Lecture Notes in Computer Science 2064

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Jens Blanck Vasco Brattka
Peter Hertling (Eds.)

Computability and Complexity in Analysis

4th International Workshop, CCA 2000
Swansea, UK, September 17-19, 2000
Selected Papers

Springer

Volume Editors

Jens Blanck
University of Wales Swansea
Singleton Park, Swansea SA2 8PP, UK
E-mail: csjens@swan.ac.uk

Vasco Brattka
Peter Hertling
FernUniversität Hagen, Informatikzentrum
58084 Hagen, Germany
E-mail: {Vasco.Brattka,Peter.Hertling}@FernUni-hagen.de

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Computability and complexity in analysis : 4th international workshop ;
selected papers / CCA 2000, Swansea, UK, September 17 - 19, 2000.
Jens Blanck ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ;
Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2064)
ISBN 3-540-42197-1

CR Subject Classification (1998): F.1.3, F.2, F.4.1, G.1

ISSN 0302-9743
ISBN 3-540-42197-1 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

Typesetting: Camera-ready by author, data conversion by Christian Grosche, Hamburg
Printed on acid-free paper SPIN: 10781632 06/3142 5 4 3 2 1 0

Preface

The workshop on Computability and Complexity in Analysis, CCA 2000, was hosted by the Department of Computer Science of the University of Wales Swansea, September 17–19, 2000. It was the fourth workshop in a successful series of workshops: CCA'95 in Hagen, Germany, CCA'96 in Trier, Germany, and CCA'98 in Brno, Czech Republic. About 40 participants from the countries United Kingdom, Germany, Japan, Italy, Russia, France, Denmark, Greece, and Ireland contributed to the success of this meeting. Altogether, 28 talks were presented in Swansea. These proceedings include 23 papers which represent a cross-section through recent research on computability and complexity in analysis. The workshop succeeded in bringing together people interested in computability and complexity aspects of analysis and in exploring connections with numerical methods, physics and, of course, computer science. It was rounded off by a number of talks and papers on exact computer arithmetic and by a competition of five implemented systems. A report on this competition has been included in these proceedings. We would like to thank the authors for their contributions and the referees for their careful work, and we hope for further inspiring and constructive meetings of the same kind.

April 2001

Jens Blanck
Vasco Brattka
Peter Hertling

Organization

CCA 2000 was hosted by the Department of Computer Science of the University of Wales Swansea and took place on September 17–19, 2000.

Program Committee

Ker-I Ko	(Stony Brook, USA)
Marian Boykan Pour-El	(Minnesota, USA)
Dana Scott	(Carnegie-Mellon, USA)
Viggo Stoltenberg-Hansen	(Uppsala, Sweden)
John V. Tucker	(Swansea, Wales)
Klaus Weihrauch, chair	(Hagen, Germany)
Mariko Yasugi	(Kyoto Sangyo, Japan)
Jeffery Zucker	(McMaster, Canada)

Organizing Committee

Jens Blanck	(Swansea, Wales)
Vasco Brattka	(Hagen, Germany)
Peter Hertling	(Hagen, Germany)

Table of Contents

Computability and Complexity in Analysis

Effectivity of Regular Spaces

Jens Blanck*

University of Wales Swansea, Singleton Park, Swansea, SA2 8PP, UK

Abstract. General methods of investigating effectivity on regular Hausdorff (T_3) spaces is considered. It is shown that there exists a functor from a category of T_3 spaces into a category of domain representations. Using this functor one may look at the subcategory of effective domain representations to get an effectivity theory for T_3 spaces. However, this approach seems to be beset by some problems. Instead, a new approach to introducing effectivity to T_3 spaces is given. The construction uses effective retractions on effective Scott–Ershov domains. The benefit of the approach is that the numbering of the basis and the numbering of the elements are derived at once.

1 Introduction

Domain theory has been used as a successful means to study effectivity on various spaces via domain representations of the spaces. This is due to the natural effectivity theory for domains and the inherit notion of approximation that exists within domains.

Representations of topological spaces by domains or embeddings of topological spaces into domains have been studied by several people. Weihrauch and Schreiber [21] considered embeddings of metric spaces into cpos with weight and distance. Stoltenberg-Hansen and Tucker [17, 18] introduced the notion of domain representability. Edalat [4–7] has used embeddings into continuous dcpos to study integration, measures and fractals. Edalat and Heckmann [8] and di Gianantonio [3] among others have also studied similar notions. Ershov's [9] representation of the Kleene–Kreisel continuous functionals is an early example of a domain representation.

A result by the author [2] characterises the T_3 spaces as exactly the ones that have a certain type of domain representations. It is therefore a natural desire to study the effectivity theory induced by this class of domain representations. This is the main aim of the paper.

Investigations of effectivity on topological spaces, even much weaker topologies, has been studied by Spreen [15] and Kreitz and Weihrauch [11, 20] among others. An effective regularity condition has been considered by Schröder in [13]. However, that condition was strong enough to imply that the space had an effective metric.

* Supported by STINT, The Swedish Foundation for International Cooperation in Research and Higher Education.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 1–15, 2001.

Section 2 gives some basic definitions and recalls some results. Section 3 gives a lifting result for continuous functions to countable based domain representations. Section 4 gives a full and faithful functor from a category of topological spaces into a category of domain representations. The problem here is that there is no canonical choice of a numbering for the basis. Section 5 is the main section introducing the new approach to constructing domain representations and in particular effective domain representations. The effective representations that are considered are those that can be obtained from effective retractions on effective domains. The primary benefit of this approach is that it gives numberings for both the elements and for a basis for the topology. Although it is shown that any space has representations constructed according to this approach, it is not clear that every interesting effective representation can be obtained in this manner. For the effective representations of T_3 spaces obtained some effective properties are investigated. For example, it is not always possible to compute the operation that takes an index of a filter base of a point to an index of the point.

During the preparation of this script I have benefitted from conversations with Andrej Bauer, Ulrich Berger, Pino Rosolini, Dieter Spreen, Viggo Stoltenberg-Hansen, and John V. Tucker.

2 Preliminaries

A space is T_3 if it is regular and T_1. Thus, any T_3 space is Hausdorff. The topological closure and interior of a set S is denoted by $\overline{S}$ and S° respectively.

We will use $\langle \cdot, \cdot \rangle$ to denote some standard recursive pairing operation on the natural numbers ω. The recursive projections π_0 and π_1 are assumed to satisfy $\pi_0\langle m, n\rangle = m$ and $\pi_1\langle m, n\rangle = n$. Fix $(W_n)_{n\in\omega}$ to be a standard enumeration of the r.e. sets.

We assume familiarity with the notion of domain representations [2, 18] and of domain theory, in particular, the theory of effective domains [16]. By a domain in this paper is meant a Scott–Ershov domain, i.e., a consistently complete algebraic cpo. We denote the compact elements of a domain D by D_{c}.

Definition 2.1. A *domain representation* of a topological space X is a tuple $(D, D^{\mathrm{R}}, \rho)$ such that D is a domain, D^{R} is a subset of D, and $\rho\colon D^{\mathrm{R}} \to X$ is a quotient map.

A domain representation is *upwards-closed* if D^{R} is an upper subset of D, and if $x \sqsubseteq y$ and $x \in D^{\mathrm{R}}$ implies $y \in D^{\mathrm{R}}$ and $\rho(x) = \rho(y)$.

If $(D, D^{\mathrm{R}}, \rho)$ is a domain representation of X and there exists an topological embedding $\eta\colon X \to D^{\mathrm{R}}$ then the tuple $(D, D^{\mathrm{R}}, \rho, \eta)$ is a *retract* domain representation of X.

We will usually consider only upwards-closed retract domain representation. The following two theorems are proven in [2].

Theorem 2.2. *Any T_3 space has a dense upwards-closed retract domain representation.*

In fact, the proven result is stronger in that the obtained domain representation has further nice properties. These extra properties will not be used in this paper, however.

Theorem 2.3. *Let $(D, D^{\mathrm{R}}, \rho, \eta)$ be an upwards-closed retract representation of X. Then X is T_3.*

The following easy result shows that functions induced from domain representations are continuous.

Proposition 2.4. *Let $(D, D^{\mathrm{R}}, \varphi)$ and $(E, E^{\mathrm{R}}, \psi)$ be domain representations of X and Y respectively. Let $\bar{f}\colon D \to E$ be continuous such that $\bar{f}[D^{\mathrm{R}}] \subseteq E^{\mathrm{R}}$ and assume $\bar{f}$ respects the equivalence relations induced by φ and ψ. Then $\bar{f}$ induces a unique continuous function $f\colon X \to Y$.*

3 Liftings of Functions to Non-dense Representations

It is known that a continuous functions defined on a dense subset of a domain D into a Scott–Ershov domain can be extended to the domain D. See for example [10]. Following an idea of Geir Waagbø [19], we show that denseness is not needed for countably based coherent domains. The actual proof is a generalisation of the proof given by Dag Normann [12].

Definition 3.1. A domain representation is *coherent* if the domain is coherent, i.e., if any inconsistent finite set of elements contains an inconsistent pair of elements.

Examples of coherent domain representations include the flat domain representation of any countable set, and the usual interval representation of the reals.

Lemma 3.2. *Let D be a domain and E a coherent domain, then the function space $[D \to E]$ is coherent.*

Proof. Let $F = \{f_1, \ldots, f_n\}$ be a finite pairwise consistent set of functions in $[D \to E]$. Define a function $g\colon D \to E$ by

$$g(x) = \bigsqcup_{1 \le i \le n} f_i(x)\,.$$

Since F is pairwise consistent, it follows that $\{f_i(x) : 1 \le i \le n\}$ is a pairwise consistent set in the coherent domain E and therefore consistent. Hence $g(x)$ is well-defined for all $x \in D$. Clearly, $f_i \sqsubseteq g$ for all $i = 1, \ldots, n$. Let $h\colon D \to E$ be an upper bound of F. For all $i = 1, \ldots, n$ and all $x \in D$ we have $f_i(x) \sqsubseteq h(x)$, and hence $g(x) = \bigsqcup_{1 \le i \le n} f_i(x) \sqsubseteq h(x)$. Thus $g \sqsubseteq h$, i.e., g is the supremum $\bigsqcup_{1 \le i \le n} f_i$.

It remains to show that g is continuous. Assume that $x \sqsubseteq y$. By monotonicity of the function f_i it follows that

$$g(x) = \bigsqcup_{1\leq i\leq n} f_i(x) \sqsubseteq \bigsqcup_{1\leq i\leq n} f_i(y) = g(y)\,.$$

Assume that A is a directed subset of D and let $x = \bigsqcup_D A$. Then

$$g(x) = \bigsqcup_{1\leq i\leq n} f_i(\bigsqcup_D A) = \bigsqcup_{1\leq i\leq n} \bigsqcup_E f_i[A] = \bigsqcup_E \bigsqcup_{1\leq i\leq n} f_i[A] = \bigsqcup_E g[A]\,.$$

□

Let (D, D^{R}, ν) be a countably based domain representation of X and $(E, E^{\mathrm{R}}, \rho, \eta)$ be a coherent retract domain representation of Y. Let $\varphi\colon X \to Y$ be a continuous function. We will lift φ to a domain function $g\colon D \to E$.

Define $f\colon D^{\mathrm{R}} \to E^{\mathrm{R}}$ by $f = \eta\varphi\nu$. We will show that there exists a function $g\colon D \to E$ such that $f = g|_{D^{\mathrm{R}}}$. Let $(a_n, b_n)_{n\in\omega}$ be an enumeration of all pairs (a_n, b_n) such that

(i) $\uparrow a_n \cap D^{\mathrm{R}} \neq \emptyset$, and
(ii) $f[\uparrow a_n \cap D^{\mathrm{R}}] \subseteq \uparrow b_n$.

Let Γ be the set of all basic step functions $\langle a; b\rangle$ for which there exists an n such that

(i) $a_n \sqsubseteq a$ and $b \sqsubseteq b_n$, and
(ii) for all $i < n$, b_i and b_n inconsistent implies that a and a_i inconsistent.

