ *iff* $\mathrm{graph}(f)$ *is a recursively closed set.*

For example, the approximately KF-computable unary function $f_0(x) = \sin(1/x)$, defined on $\mathbb{R}\backslash\{0\}$, is not representable in that special way, since the closure of its graph contains the set $\{0\}\times[-1,+1]$.

The following lemma gives a sufficient condition for the recursive closedness of a graph.

Lemma 5.2. *If the function* f *is approximately computable and, moreover,* $\mathrm{dom}(f)$ *is recursively closed, then* $\mathrm{graph}(f)$ *is recursively closed, too.*

Proof. Let $\mathscr{A}$ be a $(d+1)$-robust FAP with $f = G_{\mathscr{A},d}$, and $\mathbb{R}^d\backslash\mathrm{dom}(f) = \bigcup_{t\in\mathbb{N}} \mathrm{set}(\Psi(t))$, according to Proposition 3.1. Then $\mathbb{R}^{d+1}\backslash\mathrm{graph}(f)$ is effectively exhausted, in the sense of Proposition 3.1, by the following sequence of open sets:

$$(H_{d+1,t}(\mathscr{A}) \cup (\mathrm{set}(\Psi(t))\times\mathbb{R}))_{t\in\mathbb{N}}. \qquad \square$$

One easily sees that the function $f_1(x) = 1/x$, $\mathrm{dom}(f_1) = \mathbb{R}\backslash\{0\}$, is representable in the form $\mathrm{graph}(f_1) = \mathrm{Pass}_{d+1}(\mathscr{A}_1)$, for a $(d+1)$-robust FAP $\mathscr{A}_1$. Its domain is open, however. Thus, the conversion of the lemma does not hold in general.

It can be shown for bounded functions, however. As usual, a function $f:\mathbb{R}^d \rightarrowtail \mathbb{R}$ is said to be (globally) *bounded* if $\mathrm{graph}(f) \subseteq \mathbb{R}^d \times [-K,K]$, for some constant K.

Lemma 5.3. *Let* $f:\mathbb{R}^d \rightarrowtail \mathbb{R}$ *be a bounded function. If* $\mathrm{graph}(f)$ *is recursively closed, then* $\mathrm{dom}(f)$ *is recursively closed, too.*

Proof. To prove this, let $\mathbb{R}^{d+1}\backslash\mathrm{graph}(f) = \bigcup_{t\in\mathbb{N}} \mathrm{set}(\Psi(t))$ be an effective exhaustion by open FO-representable sets according to Proposition 3.1. Then

$$\mathbb{R}^d\backslash\mathrm{dom}(f) = \bigcup_{n\in\mathbb{N}} S_n, \text{ where } S_n = \left\{ \boldsymbol{r}\in\mathbb{R}^d \colon \{\boldsymbol{r}\}\times[-K,K] \subseteq \bigcup_{t\leqslant n} \mathrm{set}(\Psi(t)) \right\}$$

and K is a bound of function f. Indeed, if $\boldsymbol{r}\in S_n$, i.e., $\{\boldsymbol{r}\}\times[-K,K] \subseteq \bigcup_{t\leqslant n} \mathrm{set}(\Psi(t)) \subseteq \mathbb{R}^{d+1}\backslash\mathrm{graph}(f)$, then it holds $\boldsymbol{r}\notin\mathrm{dom}(f)$.

Conversely, if $\boldsymbol{r}\notin\mathrm{dom}(f)$, then $\{\boldsymbol{r}\}\times[-K,K] \subseteq \bigcup_{t\in\mathbb{N}} \mathrm{set}(\Psi(t))$. By the Heine–Borel covering theorem, there is some $n\in\mathbb{N}$ such that $\{\boldsymbol{r}\}\times[-K,K] \subseteq \bigcup_{t\leqslant n} \mathrm{set}(\Psi(t))$. $\square$

There is no nontrivial set $S\neq\emptyset$, $\mathbb{R}^d$ which is both (recursively) open and (recursively) closed. Thus, Lemma 5.3 implies that for no approximately KF-computable bounded function f with $\emptyset \subset \mathrm{dom}(f) \subset \mathbb{R}^d$ the equation $\mathrm{graph}(f) = \mathrm{Pass}_{d+1}(\mathscr{A})$ is satisfiable by a robust FAP $\mathscr{A}$.

The example below will show that one would leave the scope of approximate computability if one would only require, for functions f, that $\mathrm{graph}(f) = \mathrm{Pass}_{d+1}(\mathscr{A})$, for a not necessarily $(d+1)$-robust FAP $\mathscr{A}$.

Next we are going to deal again with computable real numbers which also can simply be characterized by means of FAPs.

Proposition 5.2. *For any real number r the following are equivalent:*

(i) *r is (approximately) computable,*

(ii) $\{r\} = \mathrm{Pass}_1(\mathscr{A})$, *for a 1-robust* FAP $\mathscr{A}$,

(iii) $\{r\} = \bigcap_{t\in\mathbb{N}} cl(P_{1,t}(\mathscr{A}))$, *for a* FAP $\mathscr{A}$.

Proof. If r is computable, the set $\mathbb{R}\backslash\{r\}$ is recursively open, by Lemma 3.3. From Proposition 5.1 it follows (ii) and this trivially implies (iii).

If $\{r\} = \bigcap_{t\in\mathbb{N}} \mathrm{cl}(P_{1,t}(\mathscr{A}))$, for a FAP $\mathscr{A}$, one easily defines a FAP $\mathscr{A}'$ that approximately determines the function $f_r = \mathbb{R}\times\{r\}$: $\mathscr{A}'$ can simply be obtained by replacing the variables x_i by x_{i+1}, everywhere in the instructions of $\mathscr{A}$. Then, on any input $(s,r)\in\mathbb{R}^2$, $\mathscr{A}'$ does not change the value s of the first variable. Thus, for any $t\in\mathbb{N}$, it holds $P_{2,t}(\mathscr{A}') = \mathbb{R}\times P_{1,t}(\mathscr{A})$. By Proposition 4.2, $\mathscr{A}'$ approximately determines f_r. □

Now we give an example showing that, for Proposition 5.2, it is essential to take the closures of the halting sets in (iii) and to require the robustness of procedure $\mathscr{A}$ in (ii). More precisely, we define a real number r_0 which is not computable but, nevertheless, allows a representation of form $\{r_0\} = \mathrm{Pass}_1(\mathscr{A})$, for a (non robust) FAP $\mathscr{A}$.

Then, by Lemma 3.3(ii), the function $\widehat{f} = \mathbb{R}\times\{r_0\}$ is not approximately computable. On the other hand, there is a representation of the graph as $\mathrm{graph}(\widehat{f}) = \mathrm{Pass}_2(\widehat{\mathscr{A}})$, for a FAP $\widehat{\mathscr{A}}$. This demonstrates the necessity to take the closures within the definition of function $G_{\mathscr{A},d}$, or to require the robustness of $\mathscr{A}$ within Lemma 5.1.

Let $(\mathscr{A}_n: n\in\mathbb{N})$ be a standard numbering of the FAPs, as used at the end of Section 3, and $\kappa:\mathbb{N}^2\to\{0,1\}$ be the corresponding step-counting function defined there.

The sequences $(\beta_n)_{n\in\mathbb{N}}$ and $(k_n)_{n\in\mathbb{N}}$ are inductively defined as follows.