Lemma 3.3. *Any finite subset of Γ is consistent.*

Proof. By coherency it is sufficient to show that any $\langle a; b\rangle$ and $\langle a'; b'\rangle$ in Γ are consistent. There exists n and n' witnessing that the step functions belong to Γ. If $n = n'$ then $b \sqsubseteq b_n$ and $b' \sqsubseteq b_{n'} = b_n$ showing that the step functions are consistent. Assume that $n < n'$, the remaining case is symmetric. Suppose b and b' are inconsistent. Since $b \sqsubseteq b_n$ and $b' \sqsubseteq b_{n'}$ we have that b_n and $b_{n'}$ are inconsistent. By condition (ii) in the definition of Γ it follows that a_n and a' are inconsistent. Hence, since $a_n \sqsubseteq a$, we have that a and a' are inconsistent. Thus, the step functions are consistent. □

Define g to be the function obtained from the ideal generated by Γ.

Lemma 3.4. *The function g is an extension of f.*

Proof. Clearly, $g|_{D^{\mathrm{R}}} \sqsubseteq f$. Let $x \in D^{\mathrm{R}}$, and let $b \sqsubseteq fx$. By continuity, the preimage $f^{-1}[\uparrow b]$ is open. Thus, there exists $a \sqsubseteq x$ such that $\uparrow a \subseteq f^{-1}[\uparrow b]$. Hence, the pair (a, b) belongs to the enumeration. Let n be the least index such that $(a, b) = (a_n, b_n)$.

We will now find c such that the step function $\langle c; b\rangle$ belongs to Γ.

Let $i < n$. Assume that b_i and b are inconsistent. Then a_i and x must be inconsistent, since if they are consistent, then $f(a_i \sqcup x) = f(x) \in \uparrow b_i \cap \uparrow b$

contradicting that b_i and b were inconsistent. Choose $c_i \in \mathrm{approx}(x)$ such that a_i and c_i are inconsistent. Let

$$c = a \sqcup \bigsqcup_{i \in I} c_i \,,$$

where I is the set of all $i < n$ such that b_i and b are inconsistent. The supremum exists since x is an upper bound. The step function $\langle c; b\rangle$ belongs to Γ. Thus, $gx = fx$. □

Proposition 3.5. *Let (D, D^{R}, ν) be a countably based domain representation of X and $(E, E^{\mathrm{R}}, \rho, \eta)$ be a coherent upwards-closed retract domain representation of Y. Then any continuous function $\varphi\colon X \to Y$ can be lifted to a domain function $g\colon D \to E$.*

Proof. Define a function $f\colon D^{\mathrm{R}} \to E^{\mathrm{R}}$ by $f = \eta\varphi\nu$. By Lemmas 3.3 and 3.4 the function f can be extended to a function $g\colon D \to E$. □

4 The Category DR

We introduce the category **DR** of upwards-closed retract domain representations and show that there exists a functor from a category of topological spaces into this category. We start by defining our categories.

The category **DR** is the category of upwards-closed retract domain representations [2]. Formally, the category **DR** of domain representations has as objects all tuples $(D, X, D^{\mathrm{R}}, \rho, \eta)$ where D is a domain, X is a topological space, D^{R} is an upper set of D, and $(\rho\colon D^{\mathrm{R}} \to X, \eta\colon X \to D^{\mathrm{R}})$ is a retraction-embedding pair between D^{R} and X such that if y is above some $x \in D^{\mathrm{R}}$ then $\rho y = \rho x$. Another way of expressing the requirements is to say that the space X is the retract of an upper set D^{R} where the retraction ρ is order-collapsing. By Theorem 2.3, the space X must be T_3. The retraction ρ induces an equivalence relation $\sim_D$ on D^{R} by

$$x \sim_D y \iff \rho x = \rho y \,.$$

Let $(D_1, X_1, D_1^{\mathrm{R}}, \rho_1, \eta_1)$ and $(D_2, X_2, D_2^{\mathrm{R}}, \rho_2, \eta_2)$ be domain representations. Consider the set

$$F = \left\{ f\colon D_1 \to D_2 \mid f[D_1^{\mathrm{R}}] \subseteq D_2^{\mathrm{R}} \text{ and } x \sim_{D_1} y \implies fx \sim_{D_2} fy \right\} .$$

Define an equivalence relation $\sim$ on F by

$$f \sim g \iff \forall x \in D_1^{\mathrm{R}} (fx \sim_{D_2} gx) \,.$$

A morphism of the category **DR** is an equivalence class with respect to $\sim$.

Let $\mathbf{T_3}$ be the category of all pairs $(X, \mathcal{S})$, where X is a T_3 topological space and $\mathcal{S}$ is a subbase for the topology on X. A morphism of $\mathbf{T_3}$ is a continuous function.

The forgetful functor $U\colon \mathbf{DR} \to \mathbf{T_3}$ is defined as follows. Let the object part of U be $(D, X, D^{\mathrm{R}}, \rho, \eta) \mapsto (X, \mathcal{B})$, where the base $\mathcal{B}$ is $\{\eta^{-1}[\uparrow a] : a \in D_{\mathrm{c}}\}$. By Proposition 2.4 we have that each morphism $[\bar{f}]\colon D_1 \to D_2$ uniquely determines a morphism $f\colon X_1 \to X_2$. Let the morphism part of U be the map that takes $[\bar{f}]$ to the uniquely determined f.

4.1 The Functor $R\colon \mathbf{T_3} \to \mathbf{DR}$

In this section we show that there exists a full and faithful functor from the category $\mathbf{T_3}$ of topological spaces to the category $\mathbf{DR}$ of domain representations.

Let $(X, \mathcal{S}) \in \mathbf{T_3}$. The family

$$P = \{\overline{S_1} \cap \cdots \cap \overline{S_n} : n < \omega, S_i \in \mathcal{S}, \overline{S_1} \cap \cdots \cap \overline{S_n} \neq \emptyset\}$$

is a *neighbourhood system* in the sense of [2]. Define $\sqsubseteq$ on P by $a \sqsubseteq b \iff b \subseteq a$. The ideal completion $D_{X,\mathcal{S}} = \mathrm{Idl}(P, \sqsubseteq)$ is a domain and $D_{X,\mathcal{S}} = (D_{X,\mathcal{S}}, X, D^{\mathrm{R}}_{X,\mathcal{S}}, \rho_{D_{X,\mathcal{S}}}, \eta_{D_{X,\mathcal{S}}})$ is a domain representation of X. (This is the construction used in the proof of Theorem 2.2). The set of representing elements is

$$D^{\mathrm{R}}_{X,\mathcal{S}} = \{I \in D_{X,\mathcal{S}} : \bigcap I = \{x\} \text{ for some } x \in X\}\,.$$

The retraction $\rho_{D_{X,\mathcal{S}}}$ is defined by

$$\rho_{D_{X,\mathcal{S}}}(I) = x \iff \bigcap I = \{x\}\,,$$

and the embedding $\eta_{D_{X,\mathcal{S}}}$ is defined by

$$\eta_{D_{X,\mathcal{S}}}(x) = \{a \in P : x \in a^\circ\}\,.$$

The representation $D_{X,\mathcal{S}}$ is, in fact, dense, i.e., $D^{\mathrm{R}}_{X,\mathcal{S}}$ is dense in $D_{X,\mathcal{S}}$.

Let R denote the map $(X, \mathcal{S}) \mapsto D_{X,\mathcal{S}}$.

Definition 4.1. Let X be a T_3 space and let $\mathcal{S}$ be a subbasis for the topology on X. The *standard domain representation* of $(X, \mathcal{S})$ is $R(X, \mathcal{S}) = D_{X,\mathcal{S}}$.

If the subbase is clear from the context we will sometimes say that we have a standard domain representation of the space X.

Given a morphism $f\colon X_1 \to X_2$ in $\mathbf{T_3}$ we can construct a continuous domain function $\bar{f}$, from a standard domain representation of X_1 to a standard domain representation of X_2, such that the morphism $[\bar{f}]\colon R(X_1, \mathcal{S}_1) \to R(X_2, \mathcal{S}_2)$ satisfies $f\rho_{X_1} = \rho_{X_2}[\bar{f}]$. We call such an $\bar{f}$ a *representation* or a *lifting* of f. The function $\bar{f}$ is defined as the extension of $\eta_2 f \rho_1\colon D^{\mathrm{R}}_1 \to D^{\mathrm{R}}_2$ to D_1. The extension exists since $\eta_2 f \rho_1$ is a continuous function from a dense subset of D_1 into an injective space D_2, see [10].

Let R map a morphism f to the morphism $[\bar{f}]$.

Proposition 4.2. *The map $R\colon \mathbf{T_3} \to \mathbf{DR}$ is a full and faithful functor.*

Proof. The choice of base for the domain representations is irrelevant for this argument and is dropped for readability.

A function f in the equivalence class $R\,\mathrm{id}_X$ maps each element in $(RX)^{\mathrm{R}} = D_X^{\mathrm{R}}$ to an element equivalent ($\sim$) to itself. Thus, $R\,\mathrm{id}_X$ clearly acts as the identity with respect to composition, i.e., $[\mathrm{id}_{RX}] = R\,\mathrm{id}_X$.

Let $f\colon X_1 \to X_2$ and $g\colon X_2 \to X_3$. We have

$$\rho_{X_3} \circ Rf \circ Rg = f \circ \rho_{X_2} \circ Rg = f \circ g \circ \rho_{X_1} = \rho_{X_3} \circ R(f \circ g)\,,$$

that is, compositions are preserved by R.

Recall that each morphism $[\bar{f}]$ in **DR** uniquely determines a morphism f between the underlying topological spaces. Clearly, Rf must be $[\bar{f}]$. Hence, the functor R is full.

If $Rf_1 = Rf_2$ then the uniquely determined functions on the underlying spaces must be identical, i.e., $f_1 = f_2$. Hence, R is faithful. □

The functor R is not an isomorphism since not all domain representations can be obtained as RX for some X, e.g., representations that are not dense.

It is true that the space X is preserved by the composition UR. However, the base $\mathcal{B}$ obtained from the standard domain representation $R(X, \mathcal{S})$ is not necessarily the base $\mathcal{B}'$ obtained by taking all finite intersections of sets in $\mathcal{S}$. The base $\mathcal{B}$ consists of interiors of the closures of sets in $\mathcal{B}'$, which are the same only if the sets in $\mathcal{B}'$ are regular ($\overline{B}^{\circ} = B$).

The existence of a full and faithful functor is encouraging as a basis for introducing effectivity. However, the effectivity introduced would depend both on the subbasis chosen to build the domain representation and the derived basis obtained from this representation. This would probably lead to an unwieldy theory of effectivity, so we will look for an alternative way of introducing effectivity.

5 An Approach to Effectivity on Regular Spaces

Upwards-closed retract domain representations can be derived from any retraction on a domain. This construction is general in the sense that all T_3 spaces can be given representations. However, it may be the case that a particular retract domain representation cannot be reconstructed from a retraction on a domain. In particular, there might exist a non-dense representation that cannot be constructed from a retraction. For countably based domain representations we can show, using Proposition 3.5, that any coherent upwards-closed retract domain representation can be reconstructed from a domain retraction. Hence, we know that any effective coherent domain representation may be obtained using this construction. Although this is encouraging, one should note that the domain retraction need not be effective, even if the domain representation is effective, since Proposition 3.5 is non-effective.

5.1 Deriving Domain Representations from Retractions

Definition 5.1. A *retraction* on a domain D is a continuous function r such that $r \circ r = r$.

Let r be a retraction on a domain D. We aim to construct upwards-closed retract domain representations of subsets of D using the retraction r and the inclusion map.

The set $\mathrm{Fix}(r)$ of all fixed points under r is clearly equal to the forward image $r[D]$. Hence, the subsets of D that get representations must in particular be subsets of $r[D]$. Let X be a subset of fixed points, and let $D^{\mathrm{R}} = r^{-1}[X]$. If for some $x \in X$ there exists a $x' \in r[D]$ such that $x \sqsubseteq x'$ then the representation cannot be upwards-closed, since x clearly is a representation of itself and x' is above x but not an approximation of x. We will therefore require the set X to consist of *maximal* fixed points.

The intended domain representation of the space X is $(D, X, D^{\mathrm{R}}, r, \iota)$. The function $r\colon D^{\mathrm{R}} \to X$ is continuous if the topology on X is weaker than the quotient topology. On the other hand, the inclusion $\iota\colon X \to D^{\mathrm{R}}$ is continuous if the topology on X is stronger than the relative topology. The topology on D^{R} is taken to be the relative topology from the Scott topology on D.

Lemma 5.2. *The quotient topology on X induced by r is weaker than the relative topology on X.*

Proof. Let U be an open set in the quotient topology on X. Then $r^{-1}[U]$ is open. For $x \in X$ we have that $x \in U$ if, and only if, $x \in r^{-1}[U]$, since x is a fixed point under r. Thus, $U = X \cap r^{-1}[U]$, which shows that U is open in the relative topology. □

The above proof does not use any information about the domain structure so the result could be stated in a more general setting. Now, somewhat surprisingly, the two topologies coincide.

Lemma 5.3. *The relative topology on X is weaker than the quotient topology on X induced by r.*

Proof. A basic open set in the relative topology is of the form $\uparrow a \cap X$ for some compact $a \in D$. Let $y \in r^{-1}[\uparrow a \cap X]$. Clearly, $ry \in \uparrow a \cap X$. By continuity of r in D there exists a compact $b \sqsubseteq y$ such that $a \sqsubseteq rb$. Thus, $y \in \uparrow b \cap D^{\mathrm{R}} \subseteq r^{-1}[\uparrow a \cap X]$. So y is in the interior of $r^{-1}[\uparrow a \cap X]$. □

Theorem 5.4. *Let D be a domain and r a retraction on D. Choose a subset X of maximal fixed points and let $D^{\mathrm{R}} = r^{-1}[X]$. Let the topology on X be the quotient topology induced by r. Then $(D, X, D^{\mathrm{R}}, r, \iota) \in \mathbf{DR}$.*

Proof. The composition $r\iota\colon X \to X$ is the identity, since the elements of X are fixed under r. Since the quotient and relative topologies coincide on X both ι and r are continuous. Hence, the retract property is satisfied.

Assume that $d \in D^{\mathrm{R}}$ and that $d \sqsubseteq d'$. By monotonicity of r, $rd \sqsubseteq rd'$. The element rd' is a fixed point of r, but rd is a maximal fixed point, hence $rd' = rd$ and $d' \in D^{\mathrm{R}} = r^{-1}[X]$. Thus, the representation is upwards closed. □

For any $a \in D_c$, $\iota^{-1}[\uparrow a]$ is open, so $\mathcal{B} = \{\iota^{-1}[\uparrow a] : a \in D_c\} = \{\uparrow a \cap X : a \in D_c\}$ is a base for the topology on X.

The following result shows that it is sufficient to consider domain representations obtained in the above manner.

Proposition 5.5. *Let (D, Y, D', ρ, η) be a dense upwards-closed retract domain representation. Then there exists a retraction r on D such that the domain representation constructed from D and r, as in Theorem 5.4, represents a space homeomorphic to Y.*

Proof. Assume that $d \in D'$ and $\eta\rho d \sqsubseteq \eta\rho d'$. By upwards-closed,

$$\rho d' = \rho\eta\rho d' = \rho\eta\rho d = \rho d\,,$$

and hence $\eta\rho d = \eta\rho d'$. That is, all elements in the image $\eta\rho$ are maximal (in the image of $\eta\rho$, not in D').

The composition $\eta\rho$ can be extended to a continuous function r on D since it is a continuous function from a dense subset of D into an injective space.

Let $X = \eta\rho[D']$. By the above, X is a subset of the maximal fixed points of $r = \eta\rho$. Let $D^{\mathrm{R}} = r^{-1}[X]$. By Theorem 5.4, $(D, X, D^{\mathrm{R}}, r, \iota)$ is an upwards-closed retract domain representation. Since $X = \eta[Y]$, X and Y are homeomorphic. □

Since each T_3 space X has a dense upwards-closed retract domain representation we have that there exists a domain representations of a space homeomorphic to X constructed from a retraction on a domain.

The above result also holds for all countably based coherent upwards-closed retract domain representations (even where D^{R} is not dense in D) by Proposition 3.5.

Definition 5.6. A domain representation $(D, X, D^{\mathrm{R}}, \rho)$ has the *closed image property* if $\rho[\uparrow a \cap D^{\mathrm{R}}]$ is closed for all $a \in D_c$.

The closed image property implies that $\rho[\uparrow d \cap D^{\mathrm{R}}]$ is closed for any $d \in D$.

Proposition 5.7. *A retract domain representation $(D, X, D^{\mathrm{R}}, \rho, \eta) \in \mathbf{DR}$ has the closed image property.*

Proof. Let $x \in X$ belong to the complement of $\rho[\uparrow a \cap D^{\mathrm{R}}]$, that is $\rho^{-1}[x] \cap \uparrow a = \emptyset$. For each $\bar{x} \in \rho^{-1}[x]$ there exists a $b_{\bar{x}} \in D_c$ such that a and $b_{\bar{x}} \sqsubseteq \bar{x}$ are inconsistent. The set

$$U = \bigcup_{\bar{x} \in \rho^{-1}[x]} \uparrow b_{\bar{x}}$$

is open. Clearly, $\eta^{-1}[U]$ and $\rho[\uparrow a \cap D^{\mathrm{R}}]$ are disjoint and x is in the open set $\eta^{-1}[U]$. Hence, the complement of $\rho[\uparrow a]$ is open, i.e., $\rho[\uparrow a]$ is closed. □

Let $(D, X, D^{\mathrm{R}}, \rho, \eta)$ be a retract domain representation. Clearly, $\eta^{-1}[\uparrow a] \subseteq \rho[\uparrow a \cap D^{\mathrm{R}}]$. However, the above result does not entail that $\rho[\uparrow a \cap D^{\mathrm{R}}]$ is the closure of $\eta^{-1}[\uparrow a]$.

5.2 Effective Domain Representations

We use the above approach of constructing domain representations from retractions to define a notion of effective domain representations.

Definition 5.8. A domain representation $(D, X, D^{\mathrm{R}}, r, \iota)$ constructed from a retraction r on D is *effective* if D is an effective domain and r is effective.

Theorem 5.9. *Let $(D, X, D^{\mathrm{R}}, r, \iota)$ be an effective domain representation. Then X is metrizable.*

Proof. The domain D is effective, so D is countably based. Since X is a retract of a countably based space, X is countably based, in fact, the base $\mathcal{B}$ is countable. The space X is T_3 by Theorem 2.3. The result now follows by Urysohn's metrization theorem. □

It also follows that the space X is normal and hence T_4.

An effective domain representation gives rise to two numberings, one of the space X, and one of a base for the topology on the space X.

For the rest of the section let $(D, X, D^{\mathrm{R}}, r, \iota)$ be an effective domain representation, where (D, α) is the effective domain. We can without loss of generality assume that α is total, i.e., $\operatorname{dom}\alpha = \omega$. There exists a canonical total numbering $\bar{\alpha}$ of the constructive subdomain D_{k} consisting of all α-computable elements of D. If $S = \alpha[W_n]$ is a consistent set, then $\bar{\alpha}n$ is defined to be the supremum of S; otherwise, $\bar{\alpha}n$ is the supremum of a consistent finite subset of S. For the detailed construction and proof, see [16, Theorem 4.4]. Let $\hat{r}$ be the recursive function tracking r with respect to $\bar{\alpha}$.

An element $x \in X$ is *computable* if there exists an $\bar{\alpha}$-index n such that $x = r\bar{\alpha}n$. Let X_{k} be the set of all computable elements of X. Define the numbering ξ of X_{k} to be the numbering $\bar{\alpha}\hat{r}$ restricted to the indices that correspond to elements in D^{R}.

In general there exists no bound on the complexity of determining if an index belongs to D^{R} since X is an arbitrary subset of maximal fixed points.

We will now look at the problem of determining if two ξ-indices represent the same element of X_{k}. Recall that $\equiv_{\bar{\alpha}}$ is Π^0_2. Compare the work by Spreen [14].

Proposition 5.10. *The relation $\equiv_\xi$ is Π^0_2 relative to* $\operatorname{dom}\xi$.

Proof. Assume that $m, n \in \operatorname{dom}\xi$. We have

$$m \equiv_\xi n \iff \xi m = \xi n \iff \bar{\alpha}\hat{r}m = \bar{\alpha}\hat{r}n \iff \hat{r}m \equiv_{\bar{\alpha}} \hat{r}n\,.$$

The result follows since $\hat{r}$ is recursive and $\equiv_{\bar{\alpha}}$ is Π^0_2. □

The following example shows that the reals have an effective representation constructed from a domain and a retraction such that $\operatorname{dom}\xi$ is Π^0_2 and equality is co-r.e. relative to $\operatorname{dom}\xi$.

Example 5.11. Let D be the interval domain with rational intervals as compact elements. Let ν be a standard numbering of the rational numbers. We note that subtraction and comparisons are computable on rational numbers with respect to the numbering ν. Let α be defined by $\alpha(\langle m, n\rangle) = [\nu m, \nu n]$, where $\operatorname{dom}\alpha = \{\langle m, n\rangle : \nu m \leq \nu n\}$. Clearly, (D, α) is an effective domain.

Define a retraction r on D_{c} by

$$r([a, b]) = \bigsqcup\{[c, d] : c < a \leq b < d\}\,,$$

and extend r to a continuous function on D. Clearly, r is an effective function. Let X be all maximal fixed points of r.

A d in D belongs to D^{R} if, and only if, for all k there exists $[a, b] \in D_{\mathrm{c}}$ such that $[a, b] \sqsubseteq d$ and $b - a \leq 2^{-k}$, so D^{R} is Π^0_2.

There exists a recursive function f that takes an $\bar{\alpha}$-index n of an element $x \in D^{\mathrm{R}}$ and a natural number k and returns an α-index of a compact element $[a, b]$ such that $\alpha f(n, k) \sqsubseteq \bar{\alpha} n$ and $b - a \leq 2^{-k}$. The compact approximations of $\bar{\alpha}$ can be enumerated uniformly in the index n. So to compute $f(n, k)$ one enumerates the compact elements until a suitable approximation is found.

Now, to decide if two ξ-indices m and n are equivalent, it is sufficient to decide if $\mathrm{Cons}(\alpha f(m, k), \alpha f(n, k))$ holds for all $k \in \omega$. Thus, $\equiv_\xi$ is co-r.e.

The base $\mathcal{B} = \{\uparrow a \cap X : a \in D_{\mathrm{c}}\}$ has a numbering β defined by

$$\beta n = \uparrow(\alpha n) \cap X\,.$$

Define a recursive relation $\prec$ on $\operatorname{dom}\beta$ by

$$m \prec n \iff \alpha m \sqsupseteq \alpha n\,.$$

Assume that $m \prec n$. Then $\alpha m \sqsupseteq \alpha n$ so $\uparrow(\alpha m) \subseteq \uparrow(\alpha n)$, and hence

$$\beta m = \uparrow(\alpha m) \cap X \subseteq \uparrow(\alpha n) \cap X = \beta n\,.$$

Thus, $m \prec n$ implies $\beta m \subseteq \beta n$.

Lemma 5.12. *There exists a recursive function f taking two β-indices of intersecting basic open sets and returning a β-index for the non-empty intersection.*

Proof. For any $x \in X$, $x \in \beta m \cap \beta n$ if, and only if, $\alpha m \sqsubseteq x$ and $\alpha n \sqsubseteq x$. Hence, $\alpha m \sqcup \alpha n \sqsubseteq x$. Thus, if $\beta m \cap \beta n \neq \emptyset$ then

$$\beta m \cap \beta n = \uparrow(\alpha m \sqcup \alpha n) \cap X = \beta\hat{\sqcup}(m, n)\,,$$

where $\hat{\sqcup}$ is the recursive function tracking the computable binary supremum operation on D_{c}.

Define f by

$$f(m, n) = \begin{cases} \hat{\sqcup}(m, n), & \text{if } \mathrm{Cons}(\alpha m, \alpha n); \\ \uparrow, & \text{otherwise}\,. \end{cases}$$

□

In the terminology of Spreen [15] the base $\mathcal{B}$ is a *strong basis* and this holds effectively with respect to $\prec$.

In fact, the function f of the above lemma computes an index for the intersection as soon as αm and αn are consistent. If there exists a known β-index of $\emptyset$ then intersection is computable.

Lemma 5.13. *Let k be a β-index for $\emptyset$. Then intersection is computable with respect to β.*

Proof. Define the tracking function f by

$$f(m,n) = \begin{cases} \hat{\sqcup}(m,n), & \text{if } \mathrm{Cons}(\alpha m, \alpha n); \\ k, & \text{otherwise}. \end{cases}$$

□

In general, $\equiv_\beta$ is not decidable, in particular, there exists no general way of determining if $\beta n = \emptyset$. However, for certain domain representations $\equiv_\beta$ might be decidable. This is the case for the interval domain representation of the reals discussed in Example 5.11. For the interval domain we have that a β-index n represents the empty set if, and only if, $\alpha n = [a,a]$ for some a.