0. Let $\beta_0 = 0$ and $k_0 = 0$.
1. Assume that k_n and β_i, for $0\leqslant i\leqslant k_n$, have been defined.
   - If there is no $l\in\mathbb{N}$ with $\kappa(n,l) = 1$ (i.e., $n\notin \mathrm{HP}$),
     then $k_{n+1} = k_n + 2$,
     $\beta_{k_n+1} = \beta_{k_n+2} = 0$.
   - If $\kappa(n,l) = 1$, for some $l\in\mathbb{N}$ (which is uniquely determined),
     then $k_{n+1} = k_n + 1 + l$,
     $\beta_{k_n+1} = 0$, and
     $\beta_{k_n+2} = \cdots = \beta_{k_n+l} = \beta_{k_{n+1}} = 1$.

We always have $k_{n+1}\geqslant k_n + 2$, by the supposition that no $\mathscr{A}_n$ begins with the stop instruction. The sequence $(\beta_n)_{n\in\mathbb{N}}$ never becomes stationary. Finally, let the real number

$r_0 \in (0,1)$ be defined by

$$r_0 = \sum_{n=0}^{\infty} \beta_n 2^{-n}.$$

Lemma 5.4. *The number r_0 is not computable. There is a* FAP $\widehat{\mathscr{A}}$ *such that*

$$\{r_0\} = \mathrm{Pass}_1(\widehat{\mathscr{A}}).$$

Proof. The first assertion easily follows. From a classically computable regular sequence $\Omega^{(0)}$ with $\mathrm{tar}(\Omega^{(0)}) = r_0$, one would obtain a recursive decision of the set HP: $n \in \mathrm{HP}$ iff $\beta_{k_n+2} = 1$.

Now we sketch the work of a FAP $\widehat{\mathscr{A}}$ satisfying $\{r_0\} = \mathrm{Pass}_1(\widehat{\mathscr{A}})$.
Starting with an input assignment $x_1 = r \in \mathbb{R}$, and $x_i = 0$, for all $i > 1$, $\widehat{\mathscr{A}}$ performs the following Stages n, for $n = 0, 1, 2, \ldots$, up to reaching the stop instruction, possibly.

In Stage n, $\widehat{\mathscr{A}}$ simulates the first n steps (as long as no stop instruction is reached) of the FAPs $\mathscr{A}_0, \mathscr{A}_1, \ldots, \mathscr{A}_n$, always starting with the empty assignment.
Then it halts iff the initial part of length $n+1$ of the binary expansion of input r is not consistent with the results obtained.
More precisely, $\widehat{\mathscr{A}}$ computes $\kappa(i,l)$, for $0 \leqslant i, l \leqslant n$. Input r passes Stage n (i.e., it does not belong to the corresponding halting set) iff

$$r \in \left(\sum_{i=0}^{n} \widehat{\beta}_i 2^{-i}, \sum_{i=0}^{n} \widehat{\beta}_i 2^{-i} + 2^{-(n+1)} \right),$$

for some finite sequence $(\widehat{\beta}_0, \ldots, \widehat{\beta}_n)$ such that
$\widehat{\beta}_0 = 0$, and
if $k = k_m$, for some $m \in \{0, 1, \ldots, n\}, k < n$,
then $\widehat{\beta}_{k+1} = 0$ and, moreover,
- if $\widehat{\beta}_{k+2} = 0$ (and $k+2 \leqslant n$),
  then $\mathscr{A}_m$ performs at least $n+1$ steps on the empty input assignment;
- if $\widehat{\beta}_{k+2} = \cdots = \widehat{\beta}_{k+l} = 1$ (and $k + l \leqslant n$),
  then $\mathscr{A}_m$ performs at least l steps on the empty input;
- if $\widehat{\beta}_{k+2} = \cdots = \widehat{\beta}_{k+l} = 1$, $\widehat{\beta}_{k+l+1} = 0$ (and $k + l + 1 \leqslant n$),
  then $\mathscr{A}_m$ performs exactly l steps on the empty input up to reaching the stop instruction.

It is easily seen that r_0 passes all the stages of $\widehat{\mathscr{A}}$, i.e., $r_0 \in \mathrm{Pass}_1(\widehat{\mathscr{A}})$.

If $r' \in (0,1)$, $r' = \sum_{i=0}^{n} \beta'_i 2^{-i}, r' \neq r_0$, there is a minimal index n such that $\beta'_n \neq \beta_n$. Then at the latest in Stage n, r' belongs to the halting set of $\widehat{\mathscr{A}}$, i.e., $r_0 \notin \mathrm{Pass}_1(\widehat{\mathscr{A}})$. □

In the basic definition of $G_{\mathscr{A},d}$, cf. also Proposition 4.2, or in Lemma 5.1, the function $G_{\mathscr{A},d}$ is treated as a set suitably determined by the $(d+1)$-dimensional passing sets $P_{d+1,t}(\mathscr{A})$ and $\mathrm{Pass}_{d+1}(\mathscr{A})$, respectively. For the definition of a function, the reader

would probably prefer a procedure which leads from a given argument $\boldsymbol{r} \in \mathbb{R}^d$ to a value $G_{\mathscr{A},d}(\boldsymbol{r})$. The following proposition is devoted to this point of view.

Herein we use the notion of *quasi-FAP*, i.e., parameter-dependent FAP. Any quasi-FAP can be thought to be obtained from a FAP $\mathscr{A}$ by replacing some variables, say $x_1,\ldots,x_d$, by real numbers $r_1,\ldots,r_d$ (called parameters) and replacing all the remaining variables x_{i+d} by x_i then, such that the procedure refers to $x_1,x_2,\ldots$ again. Let us denote such a quasi-FAP by $\mathscr{A}[r_1/x_1,\ldots,r_d/x_d]$. These parameter-dependent FAPs are originally used by Friedman [9]. Moreover, they correspond to finite-dimensional BSS machines [1, 2].

The notion of passing set is straightforwardly transferable to quasi-FAPs. Now the approximate determination of a function by a FAP can be expressed in the following way.

Proposition 5.3. *A function $f:\mathbb{R}^d \rightarrowtail \mathbb{R}$ is approximately computable iff there is a $(d+1)$-robust FAP $\mathscr{A}$ such that, for any $\boldsymbol{r}=(r_1,\ldots,r_d)\in\mathbb{R}^d$, the quasi-FAP $\mathscr{A}[r_1/x_1,\ldots,r_d/x_d]$ determines $f(\boldsymbol{r})$ as follows:*

$$\boldsymbol{r} \in dom(f) \text{ iff } Pass_1(\mathscr{A}[r_1/x_1,\ldots,r_d/x_d]) = \{y_r\}, \text{ for some } y_r \in \mathbb{R},$$
$$\text{i.e., } Pass_1(\mathscr{A}[r_1/x_1,\ldots,r_d/x_d]) \text{ is a singleton,}$$

and then let $f(\boldsymbol{r}) = y_r$.

Proof. This immediately follows from the definition of $\mathscr{A}[r_1/x_1,\ldots,r_d/x_d]$ and Lemma 5.1 by means of Theorem 5.1. □

We close this section with a result on the avoidability of multiplications and divisions in approximate determinations of functions by FAPs. This will be of some interest in connection with the definition of (time) complexity classes of approximately computable functions on the basis of FAPs. The details of this application are beyond the scope of this paper, however.