Lemma 5.14. *An index of an r.e. set V can be computed, uniformly in n, such that for all $i \in \mathrm{dom}\,\xi$, $i \in V \iff \xi i \in \beta n$.*

Proof. For any index i of a computable $x \in X_\mathrm{k}$ we have

$$\begin{aligned} \xi i \in \beta n &\iff \xi i \in \uparrow(\alpha n) \cap X \\ &\iff \xi i \in \uparrow(\alpha n) \\ &\iff \alpha n \sqsubseteq \xi i \\ &\iff \alpha n \sqsubseteq r\bar{\alpha} i \\ &\iff \alpha n \sqsubseteq \bar{\alpha}\hat{r} i\,. \end{aligned}$$

The compact approximations of $\bar{\alpha}\hat{r}i$ can therefore be enumerated. The set V is obtained by enumerating all i such that αn is a compact approximation of $\bar{\alpha}\hat{r}i$. □

The above lemma states in the terminology of Spreen [15] that the numbering ξ is *computable*. Moreover, since $\mathcal{B}$ is an effectively strong basis with respect to β we have that X is an *effective* T_0 space.

Let us now look at effectivity of topological convergence with respect to our numberings.

Definition 5.15. If the r.e. set W_n enumerates β-indices of a neighbourhood base for a point $x \in X_\mathrm{k}$ then we say that n is a *neighbourhood index* of x.

Lemma 5.16. *Let n be a ξ-index of a point $x \in X_\mathrm{k}$. Then $\hat{r}n$ is a neighbourhood index of x.*

Proof. Assume that U is an open set containing x. Then there exists a basic open set $B \in \mathcal{B}$ such that $x \in B \subseteq U$. The set B is of the form $\uparrow a \cap X$ for some $a \sqsubseteq x$. By definition of ξ, $\hat{r}n$ is an $\bar{\alpha}$-index of x. Hence there exists an α-index $b \in W_{\hat{r}n}$ such that $a \sqsubseteq \alpha b \sqsubseteq x$. Clearly, $x \in \beta b \subseteq \uparrow a \cap X \subseteq U$. □

We are interested in when a converse of the above result exists, i.e., when ξ-indices can be computed from neighbourhood indices.

Definition 5.17. The numberings ξ and β *allows effective limit passing* if a ξ-index effectively and uniformly can be computed from a neighbourhood index.

This is closely related to *acceptable* numberings as defined by Spreen [15] and to *admissible* numberings as defined by Kreitz and Weihrauch [11, 20]. The main difference is that while the numbering β is fixed in their settings, β is derived along with the numbering ξ in our case. That is, they investigate different numberings of the elements with respect to some numbering β of a base.

Let n be a neighbourhood index of $x \in X_{\mathrm{k}}$. For all $m \in W_n$ we have $x \in \beta m = \uparrow(\alpha m) \cap X$. If $m, m' \in W_n$ then $x \in \beta m \cap \beta m'$, and hence $\mathrm{Cons}(\alpha m, \alpha m')$. Thus, the set $\{\alpha m : m \in W_n\}$ is directed and bounded by x, so the supremum

$$d = \bigsqcup_{m \in W_n} \alpha m$$

exists and $d \sqsubseteq x$. Note that n is an $\bar{\alpha}$-index of d.

Lemma 5.18. *Let n be a neighbourhood index of $x \in X$, and let $d = \bar{\alpha}n$. If $d \in D^{\mathrm{R}}$, or, equivalently, if $n \in \operatorname{dom} \xi$, then $\hat{r}n$ is a ξ-index of x.*

Proof. Since the representation is upwards-closed and x is a fixed point we have $rd = rx = x$. □

Proposition 5.19. *Let an effective domain representation be derived from a domain D and a retraction r on D. If, for any $d \in D$, $\uparrow d \cap X = \{x\} \implies d \in D^{\mathrm{R}}$, then the derived ξ and β allows effective limit passing.*

Proof. Let n be a neighbourhood index of $x \in X$, and let $d = \bar{\alpha}n$. Since n is a neighbourhood index of x it follows that $\uparrow d \cap X = \{x\}$. By the condition and Lemma 5.18 we have that $\hat{r}$ uniformly computes effective limit passing.

Even if the condition in the proposition above is not satisfied it is often possible to find some effective function computing effective limit passing. However, for certain cases this is impossible.

Example 5.20. Let D be the domain depicted in Figure 1. The retraction r is the identity and the set X is the set of all maximal points. The set X will get the discrete topology. Any $\bar{\alpha}$-index n of a_ω is a neighbourhood index of x. But in order to check that n is a neighbourhood index of x it is necessary to check that W_n contains indices of infinitely many a_i. This is clearly not effective.

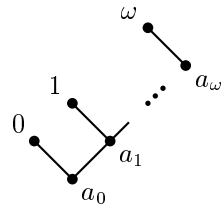


Fig. 1. Domain representation without effective limit passing.

6 Conclusions

In the search for a general effectivity theory to be given to T_3 spaces one problem that arises is the choice of base. It becomes visible in the competing bases that we get from the functor R of Section 4. Studying the effectivity in that setting would probably be ad hoc and confusing.

We suggest a different way of building domain representations which has the benefit that it simultaneously gives numberings of both the base and the elements. These two numberings usually work well together, although some limitations exist. For example, the numberings do not always allow effective limit passing.

Many open questions remain. We know that the effectively represented spaces must be T_3, in fact they are metrizable. The represented spaces will be effectively Hausdorff. However, I conjecture that some spaces will fail to be effectively regular, that is, given a point in a Lacombe set, it is in general not possible to find an open set containing the point and whose closure is a subset of the Lacombe set. Is there a characterisation of the representations that will be effectively regular? Which representation are effectively metrizable?

References

1. J. Blanck. Domain representability of metric spaces. *Annals of Pure and Applied Logic*, 83:225–247, 1997.
2. J. Blanck. Domain representations of topological spaces. *Theoretical Computer Science*, 247:229–255, 2000.
3. P. di Gianantonio. Real number computability and domain theory. *Information and Computation*, 127:11–25, 1996.
4. A. Edalat. Domain theory and integration. *Theoretical Computer Science*, 151:163–193, 1995.
5. A. Edalat. Dynamical systems, measures, and fractals via domain theory. *Information and Computation*, 120:32–48, 1995.
6. A. Edalat. Power domains and iterated function systems. *Information and Computation*, 124:182–197, 1996.
7. A. Edalat. Domains for computation in mathematics, physics and exact real arithmetic. *Bulletin of Symbolic Logic*, 3(4):401–452, 1997.

8. A. Edalat and R. Heckmann. A computational model for metric spaces. *Theoretical Computer Science*, 193:53–73, 1998.
9. Y. L. Ershov. Model c of partial continuous functionals. In R. O. Gandy and J. M. E. Hyland, editors, *Logic Colloquium 76*, volume 87 of *Studies in Logic and Foundations in Mathematics*, pages 455–467. North-Holland, 1977.
10. M. H. Escardó. Injective spaces via the filter monad. *Topology Proceedings*, 22(2):97–110, 1997.
11. C. Kreitz and K. Weihrauch. Theory of representations. *Theoretical Computer Science*, 38:35–53, 1985.
12. D. Normann. The continuous functionals of finite types over the reals. Preprint, Department of Mathematics, University of Oslo, 1998.
13. M. Schröder. Effective metrization of regular spaces. In K.-I. Ko et al., editors, *Computability and Complexity in Analysis*, volume 235 of *Informatik-Berichte*, pages 63–80. FernUniversität Hagen, August 1998. CCA Workshop, Brno, Czech Republic, August, 1998.
14. D. Spreen. On some decision problems in programming. *Information and Computation*, 122(1):120–139, 1995. (Corrigendum: Inform. and Comp. 148, 241–244, 1999).
15. D. Spreen. On effective topological spaces. *The Journal of Symbolic Logic*, 63(1):185–221, 1998.
16. V. Stoltenberg-Hansen, I. Lindström, and E. R. Griffor. *Mathematical Theory of Domains*. Cambridge University Press, 1994.
17. V. Stoltenberg-Hansen and J. V. Tucker. Complete local rings as domains. *Journal of Symbolic Logic*, 53:603–624, 1988.
18. V. Stoltenberg-Hansen and J. V. Tucker. Effective algebra. In S. Abramsky et al., editors, *Handbook of Logic in Computer Science*, volume IV, pages 357–526. Oxford University Press, 1995.
19. G. A. Waagbø. *Domains-with-totality semantics for Intuitionistic Type Theory*. PhD thesis, University of Oslo, 1997.
20. K. Weihrauch. *An Introduction to Computable Analysis*. Springer, 2000.
21. K. Weihrauch and U. Schreiber. Embedding metric spaces into cpo's. *Theoretical Computer Science*, 16:5–24, 1981.

The Degree of Unsolvability of a Real Number

Anthony J. Dunlop[1] and Marian Boykan Pour-El*[2]

[1] College of St. Benedict, St. Joseph, MN 56374, USA
[2] School of Mathematics, University of Minnesota, Minneapolis, MN 55455, USA

Abstract. Each real number x can be assigned a degree of unsolvability by using, for example, the degree of unsolvability of its binary or decimal expansion, or of its Dedekind cut, or of some other representation of x. We show that the degree of unsolvability assigned to x in any such way is the same regardless of the representation used. This gives to each real number a unique degree of unsolvability. If x is of computably enumerable degree, there is a computable sequence of rationals which converges to x with a modulus of convergence having the same degree of unsolvability as x itself. In contrast, if x is computable relative to the halting set but is *not* of computably enumerable degree, this is not true. Specifically, if $\{r_n\}$ is any computable sequence of rationals converging to such a real number x, the modulus of convergence of $\{r_n\}$ must have degree of unsolvability *strictly higher* than that of x. Thus there is an inherent gap between the degree of unsolvability of such an x and the degree of unsolvability of the modulus of convergence of an approximating computable sequence of rationals; this gap is bridged (in the sense of the "join operator" of degree theory) by a set of natural numbers which measures the twists and turns of the computable sequence $\{r_n\}$.

1 Introduction

In classical computability theory, degrees of unsolvability arise in the context of sets of nonnegative integers. This paper introduces the study of the degrees of unsolvability of real numbers.

The notion of a computable real number is well-understood [2], [3]. A real number x is computable if there is a computable sequence of rational numbers which converges to x with a computable modulus of convergence. All other definitions, involving Dedekind cuts, nested intervals, etc., are known to be equivalent.

In section 2 of this paper, we relativize this idea to arrive at a precise characterization of the degree of difficulty of computing an arbitrary real number. We will see that any real number can be associated, in a natural way, with a degree of unsolvability. This association is then independent of the particular way (Dedekind cut, Cauchy sequence, etc.) in which we represent the number. So every real number is given a *unique* degree of unsolvability.

* Please address all correspondence to M. B. Pour-El. The authors are grateful to the referees for simplifications of two of the proofs.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 16–29, 2001.

When it comes to real numbers which are computable relative to the halting problem for Turing machines, however, things are less simple. An obvious definition of a real number computable relative to the halting problem is this: x is computable relative to the halting problem if there is a sequence of rational numbers, computable relative to the halting problem, which converges to x with a modulus of convergence which is also computable relative to the halting problem. It is easy to see that the modulus of convergence can be subsumed into the sequence itself, so that x is computable relative to the halting problem if and only if there is a sequence of rationals, computable relative to the halting problem, which converges to x with a computable modulus. On the other hand, Ho has shown [1] that a real number x is computable relative to the halting problem if and only if there is a computable sequence of rational numbers which converges to x, and that there is always a modulus of convergence for this sequence which is computable relative to the halting problem.

Thus there are three things to consider. We have the real number x, a sequence of rational numbers converging to x, and a modulus of convergence for that sequence. Each of these has a degree of unsolvability. One might expect that the degree of the sequence combines with the degree of the modulus to give *exactly* the degree of x. We show, in section 3 of the paper (theorem 4), that this does not happen in general. In particular, we show the existence of a real number x such that, for any *computable* sequence of rationals converging to x, the modulus of convergence of that sequence must have degree *strictly higher* than the degree of x itself. Thus the sequence and the modulus combine to "overshoot" the degree of unsolvability of x. This happens whenever the degree of unsolvability of x is below the degree of the halting problem, but is not the degree of any computably enumerable set. By contrast, we show (theorem 3) that x is computable relative to a computably enumerable set A if and only if there is a computable sequence of rational numbers converging to x with a modulus of convergence which is computable in A. In general, though, we must expect an inherent gap between the degree of unsolvability of x and the degree of the modulus of convergence of a computable sequence of rationals converging to x. We conclude by exhibiting a set of natural numbers which serves to "bridge" this gap, in the sense of the degree-theoretic join operator $\oplus$. This set measures the extent to which the approximating computable sequence of rationals "jumps back" away from x during the convergence process.

We now introduce some notation, which is standard in the literature of computability theory. $\emptyset'$ denotes the halting set for Turing machines, and $\mathbf{0}'$ represents the degree of unsolvability for that set. For any $A \subseteq \mathbb{N}$, $\mathbf{a}$ denotes the degree of unsolvability of A.

One remark: Since the integer part of a real number x is finite, the degree of unsolvability of x is completely determined by its decimal part. For this reason, we consider only reals in the unit interval $[0, 1]$, with no loss of generality.

2 The Degree of Unsolvability of a Real Number

The equivalence of all standard definitions of a computable real number relativizes directly, to give a stable definition of an A-computable real number, for any $A \subseteq \mathbb{N}$.

Definition 1. *A sequence $\{r_n\}$ of rational numbers is A-**computable** if there exist functions s, a, $b \leq_T A$ such that $b(n) \neq 0$ for all n, and such that $r_n = (-1)^{s(n)} \frac{a(n)}{b(n)}$.*

Theorem 1. *Let $A \subseteq \mathbb{N}$, and let $x \in [0,1]$. Then the following are equivalent:*

1) (Cauchy function.) There exists an A-computable sequence $\{q_n\}$ of rational numbers in [0,1] such that, for all $n \in \mathbb{N}$, we have $|q_n - x| < 2^{-n}$.

2) (Dedekind cut.) There exists a function $D : \mathbb{Q} \cap [0,1] \to \{0,1\}$ such that $D \leq_T A$ and such that, for all $q \in \mathbb{Q} \cap [0,1]$, we have

$$D(q) = \begin{cases} 1 & \text{if } q < x \\ 0 & \text{if } q > x. \end{cases}$$

3) (p-ary expansion.) For every $p \in \{2, 3, \cdots\}$, there exists a function $E_p : \mathbb{N} \to \{0, 1, \cdots, p-1\}$ such that $E_p \leq_T A$ and such that $x = \sum_{k=1}^{\infty} E_p(k) \cdot p^{-k}$.

4) (nested interval.) There exist functions l, $r : \mathbb{N} \to \mathbb{Q} \cap [0,1]$ such that l, $r \leq_T A$, such that $l(1) \leq l(2) \leq \cdots \leq l(n) \leq \cdots \leq r(n) \leq \cdots \leq r(2) \leq r(1)$, and such that $\{x\} = \bigcap_{n=1}^{\infty} (l(n), r(n))$.

The proof is a relativization of the well-known analogous result about computable real numbers (see, e.g., [3] or [6]), and is omitted.

We may now do better than relative computability. Namely, we may ascribe to each real number x a degree of unsolvability, in a natural way. Note that, if x is rational, then x is computable, and its degree of unsolvability is $\mathbf{0}$. Rational numbers here are analogous to finite sets in classical computability theory. If x is irrational, then its Dedekind cut representation is unique and, for each $p \in \{2, 3, \cdots\}$, its p-ary expansion is unique. By theorem 1, the functions defining these representations all belong to the same degree of unsolvability. However, for each $x \in [0,1]$, there are uncountably many Cauchy sequence representations, and there are uncountably many nested interval representations. The next theorem tells us that both the set of Cauchy sequence representations for x and the set of nested interval representations for x have members of least Turing degree, and that this degree coincides with the degree of those representations of x which are unique. It is this degree which gives the natural degree of unsolvability for x.

Theorem 2. *Let $x \in [0,1]$. Then there is a unique degree of unsolvability* $\mathbf{a}$ *such that, for any $A \in \mathbf{a}$, x is A-computable, and for any $A \in \mathbf{a}$ and for any $B \subseteq \mathbb{N}$, if x is B-computable, then $A \leq_T B$.*

Informally, then, $\mathbf{a}$ is the least degree (in their usual ordering) which is required to obtain a computation of x.

Proof. Let $\tilde{A} \subseteq \mathbb{N}$ be such that $x = \sum_{k=1}^{\infty} \chi_{\tilde{A}}(k) \cdot 2^{-k}$, and let $\mathbf{a}=\deg(\tilde{A})$. Then for any $A \in \mathbf{a}$, we have $\tilde{A} \leq_T A$, so that by (3) of theorem 1, x is A-computable.

Now let $B \subseteq \mathbb{N}$ such that x is B-computable, and let $A \in \mathbf{a}$. By (3) of theorem 1, we have $\tilde{A} \leq_T B$. Since $A \equiv_T \tilde{A}$, it follows that $A \leq_T B$.

To establish the uniqueness of $\mathbf{a}$, suppose $\mathbf{b}$ is any degree of unsolvability which satisfies the two above conditions. Then let $A \in \mathbf{a}$ and $B \in \mathbf{b}$. By the second condition of the theorem, since $A \in \mathbf{a}$ and x is B-computable, we have $A \leq_T B$; since $B \in \mathbf{b}$ and x is A-computable, we have $B \leq_T A$. Hence $A \equiv_T B$, so $\mathbf{a}=\mathbf{b}$. This proves theorem 2. □

Definition 2. *Let $x \in [0,1]$. Then the* ***degree of unsolvability*** *of x is the unique degree of unsolvability satisfying the two conditions in theorem 2.*

3 Approximating a $\emptyset'$-Computable Real with a Computable Sequence of Rationals

Definition 3. *Suppose $\{x_n\}$ is a sequence of real numbers which converges to some limit x. Then a function $m : \mathbb{N} \to \mathbb{N}$ is a* ***modulus of convergence*** *for $\{x_n\}$ if $|x_n - x| < 2^{-p}$ whenever $n \geq m(p)$.*

Chun-Kuen Ho has shown in [1] that a real number x is $\emptyset'$-computable if and only if there is a computable sequence $\{r_n\}$ of rational numbers which converges to x. Ho also showed that any convergent computable sequence of rational numbers has a modulus of convergence which is computable in $\emptyset'$. The degree of this modulus can be pushed downward if x is A-recursive, where A is a c.e. set with $A <_T \emptyset'$. Later we will see that an analogous result does not hold if x is a real number of degree $\mathbf{a}$, where $\mathbf{a} < \mathbf{0}'$ but $\mathbf{a}$ is not a computably enumerable degree.

Theorem 3. *Let $x \in [0,1]$, and let $A \subseteq \mathbb{N}$ be a computably enumerable set. Then x is A-computable if and only if there is a computable sequence $\{r_s\}$ of rational numbers, and an A-computable function $m : \mathbb{N} \to \mathbb{N}$ such that, for all $s \in \mathbb{N}$ and all $p \in \mathbb{N}$, if $s \geq m(p)$, then $|r_s - x| < 2^{-p}$.*

In particular, then, if x is a real number of c.e. degree, the degree of unsolvability of x can be exactly transferred to the modulus of convergence of a computable sequence of rational numbers which converges to x. We will see that this fails if x is a $\emptyset'$-computable real number which does *not* have c.e. degree.

Proof. $\Leftarrow$: Suppose $\{r_s\}$ is a computable sequence of rationals, and $m \leq_T A$ has the property that $s \geq m(p)$ implies $|r_s - x| < 2^{-p}$, for all s and p. Then let $r_s^* = r_{m(s)}$. Then $\{r_s^*\}$ is an A-computable sequence of rationals with the property that, for all $s \in \mathbb{N}$, we have $|r_s^* - x| = |r_{m(s)} - x| < 2^{-s}$, so x is A-computable, since it satisfies condition (1) of theorem 1.
$\Rightarrow$: Suppose x is A-computable. If x is computable, then there is nothing to prove. So suppose x is not computable. This implies that A is a noncomputable c.e. set. Let $a : \mathbb{N} \to \mathbb{N}$ be a one-to-one computable function which enumerates A, and let $A_s = \{a(0), \cdots, a(s)\}$.

Since x is A-computable, it follows that there is a $B \subset \mathbb{N}$ such that $B \leq_T A$ and $x = \sum_{k \in B} 2^{-k}$. Let $e \in \mathbb{N}$ such that $\chi_B = \Phi_e^A$, and define the set B_s by

$$\chi_{B_s}(k) = \begin{cases} \Phi_{e,s}^{A_s}(k) & \text{if } \Phi_{e,s}^{A_s}(k) \downarrow \\ 0 & \text{otherwise.} \end{cases}$$

(Here Φ_e denotes the eth oracle Turing machine, and $\Phi_{e,s}$ denotes the machine running for s steps.) Then let $r_s = \sum_{k \in B_s} 2^{-k}$. Clearly $\{r_s\}$ is a computable sequence of rational numbers converging to x. We define a modulus of convergence m as follows:

On input $p \in \mathbb{N}$, use the set A as an oracle to determine the largest $N \in \mathbb{N}$ for which $A(N)$ is queried in the computations $\Phi_e^A(k)$, $0 \leq k \leq p+1$. Then define $m(p)$ to be the smallest $n \in \mathbb{N}$ for which the following hold:
- $\Phi_{e,n}^{A_n}(k) \downarrow$ for $0 \leq k \leq p+1$;
- $A_n \upharpoonright N = A \upharpoonright N$, where $A \upharpoonright N$ refers to the (characteristic function of the) set A, restricted to the first N arguments.

The only noneffective parts of the computation of $m(p)$ involve queries to the oracle A, so $m \leq_T A$. Also, if $s \geq m(p)$, we know that $\chi_{B_s}(k) = \chi_B(k)$ for $0 \leq k \leq p+1$, since $\chi_{B_s}(k) = \Phi_{e,s}^{A_s}(k)$ and, since $s \geq m(p)$, $A_s = A$ for all elements actually used in the computation of $\Phi_{e,s}^{A_s}(k)$, whence $\Phi_{e,s}^{A_s}(k) = \Phi_e^A(k) = \chi_B(k)$. It follows that

$$\begin{aligned} |r_s - x| &= |\sum_{k=0}^{\infty} \chi_{B_s}(k) 2^{-k} - \sum_{k=0}^{\infty} \chi_B(k) 2^{-k}| \\ &= |\sum_{k=p+2}^{\infty} (\chi_{B_s}(k) - \chi_B(k)) 2^{-k}| \\ &\leq \sum_{k=p+2}^{\infty} 2^{-k} = 2^{-(p+1)} < 2^{-p}, \end{aligned}$$

so m is a modulus for $r_s \to x$. This proves theorem 3. □

The preceding theorem does not generalize to arbitrary $A \leq_T \emptyset'$. If $\mathbf{a}$ is the degree of unsolvability of $x \in [0,1]$, and if $\mathbf{a} < \mathbf{0}'$ but $\mathbf{a}$ is not c.e., then for any computable sequence $\{r_n\}$ of rational numbers which converges to x, the degree of unsolvability of any modulus of convergence of $\{r_n\}$ must necessarily

be strictly greater than **a**. Hence for x computable in $\emptyset'$ but not of c.e. degree, there is an inherent gap between $\deg(x)$ and the degree of unsolvability of a modulus of convergence for a computable sequence of rationals converging to x.

Definition 4. *Let $x \in [0,1]$ and let* **a** *be a degree of unsolvability. Then x is an* **a-*computable real number*** *if there is some $A \in$ **a** for which x is A-computable. Equivalently, x is* **a***-computable if deg(x)$\leq$**a** in the usual ordering of the degrees of unsolvability.*

Definition 5. *Let $f : \mathbb{N} \to \mathbb{N}$ and let* **a** *be a degree of unsolvability. Then f is* **a-computable** *if deg(f)$\leq$**a**.*

Theorem 4. *Let $x \in [0,1]$. Suppose the degree of unsolvability of x is* **a***, where* **a**$<$**0**$'$ *but* **a** *is not computably enumerable. Then there is no computable sequence $\{r_n\}$ of rational numbers such that $r_n \to x$ with an* **a***-computable modulus of convergence. Furthermore, for any computable sequence $\{r_n\}$ of rational numbers converging to x, any modulus of convergence for $\{r_n\}$ must have degree strictly higher than* **a**.