Proposition 5.4. *Let $d \in \mathbb{N}_+$. To every d-robust* FAP $\mathscr{A}$, *one can effectively construct a d-robust* FAP $\mathscr{A}'$ *such that*
- *the instructions of $\mathscr{A}'$ do not use multiplications or divisions,*
- *it holds $Halt_d(\mathscr{A}) = Halt_d(\mathscr{A}')$.*

Proof. If $\mathscr{A}$ is a d-robust FAP, by Proposition 5.1, the set $\mathrm{Halt}_d(\mathscr{A})$ is recursively open. Thus,

$$\mathrm{Halt}_d(\mathscr{A}) = \bigcup_{n\in\mathbb{N}} \Phi(n),$$

where Φ is a recursive function of $\mathbb{N}$ into the set of d–dimensional rational open cubes.

Given an input $\boldsymbol{r} \in \mathbb{R}^d$, let the FAP $\mathscr{A}'$ successively compute the (codes of the) cubes $\Phi(n)$, for $n=0,1,2,\ldots$, and stop when it realizes that $\boldsymbol{r} \in \Phi(n)$. Then, for any

depth t,

$$H_{d,t}(\mathscr{A}') = \bigcup_{n=0}^{n_t} \Phi(n),$$

for some number n_t. Thus, $\mathscr{A}'$ is d-robust and $\mathrm{Halt}_d(\mathscr{A}) = \mathrm{Halt}_d(\mathscr{A}')$.

Moreover, the computations of $\Phi(n)$ can be performed without using multiplications or divisions. Remember that rational numbers q can be encoded by pairs of integers (k,l) such that $q = k/l$, $l \neq 0$. The checks if $r \in \Phi(n)$ require only order tests of form "$a_{ni} < r_i < b_{ni}$", for rational components a_{ni} and b_{ni} encoded by pairs of integers. Hence, these tests are also executable without using multiplications or divisions: it holds $k/l < r$ iff $k < r + \cdots + r$ (l times r). □

By means of Theorem 5.1, now we have

Corollary 5.2. *Every approximately computable function $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$ can approximately be determined by a $(d+1)$-robust FAP which does not use multiplications or divisions.*

6. Arithmetical hierarchies

We are now going to introduce two modifications of the classical arithmetical hierarchy which seem to be quite suitable to classify subsets of some $\mathbb{R}^d$ from the point of view of algebraic and approximate computability, respectively. Below they will be used to compare the domains, ranges and graphs of algebraically resp. approximately computable functions.

The classes of our first hierarchy, called the *discrete arithmetical hierarchy* (briefly: DAH), are denoted by

$$\Sigma_k^{\mathrm{da}}, \quad \Pi_k^{\mathrm{da}} \quad \text{and} \quad \Delta_k^{\mathrm{da}} \qquad (k \geqslant 1).$$

The second one, called *topological arithmetical hierarchy* (briefly: TAH), consists of the classes

$$\Sigma_k^{\mathrm{ta}}, \quad \Pi_k^{\mathrm{ta}} \quad \text{and} \quad \Delta_k^{\mathrm{ta}} \qquad (k \geqslant 1).$$

By Σ_k^0, Π_k^0, and Δ_k^0, the classes of the well-known classical arithmetical hierarchy (AH) are denoted. They consist of subsets of the Cartesian products $\mathbb{N}^d$, for $d \in \mathbb{N}_+$.

A set S belongs to Σ_k^{da} (we also say that S is a Σ_k^{da}-set), for $k \in \mathbb{N}_+$, if

- $S \subseteq \mathbb{R}^d$, for some $d \in \mathbb{N}_+$, and
- there is a total recursive function Ψ of $\mathbb{N}^k$ into the set of FO-formulas with the free variables $x_1, \ldots, x_d$ such that

$$S = \bigcup_{n_1 \in \mathbb{N}} \bigcap_{n_2 \in \mathbb{N}} \bigcup_{n_3 \in \mathbb{N}} \cdots \bigodot_{n_k \in \mathbb{N}} \mathrm{set}(\Psi(n_1, n_2, \ldots, n_k)),$$

where $\bigodot \in \{\bigcup, \bigcap\}$ such that the prefix of unions and intersections becomes alternating.

If, moreover, for all $(n_1, n_2, \ldots, n_k) \in \mathbb{N}^k$,

- set($\Psi(n_1, n_2, \ldots, n_k)$) is an open set if k is an odd number, and
- set($\Psi(n_1, n_2, \ldots, n_k)$) is a closed set if k is an even number,

then the set S belongs to Σ_k^{ta} (or is said to be a Σ_k^{ta}-set).

Due to EQE, we can suppose that the FO-formulas $\Psi(n_1, n_2, \ldots, n_k)$ within the definition are quantifier free.

As usual, the Π_k-sets are defined to be the complements of the Σ_k-sets, and the Δ_k-sets are the corresponding intersections. More precisely,

$$S' \in \Pi_k^{\flat a} \text{ iff } S' = \mathbb{R}^d \setminus S \quad \text{for some } \Sigma_k^{\flat a}\text{-set } S \subseteq \mathbb{R}^d,$$

$$\Delta_k^{\flat a} = \Sigma_k^{\flat a} \cap \Pi_k^{\flat a} \quad \text{for } \flat a \in \{\mathrm{da}, \mathrm{ta}\}.$$

It immediately follows that the classes of the TAH are included in the corresponding classes of the DAH,

$$\Sigma_k^{\mathrm{ta}} \subseteq \Sigma_k^{\mathrm{da}}, \qquad \Pi_k^{\mathrm{ta}} \subseteq \Pi_k^{\mathrm{da}}, \qquad \Delta_k^{\mathrm{ta}} \subseteq \Delta_k^{\mathrm{da}}.$$

The DAH is similar to Cucker's arithmetical hierarchy over the reals, see [5]. In contrast to the BSS setting, however, we here consider sets of tuples of some dimension d instead of sets of strings (which are finite sequences of arbitrary length). More essentially, we do not allow the use of real parameters within the programs. This implies that our concepts are closely related to classical theory of computability. In particular, the sets in the classes of our hierarchies are generated by complementations and *effective* unions and intersections from the FO-representable sets and FO-representable open sets, respectively.

By means of EQE, cf. Proposition 2.1, one obtains

Lemma 6.1. *$S \in \Sigma_1^{\mathrm{da}}$ iff $S = \mathrm{Halt}_d(\mathscr{A})$, for some FAP $\mathscr{A}$ and a dimension d.*

So Δ_1^{da} consists just of the sets which are *decidable* by FAPs, i.e., whose (total) characteristic functions are algebraically computable. For example, $\mathbb{N} \in \Delta_1^{\mathrm{da}}$ and $\mathbb{Q} \in \Sigma_1^{\mathrm{da}} \setminus \Pi_1^{\mathrm{da}}$.

The TAH connects computational aspects with topological ones. The Σ_k^{ta}-sets can also be represented by means of enumerations of open cubes.

Proposition 6.1. *A set $S \subseteq \mathbb{R}^d$ belongs to Σ_k^{ta} iff there is a recursive function Φ of $\mathbb{N}^k$ into the class of d-dimensional rational open cubes such that*

$$S = \begin{cases} \bigcup\limits_{n_1 \in \mathbb{N}} \bigcap\limits_{n_2 \in \mathbb{N}} \cdots \bigcup\limits_{n_k \in \mathbb{N}} \Phi(n_1, n_2, \ldots, n_k) & \text{if } k \text{ is odd,} \\ \bigcup\limits_{n_1 \in \mathbb{N}} \bigcap\limits_{n_2 \in \mathbb{N}} \cdots \bigcap\limits_{n_k \in \mathbb{N}} \overline{\Phi(n_1, n_2, \ldots, n_k)} & \text{if } k \text{ is even.} \end{cases}$$

Herein the overline denotes the complement: $\overline{\Phi(n_1,n_2,\ldots,n_k)} = \mathbb{R}^d \setminus \Phi(n_1,n_2,\ldots,n_k)$.