Proof. The proof follows from two lemmas.

Lemma 1. *Suppose $x \in [0,1]$ is a noncomputable,* **0**$'$*-computable real number. Let $\{r_n\}$ be a computable sequence of rational numbers which converges to x. Then there is a modulus of convergence $m : \mathbb{N} \to \mathbb{N}$ for $r_n \to x$ such that, if $\tilde{m}$ is any other modulus of convergence for $r_n \to x$, we have $m \leq_T \tilde{m}$. Furthermore, m has computably enumerable degree of unsolvability.*

Proof. We define m as follows:

$$m(p) = \mu n((\forall k \geq n)|r_n - r_k| < 2^{-(p+1)}).$$

To see that m is a modulus of convergence for $r_n \to x$, let $p \in \mathbb{N}$ be given, let $n \geq m(p)$, let $\epsilon > 0$, and let $N \in \mathbb{N}$ be large enough that $N \geq m(p)$ and $|r_N - x| < \epsilon$. Then

$$\begin{aligned} |r_n - x| &\leq |r_n - r_{m(p)}| + |r_{m(p)} - x| \\ &< 2^{-(p+1)} + |r_{m(p)} - x| \text{ (since } n \geq m(p)) \\ &\leq 2^{-(p+1)} + |r_{m(p)} - r_N| + |r_N - x| \\ &< 2^{-(p+1)} + 2^{-(p+1)} + \epsilon \text{ (by choice of } N) \\ &= 2^{-p} + \epsilon. \end{aligned}$$

Since $\epsilon > 0$ is arbitrary, it follows that $|r_n - x| \leq 2^{-p}$; but since x is noncomputable, and hence irrational, it follows that $|r_n - x| < 2^{-p}$ whenever $n \geq m(p)$, so that m is indeed a modulus of convergence for $r_n \to x$.

Now the following claims prove lemma 1:

Claim 1. If $\tilde{m}$ is any modulus of convergence for $r_n \to x$, then $m \leq_T \tilde{m}$.

Proof. Let $\tilde{m}$ be any modulus of convergence for $r_n \to x$, and let $M(p) = \tilde{m}(p+2)$. Clearly $M \equiv_T \tilde{m}$. We show that $m \leq_T M$, and this will establish claim 1.

To see that $m \leq_T M$, first we note that, if $k \geq M(p)$, we have

$$|r_{M(p)} - r_k| \leq |r_{M(p)} - x| + |x - r_k| < 2^{-(p+2)} + 2^{-(p+2)} = 2^{-(p+1)},$$

so we know that $m(p) \leq M(p)$ for all p. Now we use M as an oracle to determine which of the finitely many numbers $\{0, 1, \cdots, M(p)\}$ is the correct value of $m(p)$.

We use the fact that x is noncomputable, and hence irrational, and hence that either $|r_{M(p)-1} - x| < 2^{-(p+1)}$ or $|r_{M(p)-1} - x| > 2^{-(p+1)}$ is true. Using this fact, we produce two M-effective procedures. Procedure (1) halts if $|r_{M(p)-1} - x| > 2^{-(p+1)}$, and procedure (2) halts if $|r_{M(p)-1} - x| < 2^{-(p+1)}$. Dovetailing these two procedures for successively decremented variables will allow us to determine the correct value of $m(p)$, as described and explained below.

Procedure (1). For $k = M(p) - 1, M(p), M(p) + 1, \cdots$, compute $|r_{M(p)-1} - r_k|$. If we find such a k with $|r_{M(p)-1} - r_k| \geq 2^{-(p+1)}$, then this says that

$$(\exists k \geq M(p) - 1)|r_{M(p)-1} - r_k| \geq 2^{-(p+1)}.$$

By definition of $m(p)$, this says that $m(p) \neq M(p) - 1$. We then halt both procedures, store the information $m(p) \neq M(p) - 1$, and go to the next stage (using $M(p) - 2$ in place of $M(p) - 1$).

Procedure (2). Using a computable pairing $<\cdot, \cdot>$ of natural numbers, look for a pair $<k_0, s>$ such that the following conditions hold:

- $s > p + 1$,
- $k_0 \geq \max\{M(p), M(s)\}$, and
- $|r_{M(p)-1} - r_{k_0}| < 2^{-(p+1)} - 2^{-s}$.

First let us see why such a pair $<k_0, s>$ will exist if $|r_{M(p)-1} - x| < 2^{-(p+1)}$. The fact that $|r_{M(p)-1} - x| < 2^{-(p+1)}$ means that there exists some $s > p + 1$ such that $|r_{M(p)-1} - x| < 2^{-(p+1)} - 2^{-(s-1)}$. Then take the least such s, and let $k_0 = \max\{M(s), M(p)\}$. Then $|r_{k_0} - x| < 2^{-(s+2)} < 2^{-s}$, so for this k_0 and s we have

$$\begin{aligned} |r_{M(p)-1} - r_{k_0}| &\leq |r_{M(p)-1} - x| + |x - r_{k_0}| \\ &< 2^{-(p+1)} - 2^{-(s-1)} + 2^{-s} \\ &= 2^{-(p+1)} - 2^{-s}. \end{aligned}$$

This pair $<k_0, s>$ thus exhibits the three bulleted properties above. Hence this search for $<k_0, s>$ will always terminate if $|r_{M(p)-1} - x| < 2^{-(p+1)}$.

If and when such a pair $<k_0, s>$ is found, do the following: For $k \in \{M(p) - 1, M(p), \cdots, k_0\}$, compute $|r_{M(p)-1} - r_k|$. If one of these finitely many k has $|r_{M(p)-1} - r_k| \geq 2^{-(p+1)}$, then we know (by the definition of $m(p)$) that $m(p) \neq M(p) - 1$. We then halt both procedures, store the information $m(p) \neq M(p) - 1$, and go to the next stage.

Otherwise, that is if we find that $|r_{M(p)-1} - r_k| < 2^{-(p+1)}$ for all $k \in \{M(p) - 1, M(p), M(p)+1, \cdots, k_0\}$, we know that $m(p) < M(p)$, for the following reason. Suppose $k \geq M(p) - 1$. If $k \leq k_0$, we know that $|r_{M(p)-1} - r_k| < 2^{-(p+1)}$ since we have just done the computations. If $k > k_0$, we have

$$\begin{aligned} |r_{M(p)-1} - r_k| &\leq |r_{M(p)-1} - r_{k_0}| + |r_{k_0} - r_{M(s)}| + |r_{M(s)} - r_k| \\ &< 2^{-(p+1)} - 2^{-s} + |r_{k_0} - x| + |x - r_{M(s)}| + |r_{M(s)} - x| + |x - r_k| \\ &< 2^{-(p+1)} - 2^{-s} + 2^{-(s+2)} + 2^{-(s+2)} + 2^{-(s+2)} + 2^{-(s+2)} \\ &= 2^{-(p+1)}, \end{aligned}$$

where the last inequality holds because k_0 was chosen to satisfy $k_0 \geq M(s)$ and $M(s) = \tilde{m}(s+2)$, where $\tilde{m}$ was our arbitrary modulus of convergence for $r_n \to x$. In short, then, we have $|r_{M(p)-1} - r_k| < 2^{-(p+1)}$ whenever $k \geq M(p) - 1$. By definition of m, this means that $m(p) \leq M(p) - 1$. We then halt both procedures, store the information $m(p) \leq M(p) - 1$, and go to the next stage.

After dovetailing procedures (1) and (2), and after one of them halts, we know one of two things: Either $m(p) \neq M(p) - 1$, or else $m(p) \leq M(p) - 1$. In either case, we re-start the process of dovetailing procedures (1) and (2), except this time we compute $|r_{M(p)-2} - r_k|$ for $k \geq M(p) - 2$ instead of $|r_{M(p)-1} - r_k|$ for $k \geq M(p) - 1$. This time, procedure (1) halts if $|r_{M(p)-2} - x| > 2^{-(p+1)}$ and procedure (2) halts if $|r_{M(p)-2} - r_k| < 2^{-(p+1)}$. At the end of this process, we will have determined either that $m(p) \neq M(p) - 2$ or that $m(p) \leq M(p) - 2$.

Continuing this process for $j = 3, 4, \cdots, M(p)$, one of two things will happen: *Case 1:* Some $j = 0, 1, \cdots, M(p) - 1$ is found such that $m(p) \leq M(p) - j$ and $m(p) \neq M(p) - \ell$ for $j < \ell \leq M(p)$. That is, $m(p) \leq M(p) - j$ but $m(p) \neq 0, 1, \cdots, M(p) - (j+1)$. Of course this means that $m(p) = M(p) - j$, so in this case we output $m(p) = M(p) - j$.
Case 2: The process of running procedures (1) and (2) for $j = 0, 1, \cdots, M(p)$ terminates, telling us that $m(p) \leq 0$. But since $m : \mathbb{N} \to \mathbb{N}$, of course this means that $m(p) = 0$. So in this case we output $m(p) = 0$.

We are now done, having determined the correct value of $m(p)$. Since the only noneffective part of the above process is the use of the function M, we have shown that $m \leq_T M$; since $M \equiv_T \tilde{m}$, it follows that $m \leq_T \tilde{m}$. This proves claim 1.

Claim 2. m has computably enumerable degree of unsolvability.

Proof. Recall that $m(p) = \mu n((\forall k \geq n)|r_n - r_k| < 2^{-(p+1)})$. Now let $M = \{(p, n) : m(p) > n\}$. Since $(p, n) \in M$ if and only if $(\forall j \leq n)(\exists k \geq j)(|r_j - r_k| \geq 2^{-(p+1)}$, and since $\{r_n\}$ is a computable sequence of rational numbers, M is a c.e. set. Further, $M \equiv_T m$. To see this, note that to compute $m(p)$ one may generate $(p, 0), (p, 1), \cdots$ until $(p, n) \notin M$; then $m(p) = n$, so $m \leq_T M$. The other reducibility is even more trivial. Hence our function m has c.e. degree, so that claim 2 is proved.

With the proof of claims 1 and 2, lemma 1 is proved. □

Lemma 2. *Let $x \in [0,1]$ be a noncomputable, $\emptyset'$-computable real number. Let $\{r_n\}$ be any computable sequence of rational numbers which converges to x, and let $m : \mathbb{N} \to \mathbb{N}$ be the least-Turing-degree modulus of convergence for $r_n \to x$, as found in lemma 1. Then deg(x) ≤ deg(m).*

Proof. The sequence $\{r_{m(n)}\}$ is an m-computable sequence of rational numbers. For all $n \in \mathbb{N}$, we have $|r_{m(n)} - x| < 2^{-n}$, since m is a modulus of convergence for $r_n \to x$. By the equivalence of the definitions in theorem 1, this says that x is an m–computable real number. By the definition of the degree of unsolvability of a real number as the *least* Turing degree required to compute arbitrarily good approximations of x, this means that $\deg(x) \leq \deg(m)$, so lemma 2 is proved. □

Now we can complete the proof of theorem 4. Suppose, as in the statement of theorem 4, that $x \in [0,1]$ has degree of unsolvability $\mathbf{a}$, where $\mathbf{a} \leq \mathbf{0}'$, but $\mathbf{a}$ is not computably enumerable. Let $\{r_n\}$ be any computable sequence of rationals which converges to x, and let m be a least-degree modulus of convergence for $r_n \to x$, as in lemma 1. Lemma 1 also guarantees that m has computably enumerable degree, and lemma 2 asserts that $\deg(x) \leq \deg(m)$. But the fact that m has c.e. degree, but x does not, forces the inequality to be strict, i.e. $\deg(x) < \deg(m)$.

Now let $\tilde{m} : \mathbb{N} \to \mathbb{N}$ be any modulus of convergence for $r_n \to x$. Since m is a least-degree modulus, we have $m \leq_T \tilde{m}$. Since $\deg(x) < \deg(m)$, it follows that $\deg(x) < \deg(\tilde{m})$. Hence $\tilde{m}$ is not $\mathbf{a}$-computable. Since $\tilde{m}$ was an arbitrary modulus of convergence, it follows that there is no $\mathbf{a}$-recursive modulus of convergence for $r_n \to x$.

This proves theorem 4. □

We now turn our attention to filling in the gap between the degree of unsolvability of a noncomputable, $\emptyset'$-computable real number, and that of the least-degree modulus of convergence of an approximating computable sequence of rational numbers. The idea is that, if we could control the sequence so that consecutive entries get successively closer to the limit x, then we could use any representation of x to compute a modulus of convergence. Hence the gap is filled by a set which tells us when a term in the sequence jumps further away from x than its predecessor was. Since we are concerned with numbers which are noncomputable, and hence irrational, we begin by showing that we lose very little generality if we assume that our approximating sequence is nonrepeating. If anything, this assumption shrinks the gap between $\deg(x)$ and the degree of a least-degree modulus, as the following lemma shows. For technical reasons, it is preferable to use a nonrepeating sequence.

Lemma 3. *Let $x \in [0,1]$ be $\mathbf{0}'$-computable but noncomputable. Let $\{r_n\}$ be any computable sequence of rational numbers which converges to x, and let $m : \mathbb{N} \to \mathbb{N}$ be a least-degree modulus of convergence for $r_n \to x$. Then there is a computable subsequence $\{r_n^*\}$ of $\{r_n\}$ such that $r_n^* \neq r_m^*$ for $n \neq m$ and such that, if m^* is a least-degree modulus of convergence for $r_n^* \to x$, we have $m^* \leq_T m$.*

Proof. We define $\{r_n^*\}$ inductively:

Set $r_1^* = r_1$. Now by induction suppose we have defined $r_n^* = r_k$ for some $k \geq n$. To define r_{n+1}^*, simply find the least $t > k$ such that $r_t \neq r_m^*$ for all $m \leq n$, and set $r_{n+1}^* = r_t$. (Such a t exists, since otherwise $\{r_m\}$ fluctuates between the finitely many values $r_1^*, \cdots, r_n^*$ for $m > k$, which implies, since $\{r_m\}$ is convergent, that it is eventually constant. This would mean its limit x is rational. But we are assuming x is noncomputable, hence irrational.)

To prove the claim about the modulus of convergence, let m^* be a least-degree modulus of convergence for $r_n^* \to x$, as in lemma 1. Then if $M : \mathbb{N} \to \mathbb{N}$ is any other modulus of convergence for $\{r_n^*\}$, we have $m^* \leq_T M$. Now let m be a least-degree modulus of convergence for $\{r_n\}$. We will construct an m-recursive modulus M for $\{r_n^*\}$, and then the fact that $m^* \leq_T M$ will give $m^* \leq_T m$.

Here is the construction of M: Let $p \in \mathbb{N}$ be given. To compute $M(p)$, first compute $m(p)$. Now, the construction of $\{r_n^*\}$ shows that there is an increasing, computable function $d : \mathbb{N} \to \mathbb{N}$ such that $r_n^* = r_{d(n)}$ for all n. So we compute $d(1), d(2), \cdots$, until some n is found with $d(n) > m(p)$. For the first such n, we set $M(p) = n$. Then since $\{r_n^*\}_{n \geq M(p)}$ is a subsequence of $\{r_n\}_{n \geq m(p)}$, we will have $n \geq M(p)$ implies $|r_n^* - x| < 2^{-p}$, so M is a modulus of convergence for $r_n^* \to x$. This proves lemma 3. □

Now we show how to "bridge the gap" between the degree of unsolvability of a real number, and that of the modulus of convergence of any nonrepeating computable sequence of rationals which converges to it.

Theorem 5. *Let $x \in [0, 1]$ be a noncomputable, $\emptyset'$-computable real number. Suppose $A \subseteq \mathbb{N}$ has the property that $x = \sum_{k=1}^{\infty} \chi_A(k) \cdot 2^{-k}$, so that $\mathbf{a} = deg(A)$ is the degree of unsolvability of x. Suppose $\{r_n\}$ is a computable, nonrepeating sequence of rational numbers which converges to x. Let $m : \mathbb{N} \to \mathbb{N}$ be a least-degree modulus of convergence for $r_n \to x$, as in lemma 1. Finally, define $G \subseteq \mathbb{N}$ as follows:*

$$G = \{n \in \mathbb{N} : (\exists k > n)|r_k - x| > |r_n - x|\}.$$

Then $m \equiv_T G \oplus A$.

Proof. The proof of this Turing equivalence will be broken down into three reducibilities, as follows:
(a) $A \leq_T m$;
(b) $G \leq_T m$;
(c) $m \leq_T A \oplus G$.
Proof of (a): This is nothing but lemma 2.
Proof of (b): Let $n \in \mathbb{N}$ be given. We give an m-effective procedure for determining whether $n \in G$ or $n \notin G$:

Since x is noncomputable, and hence irrational, we know that $|r_n - x| \neq 2^{-p}$ for all p; since $|r_n - x| > 0$, there exists some $p \in \mathbb{N}$ such that $|r_n - x| > 2^{-p}$.

We may find such a p, using m as an oracle, as follows.For $k = n+1, n+2, \cdots$, compute $|r_n - r_k|$ until we find integers k, p, s such that

$$k > n,\ s > p,\ k \geq m(s), \text{ and } |r_n - r_k| > 2^{-p} + 2^{-s}. \qquad (\star)$$

To see that such a triple does, in fact, exist, suppose $p \in \mathbb{N}$ is such that $|r_n - x| > 2^{-p}$. Then there exists some $s > p$ such that $|r_n - x| > 2^{-p} + 2^{-s}$. Since $|r_n - r_k| \to |r_n - x|$ as $k \to \infty$, there exists some k_0 such that $k \geq k_0$ implies $|r_n - r_k| > 2^{-p} + 2^{-s}$. Eventually some $k \geq \max\{k_0, n\}$ has $k \geq m(s)$. Then this triple (k, p, s) satisfies $(\star)$. Hence if we conduct an effective search through all triples $(k, p, s) \in \mathbb{N}^3$ such that $k > n$ and $s > p$, using the m-oracle to check whether $k \geq m(s)$ and checking effectively whether $|r_n - r_k| > 2^{-p} + 2^{-s}$, we may m-computably find a triple (k, p, s) satisfying $(\star)$.

Once we find such a triple, we know that $|r_n - x| > 2^{-p}$, as follows:

$$|r_n - r_k| \leq |r_n - x| + |x - r_k| < |r_n - x| + 2^{-s} \text{ (since } k \geq m(s))$$

so that

$$|r_n - x| > |r_n - r_k| - 2^{-s} > 2^{-p} + 2^{-s} - 2^{-s} = 2^{-p}.$$

Hence we have m-effectively found a $p \in \mathbb{N}$ such that $|r_n - x| > 2^{-p}$.

Now to continue with the process of determining whether $n \in G$, i.e. whether $(\exists k > n)|r_k - x| > |r_n - x|$.

Certainly if $k \geq m(p)$, then $|r_k - x| < 2^{-p} < |r_n - x|$, so we only need to determine whether $|r_k - x| > |r_n - x|$ for the finitely many $k \in \{n+1, n+2, \cdots, m(p)-1\}$. If $|r_k - x| < |r_n - x|$ for every $k \in \{n+1, n+2, \cdots, m(p)-1\}$, then $n \notin G$; and if we find some $k \in \{n+1, n+2, \cdots, m(p)-1\}$ such that $|r_k - x| > |r_n - x|$, then $n \in G$.

The next step, then, is to describe an m-effective procedure which determines, for any given pair n, k of natural numbers, whether or not $|r_k - x| > |r_n - x|$. First we note that, since $\{r_n\}$ is nonrepeating and x is irrational, we have $|r_k - x| \neq |r_n - x|$ whenever $k \neq n$; for if we had $k \neq n$ but $|r_k - x| = |r_n - x|$, then either $r_k = r_n$, which is impossible since $\{r_n\}$ is nonrepeating, or else we would have $x = \frac{1}{2}(r_k + r_n)$, which is impossible since x is irrational. To determine whether $|r_k - x| < |r_n - x|$ or $|r_k - x| > |r_n - x|$: By the above reasoning, we know that one inequality or the other must hold. We produce an m-effective procedure which tells us that $|r_k - x| < |r_n - x|$, if this is indeed the case. Since there are no assumptions about the indices other than $k \neq n$, we can simultaneously apply the procedure to the question "$|r_n - x| < |r_k - x|$?" Since one inequality or the other must hold, one of these procedures will eventually halt, telling us which inequality holds. Then carrying out this procedure for $k = n+1, \cdots, m(p)-1$ will, finally, tell us whether $n \in G$ or not.

Here is the m-effective procedure which, if $|r_k - x| < |r_n - x|$, halts and tells us so. Note that, if $|r_k - x| < |r_n - x|$, then there is some $s \in \mathbb{N}$ such that $|r_k - x| < |r_n - x| - 2^{-s}$. With this in mind, we do the following:

For $l = k+1, k+2, \cdots$, compute $|r_k - r_l|$ and $|r_n - r_l|$ until we find a pair (l, s) such that $l \geq m(s+1)$ and $|r_k - r_l| < |r_n - r_l| - 2^{-s}$. Such a pair (l, s) exists

whenever $|r_k - x| < |r_n - x|$, because if $s \in \mathbb{N}$ is such that $|r_k - x| < |r_n - x| - 2^{-s}$, then the fact that $|r_k - r_l| \to |r_k - x|$ and $|r_n - r_l| \to |r_n - x|$ as $l \to \infty$ implies that there exists some $l_0 \in \mathbb{N}$ such that, if $l \geq l_0$, then $|r_k - r_l| < |r_n - r_l| - 2^{-s}$. Then if $l \geq l_0$ is the first such integer with $l \geq m(s+1)$, (l, s) is a pair of the desired type. Hence if we list, in an effective way, all pairs $(l, s) \in \mathbb{N}^2$ with $l > k$, we will eventually find a pair satisfying $l \geq m(s+1)$ and $|r_k - r_l| < |r_n - r_l| - 2^{-s}$. But once we find such a pair (l, s), we know that $|r_k - x| < |r_n - x|$, as the following string of inequalities shows:

$$\begin{aligned} |r_k - x| &\leq |r_k - r_l| + |r_l - x| \\ &< (|r_n - r_l| - 2^{-s}) + 2^{-(s+1)} \\ &\leq |r_n - x| + |x - r_l| - 2^{-s} + 2^{-(s+1)} \\ &< |r_n - x| + 2^{-(s+1)} - 2^{-s} + 2^{-(s+1)} \\ &= |r_n - x|. \end{aligned}$$

Hence once we have found a pair (l, s) as in the preceding procedure, we know that $|r_k - x| < |r_n - x|$. If no such pair exists, then it must be true that $|r_n - x| < |r_k - x|$, and we can m-computably find a pair (l, s) which tells us so, using exactly the same procedure with k and n reversed. By running both m-recursive procedures simultaneously, we may determine whether $|r_k - x| < |r_n - x|$ or $|r_k - x| > |r_n - x|$.

We make this determination for $k = n+1, \cdots, m(p) - 1$, where n was the original input to our algorithm and p was found earlier in the proof. If we find such a $k \in \{n+1, \cdots, m(p)-1\}$ with $|r_k - x| > |r_n - x|$, we halt and say "$n \in G$". Otherwise, we halt and say "$n \notin G$". The only noneffective part of this decision procedure is the use of the function m as an oracle, so that $G \leq_T m$. This proves (b).

Proof of (c): We must show that $m \leq_T A \oplus G$. We prove this reducibility as follows: A modulus $m^* : \mathbb{N} \to \mathbb{N}$ of convergence for $r_n \to x$ will be constructed, using A and G as oracles, giving $m^* \leq_T A \oplus G$. Since m is a least-degree modulus of convergence, we know that $m \leq_T m^*$, which in turn gives $m \leq_T A \oplus G$.

First recall that x is irrational, so that we have $|r_n - x| \neq 2^{-p}$ for all $n, p \in \mathbb{N}$. Here is an A-effective procedure for determining whether $|r_n - x| < 2^{-p}$ or $|r_n - x| > 2^{-p}$, for any given pair (n, p).

Recall $x = \sum_{k=1}^{\infty} \chi_A(k) \cdot 2^{-k}$. Set $q_j = \sum_{k=1}^{j} \chi_A(k) \cdot 2^{-k}$, so that $q_j < x$ and

$$|q_j - x| = x - q_j = \sum_{k=j+1}^{\infty} \chi_A(k) \cdot 2^{-k} < \sum_{k=j+1}^{\infty} 2^{-k} = 2^{-j}$$

for all j. Also, $\{q_j\}$ is obviously an A-recursive sequence of rational numbers. We use A as an oracle to compute $|r_n - q_1|, |r_n - q_2|, \cdots$, until some pair (j, s) is found such that $j \geq s$ and one of the following holds:

(i) $|r_n - q_j| < 2^{-p} - 2^{-s}$, or

(ii) $|r_n - q_j| > 2^{-p} + 2^{-s}$.