Proof. Every representation of S like in the proposition fulfils the requirements from the definition of the Σ_k^{ta}-sets.

Conversely, in the proof of Proposition 3.1, we have shown that every representation of form $\bigcup_{n_k\in\mathbb{N}} \mathrm{set}(\Psi(n_1,n_2,\ldots,n_k))$, where any $\mathrm{set}(\Psi(n_1,n_2,\ldots,n_k))$ is open, can effectively be transformed into a form $\bigcup_{n_k\in\mathbb{N}} \Phi(n_1,n_2,\ldots,n_k)$, with rational open cubes $\Phi(n_1,n_2,\ldots,n_k)$. This shows the direction "$\rightarrow$" if k is odd.

If k is even and any $\mathrm{set}(\Psi(n_1,n_2,\ldots,n_k))$ is closed, we consider the sets

$$S_{(n_1,n_2,\ldots,n_{k-1})} = \bigcap_{n_k\in\mathbb{N}} \mathrm{set}(\Psi(n_1,n_2,\ldots,n_k)).$$

Now $\overline{S_{(n_1,n_2,\ldots,n_{k-1})}} = \bigcup_{n_k\in\mathbb{N}} \overline{\mathrm{set}(\Psi(n_1,n_2,\ldots,n_k))}$, and the $\overline{\mathrm{set}(\Psi(n_1,n_2,\ldots,n_k))}$ are open sets. Thus, one effectively obtains a representation

$$\overline{S_{(n_1,n_2,\ldots,n_{k-1})}} = \bigcup_{n_k\in\mathbb{N}} \Phi(n_1,n_2,\ldots,n_k),$$

with rational open cubes $\Phi(n_1,n_2,\ldots,n_k)$, and

$$S_{(n_1,n_2,\ldots,n_{k-1})} = \bigcap_{n_k\in\mathbb{N}} \overline{\Phi(n_1,n_2,\ldots,n_k)}. \qquad \square$$

The sets of Σ_1^{ta} are just the recursively open sets, whereas Π_1^{ta} consists of the recursively closed sets, and $\Delta_1^{\mathrm{ta}} = \{\emptyset, \mathbb{R}, \mathbb{R}^2, \mathbb{R}^3, \cdots\}$. So the 1-dimensional sets from Π_2^{ta} are well-known as recursively G_δ sets, cf. [39, 20], Σ_2^{ta} consists of the recursively F_σ sets, etc. More generally, the TAH represents just the effective counterpart of the hierarchy of Borelian subsets of finite order in the spaces $\mathbb{R}^d$, see also [14, 25]. Moreover, like in classical recursion theory [29, 26], our hierarchies contain d-ary relations over $\mathbb{R}$, $S \subseteq \mathbb{R}^d$, for arbitrary dimensions $d \in \mathbb{N}_+$.

By Proposition 5.1, it follows

Lemma 6.2. *Let $S \subseteq \mathbb{R}^d$. Then $S \in \Sigma_1^{\mathrm{ta}}$ iff $S = \mathrm{Halt}_d(\mathscr{A})$, for a d-robust* FAP *$\mathscr{A}$; and $S \in \Pi_1^{\mathrm{ta}}$ iff $S = \mathrm{Pass}_d(\mathscr{A})$, for a d-robust* FAP *$\mathscr{A}$.*

Considering the *discrete parts* of the (sets contained in the) classes of DAH or TAH, we essentially obtain the classical AH. More precisely, let

$$\mathrm{dis}(S) = S \cap \mathbb{N}^d \quad \text{for } S \subseteq \mathbb{R}^d,$$

and

$$\mathrm{dis}(\Gamma) = \{\mathrm{dis}(S)\colon S \in \Gamma\} \quad \text{for a class } \Gamma,$$

It is easily seen that $\mathbb{N}^d \in \Delta_1^{\mathrm{da}}$ and $\mathbb{N}^d \in \Pi_1^{\mathrm{ta}}$ (but $\mathbb{N}^d \notin \Sigma_1^{\mathrm{ta}}$). Moreover, the classes of the DAH and TAH are closed under (finite) intersections and under (finite) unions. So we have

Lemma 6.3. *For all classes Γ of the* DAH *and* TAH, *with the exception of Σ_1^{ta} and Δ_1^{ta},* $\mathrm{dis}(\Gamma) \subseteq \Gamma$.

Lemma 6.4. *For any $k \in \mathbb{N}_+$, we have*

$$\mathrm{dis}(\Sigma_k^{\mathrm{da}}) = \mathrm{dis}(\Sigma_k^{\mathrm{ta}}) = \Sigma_k^0,$$

$$\mathrm{dis}(\Pi_k^{\mathrm{da}}) = \mathrm{dis}(\Pi_k^{\mathrm{ta}}) = \Pi_k^0.$$

Moreover, $\mathrm{dis}(\Delta_k^{\mathrm{da}}) = \Delta_k^0$ *for all* $k \in \mathbb{N}_+$, *and* $\mathrm{dis}(\Delta_k^{\mathrm{ta}}) = \Delta_k^0$ *for all* $k > 1$.

Proof. We show the first line of equations, the second one follows analogously. Using Lemma 6.3, one obtains the assertion on the Δ-sets. If $S \in \Sigma_k^{\mathrm{da}}$ or $S \in \Sigma_k^{\mathrm{ta}}$, there is a representation of form

$$S = \{\boldsymbol{r} \in \mathbb{R}^d\colon\ \exists(z_1 \in \mathbb{N})\forall(z_2 \in \mathbb{N})\exists(z_3 \in \mathbb{N})\cdots\Diamond(z_k \in \mathbb{N})\ \boldsymbol{r} \in \mathrm{set}(\Psi(z_1, z_2, \ldots, z_k))\},$$

where Ψ is a recursive function of $\mathbb{N}^k$ into the set of FO-formulas with the free variables $x_1, \ldots, x_d$, and $\Diamond \in \{\exists, \forall\}$ such that the block of quantifiers becomes alternating.

The $(d+k)$-ary predicate over $\mathbb{N}$, $\psi(x_1, \ldots, x_d, z_1, \ldots, z_k) =$ "$\boldsymbol{x} \in \mathrm{set}(\Psi(z_1, \ldots, z_k))$" is recursively decidable. Thus, the set $\mathrm{dis}(S) = \{\boldsymbol{r} \in \mathbb{N}^d\colon\ \boldsymbol{r} \in S\}$ possesses a presentation characterizing the Σ_k^0-sets of the classical AH.

Conversely, let $S \in \Sigma_k^0$. Then there is a representation

$$S = \{(n_1, \ldots, n_d) \in \mathbb{N}^d\colon\ \exists(z_1 \in \mathbb{N})\forall(z_2 \in \mathbb{N})\exists(z_3 \in \mathbb{N})\cdots\Diamond(z_k \in \mathbb{N})$$
$$\psi(n_1, \ldots, n_d, z_1, z_2, \ldots, z_k)\},$$

where ψ is a recursive $(d+k)$-ary predicate over the natural numbers, and $\Diamond \in \{\exists, \forall\}$ as above.

If k is odd, from ψ one easily obtains a recursive total function Φ of $\mathbb{N}^k$ into the set of d-dimensional rational open cubes such that, for all $(n_1, \ldots, n_d, z_1, \ldots, z_{k-1}) \in \mathbb{N}^{d+k-1}$,

$$\exists(z_k \in \mathbb{N})\,\psi(n_1, \ldots, n_d, z_1, \ldots, z_k) \quad \text{iff} \quad \exists(z_k \in \mathbb{N})\,(n_1, \ldots, n_d) \in \Phi(z_1, \ldots, z_k).$$

Here the $\Phi(z_1, \ldots, z_k)$ may have the form $C_{1/2}^d(n_1, \ldots, n_d)$. Then $S = \mathrm{dis}(\widehat{S})$, for

$$\widehat{S} = \bigcup_{z_1 \in \mathbb{N}} \bigcap_{z_2 \in \mathbb{N}} \cdots \bigcup_{z_k \in \mathbb{N}} \Phi(z_1, z_2, \ldots, z_k).$$

For an even index k, a corresponding representation is analogously obtained. □

From Lemma 6.4 and the related property of the classical AH, we have the following proposition.

Proposition 6.2. *For all* $k \in \mathbb{N}_+$ *and* $\flat\mathrm{a} \in \{\mathrm{da}, \mathrm{ta}\}$,

$$\Delta_k^{\flat\mathrm{a}} \subset \left\{ \begin{matrix} \Sigma_k^{\flat\mathrm{a}} \\ \Pi_k^{\flat\mathrm{a}} \end{matrix} \right\} \subset \Sigma_k^{\flat\mathrm{a}} \cup \Pi_k^{\flat\mathrm{a}} \subset \Delta_{k+1}^{\flat\mathrm{a}},$$

$$\Sigma_k^{\flat\mathrm{a}} \not\subseteq \Pi_k^{\flat\mathrm{a}} \quad \textit{and} \quad \Pi_k^{\flat\mathrm{a}} \not\subseteq \Sigma_k^{\flat\mathrm{a}}.$$

From Proposition 6.2 and Lemma 6.3, it follows that $\Sigma_{k+1}^{\mathrm{ta}} \cap \Pi_{k+1}^{\mathrm{ta}} \not\subseteq \Sigma_k^{\mathrm{da}} \cup \Pi_k^{\mathrm{da}}$. So also the classes $\Sigma_{k+1}^{\mathrm{ta}}$ or Π_{k+1}^{ta} cannot be subsets of $\Sigma_k^{\mathrm{da}} \cup \Pi_k^{\mathrm{da}}$. On the other hand, the k-level classes of DAH are included in the $(k+1)$-level classes of TAH.

To show this, we consider a quantifier-free FO-formula $\Psi = \Psi(x_1, \ldots, x_d)$. It is equivalent to a (finite) disjunction of conjunctions of rational polynomial inequations,

$$\Psi(x_1, \ldots, x_d) \leftrightarrow \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} (p_{ij}(x_1, \ldots, x_d)\, \varrho_{ij}\, 0),$$

with (rational) polynomials p_{ij} and $\varrho_{ij} \in \{\leqslant, >\}$. In other words,

$$\mathrm{set}(\Psi(x_1, \ldots, x_d)) = \bigcup_{i=1}^{n} \bigcap_{j=1}^{m} \mathrm{set}(p_{ij}(x_1, \ldots, x_d)\, \varrho_{ij}\, 0).$$

It holds

$$\mathrm{set}(p_{ij}(x_1, \ldots, x_d) > 0) = \bigcup_{k \in \mathbb{N}} \mathrm{set}\left(p_{ij}(x_1, \ldots, x_d) - \frac{1}{k+1} \geqslant 0 \right),$$

$$\mathrm{set}(p_{ij}(x_1, \ldots, x_d) \leqslant 0) = \bigcap_{k \in \mathbb{N}} \mathrm{set}\left(p_{ij}(x_1, \ldots, x_d) - \frac{1}{k+1} < 0 \right).$$

Therefore, we have representations

$$\mathrm{set}(\Psi(x_1, \ldots, x_d)) = \bigcup_{i=1}^{n} \bigcap_{j=1}^{m} \bigcup_{k \in \mathbb{N}} \mathrm{set}(p_{ijk}^{c}(x_1, \ldots, x_d) \leqslant 0)$$

$$= \bigcup_{i=1}^{n} \bigcap_{j=1}^{m} \bigcap_{k \in \mathbb{N}} \mathrm{set}(p_{ijk}^{o}(x_1, \ldots, x_d) > 0),$$

with rational polynomials p_{ijk}^{c}, p_{ijk}^{o}. By the distributive laws saying that $\bigcap_{i=1}^{n} \bigcup_{j \in \mathbb{N}} S_{ij} = \bigcup_{(j_1, \ldots, j_n) \in \mathbb{N}^n} \bigcap_{i=1}^{n} S_{ij_i}$ and dually, combined with suitable modifications of the enumerations by means of Cantor's recursive n-tuple denumeration, one obtains

$$\mathrm{set}(\Psi(x_1, \ldots, x_d)) = \bigcup_{k \in \mathbb{N}} \bigcap_{l=1}^{L} \mathrm{set}(\tilde{p}_{kl}^{c}(x_1, \ldots, x_d) \leqslant 0)$$

$$= \bigcap_{k \in \mathbb{N}} \bigcup_{l=1}^{L} \mathrm{set}(\tilde{p}_{kl}^{o}(x_1, \ldots, x_d) > 0),$$

with certain (rational) polynomials $\tilde{p}_{kl}^{c}$, $\tilde{p}_{kl}^{o}$ and $L \in \mathbb{N}$.

Obviously, these representations of set($\Psi(x_1,\ldots,x_d)$) can effectively be obtained from the originally given formula Ψ. Applying Cantor's recursive pairing function over $\mathbb{N}$, two consecutive intersections and unions, respectively, can be combined in one. We now have the following proposition.

Proposition 6.3. *For any* $k \in \mathbb{N}_+$,

$$\Sigma_k^{\mathrm{da}} \subseteq \Sigma_{k+1}^{\mathrm{ta}}, \qquad \Pi_k^{\mathrm{da}} \subseteq \Pi_{k+1}^{\mathrm{ta}}, \qquad \Delta_k^{\mathrm{da}} \subseteq \Delta_{k+1}^{\mathrm{ta}}.$$

So we have seen that the classes of DAH and TAH are rather related to each other. It is still open if $\Sigma_k^{\mathrm{ta}} \subset \Sigma_k^{\mathrm{da}}$, for all $k \in \mathbb{N}_+$.

Now we want to classify the domains, graphs and ranges of computable functions within the corresponding arithmetical hierarchy, DAH resp. TAH.

With respect to the algebraic computability, we already know that Σ_1^{da} consists exactly of the domains of computable functions. Moreover, it is well-known that every range S of an algebraically computable function can also be represented as a domain of such a function, and conversely if $S \subseteq \mathbb{R}$, cf. [11]. Remark that the latter conversion would hold for any range $S \subseteq \mathbb{R}^d$ if algebraic computability would have been defined straightforwardly for vector-valued functions of types $f : \mathbb{R}^{d_1} \rightarrowtail \mathbb{R}^{d_2}$.

If a function f is algebraically computable, then the set graph(f) is the domain of another algebraically computable function, see [11]; thus, graph(f) $\in \Sigma_1^{\mathrm{da}}$. The conversion does not hold: for $f_{\pi}(r) = \sqrt{|r|}$, graph $(f_{\pi}) \in \Sigma_1^{\mathrm{da}}$, but f_{π} is not algebraically computable.

Since the class of algebraically computable functions of natural numbers, $f : \mathbb{N}^d \rightarrowtail \mathbb{N}$, coincides with the class of recursive functions, their domains and ranges are just the Σ_1^0-sets. Moreover, $f : \mathbb{N}^d \rightarrowtail \mathbb{N}$ is algebraically computable iff graph(f) $\in \Sigma_1^0$.

In the remaining part of this section, the related questions with respect to approximate computability will be discussed.

Proposition 6.4. *The domains of approximately computable functions* $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$ *are just the* Π_2^{ta}*-sets.*

Proof. For $d = 1$, this was already shown i n [22]. To prove direction "$\leftarrow$" within our framework, let $S \in \Pi_2^{\mathrm{ta}}$. By Proposition 6.1 and the definition of TAH, there is a representation

$$S = \bigcap_{n_1 \in \mathbb{N}} \bigcup_{n_2 \in \mathbb{N}} \Phi(n_1, n_2),$$

where Φ maps recursively $\mathbb{N}^2$ into the set of rational open cubes (of some dimension d).

To compute approximately the function $f_S = S \times \{0\}$, let an OTM $\mathcal{M}$ work as follows, on an input $n \in \mathbb{N}$ and an oracle $\Omega = (\Omega_l)_{l \in \mathbb{N}}$.

For all $n_1 \leqslant n$, compute $\Phi(n_1, k)$ and Ω_l, for the pairs $(k, l) \in \mathbb{N}^2$ (according to the ordering induced via Cantor's pairing function by the ordering of natural numbers) and decide whether $\Omega_l \subseteq \Phi(n_1, k)$.

When such a pair (k,l) has been found for every $n_1 \leqslant n$,
let $\mathscr{M}$ stop with the output $[-2^{-(n+2)}, 2^{-(n+2)}]$.
If such a pair does not exist for some $n_1 \leqslant n$,
let $\mathscr{M}$ work ad infinitum, i.e., $\mathscr{M}^{\Omega}(n)$ is undefined.

If $\boldsymbol{r} \in S$ and $\mathrm{tar}(\Omega) = \boldsymbol{r}$, then, for every $n_1 \in \mathbb{N}$, there is a $k \in \mathbb{N}$ such that $\boldsymbol{r} \in \Phi(n_1,k)$. Since the $\Phi(n_1,k)$ are open cubes, it follows $\Omega_l \subseteq \Phi(n_1,k)$, for some $l \in \mathbb{N}$. Thus, $\mathscr{M}^{\Omega}(n)$ is always defined. Obviously, $\mathrm{tar}((\mathscr{M}^{\Omega}(n))_{n\in\mathbb{N}}) = 0$.

Conversely, let $\mathscr{M}^{\Omega}(n)$ be always defined, for some regular sequence Ω. Then it holds $\mathrm{tar}((\mathscr{M}^{\Omega}(n))_{n\in\mathbb{N}}) = 0$, and for any $n_1 \in \mathbb{N}$, there is a pair (k,l) such that $\Omega_l \subseteq \Phi(n_1,k)$. It follows $\mathrm{tar}(\Omega) \in S$.

So we have seen that $\mathscr{M}$ approximately computes the function f_S.

For direction "$\rightarrow$" of the proof, we first remark that every bounded function, say $f : \mathbb{R}^d \rightarrowtail [-K,K]$, which is approximately computable, can approximately be determined by a $(d+1)$-robust FAP $\mathscr{A}$ such that

$$P_{d+1,t}(\mathscr{A}) \subseteq \mathbb{R}^d \times [-(K+1), K+1] \quad \text{for all } t \in \mathbb{N}.$$

Indeed, given a FAP $\mathscr{A}'$ approximately determining f, the tuples $(\boldsymbol{r}, y) \in P_{d+1,t}(\mathscr{A}')$ with $y \notin [-(K+1), K+1]$ can be replaced by the two tuples $(\boldsymbol{r}, -(K+1))$ and $(\boldsymbol{r}, K+1)$. This yields a FAP $\mathscr{A}_1$ determining the same function f. Its robustification, according to the proof of Theorem 5.1, satisfies all the required properties. The details and verification are left to the reader.

Now let $S = \mathrm{dom}(f_S)$, for an approximately computable function $f_S : \mathbb{R}^d \rightarrowtail \mathbb{R}$. By Lemma 3.2, we can suppose that $f_S = S \times \{0\}$. By Lemma 5.1, we have

$$S = \{\boldsymbol{r} \colon \exists^{=1} y[(\boldsymbol{r}, y) \in \mathrm{Pass}_{d+1}(\mathscr{A})]\},$$

for a $(d+1)$-robust FAP $\mathscr{A}$. Due to the remark above, we can suppose that

$$P_{d+1,t}(\mathscr{A}) \subseteq \mathbb{R}^d \times [-1,1] \quad \text{on all levels } t.$$

Now we have

$$\boldsymbol{x} \in S \quad \text{iff} \quad (\boldsymbol{x},0) \in \mathrm{Pass}_{d+1}(\mathscr{A}) \wedge \forall y[(\boldsymbol{x}, y) \in \mathrm{Pass}_{d+1}(\mathscr{A}) \rightarrow y = 0].$$

The first condition is expressible as

$$\varphi_1(\boldsymbol{x}) \equiv \text{“}\forall (t \in \mathbb{N})[(\boldsymbol{x},0) \in P_{d+1,t}(\mathscr{A})]\text{”}.$$

Since the sets $P_{d+1,t}(\mathscr{A})$ are closed, this formula defines a Π_1^{ta}-set $S_1 = \{\boldsymbol{r} \in \mathbb{R}^d \colon \varphi_1(\boldsymbol{r})\}$. We consider the second condition,

$$\varphi_2(\boldsymbol{x}) \equiv \text{“}\forall y[(\boldsymbol{x}, y) \in \mathrm{Pass}_{d+1}(\mathscr{A}) \rightarrow y = 0]\text{”}.$$

Since $\mathscr{A}$ is $(d+1)$-robust,

$$\varphi_2(\boldsymbol{x}) \text{ iff } \forall(m\in\mathbb{N})\,\exists(t\in\mathbb{N})\,[\forall y((\boldsymbol{x},y)\in P_{d+1,t}(\mathscr{A})\rightarrow y\in(-2^{-m},2^{-m}))].$$

Due to our supposition on $\mathscr{A}$,

$$\neg\varphi_2(\boldsymbol{x}) \quad \text{iff} \quad \exists(m\in\mathbb{N})\forall(t\in\mathbb{N})[\exists y((\boldsymbol{x},y)\in P_{d+1,t}(\mathscr{A}) \wedge y\in[-1,1]\backslash(-2^{-m},2^{-m}))].$$

The sets

$$\{\boldsymbol{x}\in\mathbb{R}^d\colon \exists y((\boldsymbol{x},y)\in P_{d+1,t}(\mathscr{A})\wedge y\in[-1,1]\backslash(-2^{-m},2^{-m}))\}$$

are closed, and, by Proposition 4.1 and EQE, representing FO-formulas $\varphi_{(m,t)}(\boldsymbol{x})$ can effectively be obtained.

Therefore, $S_2=\{\boldsymbol{r}\in\mathbb{R}^d\colon \varphi_2(\boldsymbol{x})\}$ is a Π_2^{ta}-set, and $S=S_1\cap S_2\in\Pi_2^{\mathrm{ta}}$, too. □

Corollary 6.1. *The classes of domains of algebraically computable functions and of approximately computable functions, respectively, are incomparable.*

Proof. For any computable transcendental number r_0, $\{r_0\}\in\Pi_1^{\mathrm{ta}}\backslash\Sigma_1^{\mathrm{da}}\subseteq\Pi_2^{\mathrm{ta}}\backslash\Sigma_1^{\mathrm{da}}$. Indeed, by Lemma 3.3, $\mathbb{R}\backslash\{r_0\}$ is recursively open, i.e., $\{r_0\}\in\Pi_1^{\mathrm{ta}}$. From $\{r_0\}\in\Sigma_1^{\mathrm{da}}$, by Corollary 2.1, it would follow that r_0 is an algebraic number.

On the other hand, it is easily seen that $\mathbb{Q}\in\Sigma_1^{\mathrm{da}}$, but $\mathbb{Q}\notin\Pi_2^{\mathrm{ta}}$. To show the latter, let $\mathbb{Q}\subseteq S=\bigcap_{n_1\in\mathbb{N}}\bigcup_{n_2\in\mathbb{N}}\Phi(n_1,n_2)$, with open sets $\Phi(n_1,n_2)$, for some $S\subseteq\mathbb{R}$. Then $\mathbb{Q}\subseteq S_{n_1}=\bigcup_{n_2\in\mathbb{N}}\Phi(n_1,n_2)$, for all $n_1\in\mathbb{N}$, and all these sets S_{n_1} are open. Thus, $S=\mathbb{R}$, due to the density of $\mathbb{Q}$ in $\mathbb{R}$. □

Proposition 6.5. *For every function $f:\mathbb{R}^d\rightarrowtail\mathbb{R}$, if f is approximately computable then* $\mathrm{graph}(f)\in\Pi_2^{\mathrm{ta}}$, *but not conversely, not even if f is total, continuous and algebraically computable.*

Proof. If f is approximately computable, then the function $\tilde{f}=\mathrm{graph}(f)\times\{0\}$ is approximately computable, too. One easily shows this via OTMs. By Proposition 6.4, $\mathrm{graph}(f)=\mathrm{dom}(\tilde{f})\in\Pi_2^{\mathrm{ta}}$.

On the other hand, for the function $\hat{f}$ from the example at the end of Section 3, which is not approximately computable, $\mathrm{graph}(\hat{f})$ is an algebraically decidable set. By Lemma 6.1 and Proposition 6.3, $\mathrm{graph}(\hat{f})\in\Delta_1^{\mathrm{da}}\subseteq\Delta_2^{\mathrm{ta}}$. □

The class of ranges of approximately computable functions exceeds Π_2^{ta} considerably. It coincides with the class of 1-dimensional *projections* of Π_2^{ta}-sets. These are the sets $S\subseteq\mathbb{R}$, which are representable as

$$S=\{s\in\mathbb{R}\colon \exists(\boldsymbol{r}\in\mathbb{R}^d)\forall(n_1\in\mathbb{N})\exists(n_2\in\mathbb{N})[(\boldsymbol{r},s)\in\Phi(n_1,n_2)]\},$$

where Φ recursively maps $\mathbb{N}^2$ into the set of $(d+1)$-dimensional rational open cubes, cf. Proposition 6.1.

To show this assertion, let $S = \mathrm{ran}(f)$, for some approximately computable function $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$. Then $\mathrm{graph}(f) \in \Pi_2^{\mathrm{ta}}$, by Proposition 6.5. This yields a representation of S as given above.

Conversely, let S be a projection of a Π_2^{ta}-set as described above. Then the function $\bar{f} : \mathbb{R}^{d+1} \rightarrowtail \mathbb{R}$ defined by

$$\bar{f}(\boldsymbol{r}, s) = \begin{cases} s & \text{if } \forall(n_1 \in \mathbb{N})\exists(n_2 \in \mathbb{N})[(\boldsymbol{r}, s) \in \Phi(n_1, n_2)], \\ \text{undefined} & \text{otherwise,} \end{cases}$$

can easily be shown to be approximately computable, and $S = \mathrm{ran}(\bar{f})$.

Since $\mathbb{N}^d \in \Pi_1^{\mathrm{ta}} \subseteq \Pi_2^{\mathrm{ta}}$, every Σ_3^{ta}-set (of arbitrary dimension) is also a projection of a Π_2^{ta}-set. The conversion does probably not hold, but it would require a more detailed investigation of TAH to confirm this.

We have shown

Proposition 6.6. *A set $S \subseteq \mathbb{R}$ is the range of an approximately computable function iff it is the 1-dimensional projection of a Π_2^{ta}-set. In particular, every 1-dimensional Σ_3^{ta}-set is such a range.*

Finally, we are going to deal with the domains, graphs and ranges of approximately computable discrete functions $f : \mathbb{N}^d \rightarrowtail \mathbb{N}$. For a function $f : \mathbb{R}^d \rightarrowtail \mathbb{R}$, the *discrete part* is defined to be the function

$$\mathrm{dis}(f) = f \cap (\mathbb{N}^d \times \mathbb{N}).$$

Lemma 6.5. *If a real function f is approximately computable, then* $\mathrm{dis}(f)$, *too.*

Proof. One easily shows that the approximate computability from f transfers to the restriction $f_{|\mathbb{N}^d}$. Moreover, the identical function $\mathrm{id}_{\mathbb{N}}$ is approximately computable (over $\mathbb{R}$). By Lemma 3.2, $\mathrm{dis}(f) = \mathrm{id}_{\mathbb{N}} \circ f_{|\mathbb{N}^d}$ is approximately computable, too. □

Therefore, the domains and graphs of approximately computable discrete functions are just the discrete parts of the domains and graphs, respectively, of approximately computable real functions. So the domains are just the Π_2^0-sets, and all discrete graphs are Π_2^0-sets. An example of a function $\tilde{f} : \mathbb{N}^2 \rightarrowtail \mathbb{N}$, which is not approximately computable, whose graph belongs to Π_2^0 however, can be obtained as follows from the function $\hat{f}$ in the example at the end of Section 3:

$$\tilde{f}(n_1, n_2) = \begin{cases} \hat{f}(n_1/n_2) & \text{if } n_2 \neq 0, \\ 0 & \text{if } n_2 = 0. \end{cases}$$

We leave the details of verification to the reader.

The ranges of approximately computable discrete functions are just the 1-dimensional Σ_3^0-sets. Indeed, from $f : \mathbb{N}^d \rightarrowtail \mathbb{N}$ and $\mathrm{graph}(f) \in \Pi_2^0$, it follows that

$$\mathrm{ran}(f) = \{y \in \mathbb{N}: \exists(n \in \mathbb{N})[((\pi_1^d(n), \ldots, \pi_d^d(n)), y) \in \mathrm{graph}(f)]\} \in \Sigma_3^0.$$

Herein π_i^d denotes the ith component function of Cantor's recursive d-tuple function.

Conversely, similar to the related part of the proof of Proposition 6.6, every 1-dimensional Σ_3^0-set can be represented as a range of an approximately computable discrete function.

So we have shown

Proposition 6.7. *The domains of approximately computable discrete functions of form $f : \mathbb{N}^d \rightarrowtail \mathbb{N}$ are just the Π_2^0-sets, and the ranges of these functions are just those subsets of $\mathbb{N}$ which belong to Σ_3^0. All graphs of such functions are Π_2^0-sets, but not conversely.*

7. Final remarks

The main result of this paper is the characterization of approximate computability of real functions from the algebraic point of view. This has been done by Theorems 4.1 and 5.1 for the generalized concept including functions with not necessarily recursively open domains. Proposition 3.2 and Corollary 3.1 state the relationship to the Ko–Friedman variant of approximate computability.

Our characterization of approximate computability by means of the passing sets of FAPs does not depend on a special naming system or representation of the reals. We treat the real numbers as actual objects of algorithms. To deal with both algebraic and approximate computability from a uniform point of view, this should contribute to a better understanding of both the settings, their special features and mutual relationships. For example, our results also stress the naturalness and usefulness of the generalized notion of approximate computability in considering partial real functions.

Further research will try to find out whether the algebraic characterization of approximate computability leads also to substantially new results or at least to easier proofs of known results. To this purpose, more detailed investigations of the passing sets of (robust) FAPs are necessary. With respect to both variants of computability, further results on the arithmetical hierarchies, DAH and TAH, also compared with Cucker's [5] hierarchy, are desirable. Finally, like in the Ko–Friedman approach, a theory of computational complexity can be founded on the approximate determinations of functions by FAPs and seems to be a promising subject of further effort.

Acknowledgements

I am indebted to G. Asser and H. Köhler for discussions, critical remarks, and for reading previous versions of this paper. K. Weihrauch informed me about his results

from [22]. Moreover, some hints and remarks by anonymous referees are gratefully acknowledged.

References

[1] L. Blum, F. Cucker, M. Shub, S. Smale, Complexity and Real Computation, Springer, New York, 1998.
[2] L. Blum, M. Shub, S. Smale, On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, Bull. AMS 21 (1) (1989) 1–46.
[3] P. Boldi, S. Vigna, δ-uniform BSS machines, Preprint 1997, J. Complexity, to appear.
[4] V. Brattka, P. Hertling, Feasible real random access machines, Informatik-Berichte, Fern Univ. Hagen 193, 1995; revised version 1998, Theoret. Comput. Sci., to appear.
[5] F. Cucker, The arithmatical hierarchy over the reals, J. Logic Comput. 2 (3) (1992) 375–395.
[6] L. van den Dries, Alfred Tarski's elimination theory for real closed fields, J. Symbolic Logic 53 (1988) 7–19.
[7] E. Engeler, Algorithmic properties of structures, Math. System Theory 1 (1967) 183–195.
[8] A.P. Ershov, Abstract Computability on Algebraic Structures, Lecture Notes in Computer Science, vol. 122, 1981, pp. 397–420.
[9] H. Friedman, Algorithmic procedures, generalized Turing algorithms, and elementary recursion theory, Logic Colloquium 1969, North-Holland, Amsterdam, 1971, pp. 361–390.
[10] H. Friedman, R. Mansfield, Algorithmic procedures, Trans. AMS 332 (1992) 297–312.
[11] A. Hemmerling, Computability of string functions over algebraic structures, Math. Log. Quart. 44 (1998) 1–44.
[12] P. Hertling, K. Weihrauch, Levels of degeneracy and exact lower bounds for geometric algorithms, Proc. 6th Canadian Conf. on Computational Geometry, Saskatoon, 1994, pp. 237–242.
[13] H. Heuser, Das Unendliche in Philosophie, Theologie und Mathematik. DMV-Mitteilungen 3/97 (1997) 35–38.
[14] P.G. Hinman, Recursion-Theoretic Hierarchies, Springer, Berlin, 1978.
[15] G. Hotz, T. Chadzelek, Analytic machines, Preprint 1997.
[16] G. Hotz, G. Vierke, B. Schieffer, Analytic machines, Electron. Colloq. Comput. Complexity (1995) TR 95–025.
[17] Iu. I. Janov, The logical schemes of algorithms, Problems Cybernet. 1 (1960) 82–140.
[18] A.J. Kfoury, Definability by programs in first-order structures, Theoret. Comput. Sci. 25 (1983) 1–66.
[19] S.C. Kleene, Introduction to Metamathematics, North-Holland, Amsterdam, 1952.
[20] K.-I. Ko, Complexity Theory of Real Functions, Birkhäuser, Boston, 1991.
[21] K.-I. Ko, H. Friedman, Computational complexity of real functions, Theoret. Comput. Sci. 20 (1982) 323–352.
[22] C. Kreitz, K. Weihrauch, Complexity Theory on Real Numbers and Functions, Lecture Notes in Computer Science, vol. 145, 1983, pp. 165–174.
[23] D.C. Luckham, D.M.R. Park, M.S. Paterson, On formalised computer programs, JCSS 4 (1970) 220–249.
[24] F. Meyer auf der Heide, J. Wiedermann, Numerical RAM: a realistic machine model for scientific computing, Preprint 1997.
[25] Y.N. Moschovakis, Descriptive Set Theory, North-Holland, Amsterdam, 1980.
[26] P. Odifreddi, Classical Recursion Theory, North-Holland, Amsterdam, 1989.
[27] M.B. Pour-El, J.I. Richards, Computability in Analysis and Physics, Springer, Berlin, 1989.
[28] F.P. Preparata, M.I. Shamos, Computational Geometry, Springer, Berlin, 1985.
[29] H. Rogers Jr., Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, 1967.
[30] J.C. Shepherdson, On the definition of computable function of a real variable, Zeitschr. f. math. Logik und Grundlagen d. Math., Bd. 22 (1976) 391–402.
[31] J.C. Shepherdson, Algorithmic procedures, generalized Turing algorithms, and elementary recursion theory, in: L.H. Harrington et al. (Eds.), Harvey Friedman's Research on the Foundations of Mathematics, North-Holland, Amsterdam, 1985, pp. 285–308.

[32] J.C. Shepherdson, Computational complexity of real functions, in: L.H. Harrington et al. (Eds.), Harvey Friedman's Research on the Foundations of Mathematics, North-Holland, Amsterdam, 1985, pp. 309–315.
[33] A. Tarski, A Decision Method for Elementary Algebra and Geometry, University of California Press, Berkeley, 1951.
[34] J.V. Tucker, J.I. Zucker, Program Correctness Over Abstract Data Types, with Error-State Semantics, North-Holland P.C., Amsterdam, 1988.
[35] J.V. Tucker, J.I. Zucker, Computation by 'while' programs on topological partial algebras, McMaster University, Department. of Computer Science, 1998, TR 98-02.
[36] J.V. Tucker, J.I. Zucker, Computable functions and semicomputable sets on many-sorted algebras, Preprint 1998, in: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (Eds.), Handbook of Logic in Computer Science, vol. 5, Oxford Univ. Press, Oxford, to appear.
[37] A.M. Turing, On computable numbers, with an application to the Entscheidungsproblem, Proc. London Math. Soc., vol. 42, 1936, pp. 230–265; vol. 43, 1937, pp. 544–546.
[38] K. Weihrauch, Computability, Springer, Berlin, 1987.
[39] K. Weihrauch, A simple introduction to computable analysis, Informatik-Berichte 171-7/1995, FernUniv. Hagen, 1995.
[40] N. Zhong, Recursively enumerable subsets of R^q in two computing models – Blum-Shub-Smale machine and Turing machine, Theoret. Comput. Sci. 197 (1998) 79–94.