To see that some such pair (j, s) exists, recall that $|r_n - x| \neq 2^{-p}$. If $|r_n - x| < 2^{-p}$, (i) will hold for some (j, s), and if $|r_n - x| > 2^{-p}$, (ii) will hold for some (j, s).

We effectively generate all pairs (j, s) with $j \geq s$, until either (i) or (ii) holds. If (i) holds, we know that $|r_n - x| < 2^{-p}$, as follows:

$$\begin{aligned} |r_n - x| \leq |r_n - q_j| + |q_j - x| &< (2^{-p} - 2^{-s}) + 2^{-j} \\ &\leq 2^{-p} - 2^{-s} + 2^{-s} \text{ (since } j \geq s) \\ &= 2^{-p}. \end{aligned}$$

On the other hand, if (ii) holds, we know that $|r_n - x| > 2^{-p}$, as follows: We have

$$|r_n - q_j| \leq |r_n - x| + |x - q_j| < |r_n - x| + 2^{-j} \leq |r_n - x| + 2^{-s},$$

so that

$$|r_n - x| > |r_n - q_j| - 2^{-s} > (2^{-p} + 2^{-s}) - 2^{-s} = 2^{-p}.$$

Hence we may A-effectively tell whether $|r_n - x| < 2^{-p}$ or $|r_n - x| > 2^{-p}$.

Now we define the modulus of convergence, m^*, which was referred to at the beginning of the proof. We set

$$m^*(p) = \mu n(|r_n - x| < 2^{-p} \wedge n \notin G).$$

Since the predicate "$|r_n - x| < 2^{-p}$" is A-computable, we see that, if m^* is total, then $m^* \leq A \oplus G$.

To see that m^* is total, we must show that some $n \in \mathbb{N}$ exists such that both $|r_n - x| < 2^{-p}$ and $n \notin G$ are true. Towards this end, we first note that, since $r_n \to x$, we have $|r_n - x| < 2^{-p}$ for all but finitely many n. We then observe that G is coinfinite. To see this, recall that

$$G = \{n \in \mathbb{N} : (\exists k > n)|r_k - x| > |r_n - x|\}.$$

If G were cofinite, there would be some N such that $n \geq N \Rightarrow n \in G$. Then $N \in G$, so there exists some $k_1 > N$ such that $|r_{k_1} - x| > |r_N - x|$. But k_1 is also in G, so there is some $k_2 > k_1$ such that $|r_{k_2} - x| > |r_{k_1} - x|$. Similarly, there is some $k_3 > k_2$ with $|r_{k_3} - x| > |r_{k_2} - x|$, and so on. In this way, we obtain a subsequence $\{r_{k_j}\}_{j=1}^{\infty}$ of $\{r_n\}$ such that $\{|r_{k_j} - x|\}$ is a strictly increasing sequence. But this is impossible, since $r_n \to x$. This contradiction means that G must be coinfinite.

So let n_0 be large enough that $|r_n - x| < 2^{-p}$ for $n \geq n_0$, and let n be the first $n \geq n_0$ with $n \notin G$. Then this n has $(|r_n - x| < 2^{-p} \wedge n \notin G)$, so $m^*(p) \downarrow$ (and $m^*(p) \leq n$ for this n).

Finally, we show that m^* is a modulus of convergence for $r_n \to x$. Suppose $k \geq m^*(p)$. Then $|r_{m^*(p)} - x| < 2^{-p}$ and, since $m^*(p) \notin G$ means that $(\forall k \geq m^*(p))(|r_k - x| < |r_{m^*(p)} - x|)$, we have $|r_k - x| < |r_{m^*(p)} - x| < 2^{-p}$, so that m^* is indeed a modulus of convergence.

So we have $m^* \leq_T A \oplus G$, and since (as was discussed at the beginning of the proof of (c)) we have $m \leq_T m^*$, it follows that $m \leq_T A \oplus G$. So (c) is proved, and hence theorem 5 is proved. □

References

1. Ho, C.-K.: Relatively recursive reals and real functions. Theoretical Computer Science 210 (1999), 99-120.
2. Pour-El, M.B. and J.I. Richards: Computability in Analysis and Physics. Berlin: Springer-Verlag, 1989.
3. Rice, H.G.: Recursive real numbers. Proc. Amer. Math. Soc. 5 (1954), 784-791.
4. Soare, R.: Recursion theory and Dedekind cuts. Trans. Amer. Math. Soc. 139 (1969), 271-294
5. Soare, R.I.: Recursively Enumerable Sets and Degrees. Berlin: Springer-Verlag, 1987.
6. Weihrauch, K.: Computable Analysis. Berlin: Springer-Verlag, 2000.

A Survey of Exact Arithmetic Implementations

Paul Gowland* and David Lester

Department of Computer Science, Manchester University
Oxford Road, Manchester M13 9PL, UK
{gowlandp,dlester}@cs.man.ac.uk

Abstract. This paper provides a survey of practical systems for exact arithmetic. We describe some of the methods used in their implementation, and suggest reasons for the performance differences displayed by some of the competing systems at this years CCA Exact Arithmetic Competition.
Because the practical aspects of the field of exact arithmetic are at an early stage, and many of the systems are prototypes, we have not discussed: portability, user-interfaces, and general usability. It is to be hoped that these aspects might be addressed by participants in any further competitions organised by the CCA committee.

1 Introduction

An exact arithmetic is one where the results of a sequence of calculations can be trusted. In this paper we wish to distinguish exact arithmetics from general variable (or multiple) precision arithmetics, by considering their behaviour. If the user does not need to provide details of the accuracy required of intermediate results, then the system is an exact arithmetic; alternatively, if whenever the user provides correct details of the accuracy required of intermediate results, the results of a sequence of calculations can be relied upon to be accurate, then the system is a variable precision arithmetic.

Since most of the currently competitive systems for exact arithmetic are implemented as C or C++ libraries, we have chosen not to concentrate on the issue of portability. Equally, there is little point in discussing most of the algorithms used in these systems, because they are mostly built on top of GNU GMP 3.0, and largely use similar algorithms for their operations. The CCA2000 arithmetic competition [Bla01] was intended to stimulate development of exact arithmetic systems, as well as to increase the range of operations provided and improve the usability of the systems. The intention was that competing systems should be able to perform the basic rational operations of $+$, $-$ $\times$, and $\div$, along with the elementary functions of sin, cos, $\tan^{-1}$, exp and log.

In Section 2, we provide an example that shows the need for exact arithmetic. In Section 3 we describe the relevant theory that we will use in the remainder of the article. In Section 4 we discuss IEEE floating point numbers, showing how they can be extended to provide: firstly, variable precision arithmetic in Section 5, and secondly, exact arithmetic in Section 6. Section 7 provides a conclusion.

* Supported by an EPSRC PhD studentship.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 30–47, 2001.

2 The Need for Exact Calculations

Exact arithmetic is not intended to be a complete replacement for the floating point arithmetic, simply because the floating point standard is so much faster and more memory efficient than the multi-precision packages. There are many applications where using an exact arithmetic package is unnecessary and would significantly reduce performance. There is however a growing need for arithmetic with a user-defined variable precision and a guaranteed level of error. Problems in numerical analysis such as root finding and solving ODE's can require very high precisions, and the answers to such problems can be nonsensical if a large precision is not used. Presented here is an example taken from [Krä98]. Solving a system of linear equations $Ax = b$ in floating point double precision:

Example 1.

$$A = a_{ij} := \begin{pmatrix} 64919121 & -159018721 \\ 41869520.5 & -102558961 \end{pmatrix}, \quad b := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

The results are given by

$$x_1 = a_{22}/(a_{11}a_{22} - a_{12}a_{21}), \quad x_2 = -a_{21}(a_{11}a_{22} - a_{12}a_{21})$$

and the floating point operations give

$$x_1 = 102558961.0, \quad x_2 = 41869520.5.$$

Every digit is wrong, the correct results are

$$x_1 = 205117922, \quad x_2 = 83739041.$$

As stated in the paper *"Although the data are exactly representable and only 5 floating point operations are done, the computed "solution" has nothing to do with the exact solution".*

3 Theoretical Approaches to the Real Numbers

In many approaches to computable analysis, for example those of Grzegorczyk [Grz55,Grz57,Grz59], Ko [Ko91], Lacombe [Lac55a,Lac55b,Lac55c,Lac58], Pour-El and Richards [PER89], and Weihrauch [Wei00], we are able to investigate computable functions over arbitrary real numbers. There are variations: Bishop and Bridges [BB85] take an intuitionistic approach to analysis, whilst Markov [Mar54] restricts attention to an analysis over the computable reals.

To make an efficient exact arithmetic we would like to be able to read and write real numbers to and from memory, and this makes it natural to consider the computable reals as a replacement for the reals; we will be particularly concerned with the details of their representation. There are a number of well known methods of constructing the real numbers from the rationals, amongst which there are:

- Nested intervals,
- Radix expansions,
- Signed digit sequences,
- Dedekind cuts,
- Continued fractions, and
- Cauchy sequences.

Some explanation of each representation will be necessary so that we may decide which are the most practical.

3.1 Nested Intervals

Nested intervals are much as they sound, they are a sequence of intervals with rational end points such that each interval is a subset of the previous one and the length of the intervals tends to zero.

3.2 Radix Expansions

Definition 1. *Radix Expansions are of the form:*

$$x = \sum_{n=1}^{\infty} \frac{\psi(n)}{\beta^n} \quad \textit{and} \quad \psi(n) < \beta \quad \textit{for } n = 1, 2, 3 \ldots$$

(Definition from [Mos57]).

With $\beta = 10$ we have the standard decimal representation of the real numbers. Whilst these systems have an admirable simplicity, many of the arithmetic operations we desire – such as addition and multiplication – are not available; see for example [Wei00, Example 2.1.4.7] or [Myh72]. In hardware engineering terms the reason is that to give the correct leading digit may involve an arbitrarily long carry propagation.

3.3 Signed Digit Expansions

To solve the carry propagation problem involved in radix expansions we can permit the expansions to have negative digits.

Definition 2. *Signed digit expansions are of the form:*

$$x = \sum_{n=1}^{\infty} \frac{\psi(n)}{\beta^n} \quad \textit{and} \quad |\psi(n)| < \beta \quad \textit{for } n = 1, 2, 3 \ldots$$

With this small change the arithmetic operations we might expect (addition, multiplication *etc.*) are available.

3.4 Dedekind Cuts

Definition 3. *There are 2 Dedekind cuts for a real number x, formulated as follows:*

- *A Dedekind left cut is the set of all rationals strictly less than x.*
- *A Dedekind right cut is the set of all rationals strictly greater than x.*

While this representation is of interest mathematically, and has been used by Harrison for mechanical theorem-proving [Har98], we know of no implementation that uses this form directly.

3.5 Continued Fractions

Definition 4. *Continued fraction expansions are of the form:*

$$x = \psi(0) + \cfrac{1}{\psi(1) + \cfrac{1}{\ddots + \cfrac{1}{\psi(n) + \cfrac{1}{\ddots}}}} \quad \textit{and} \quad 1 \leq \psi(n) \quad \textit{for } n = 1, 2, 3 \ldots$$

Hurwitz has shown that it is possible to represent the reals as continued fractions [Hur88,Hur89]. Gosper [Gos72,Gos77] has shown how to implement the operations of addition and multiplication on the continued fraction representations of the rationals. If we consider operations over the reals, we find that we have introduced an analogue of the carry propagation problem seen in the radix expansions [LL90,KM88]. Various attempts have been made to overcome this defect. In effect the solutions attempt to do for Hurwitz's continued fractions, what signed-digit expansions do for the radix expansions; see for example [KM90,Les01,MM94,Vui88]

3.6 Cauchy Sequences

Finally we have the Cauchy sequences. The definition differs slightly from paper to paper [Mos57,Ric54,Rob51], the version used in [GL00] is as follows:

Definition 5. *A computable real number x can be represented as an* Effective Cauchy Sequence *if there is an infinite computable sequence of rationals*

$$\left\{ \frac{n_0}{d_0}, \ \frac{n_1}{d_1}, \ \ldots, \ \frac{n_p}{d_p}, \ \ldots \right\},$$

with $d_p > 0$, and a modulus of convergence function $e : \mathcal{N} \to \mathcal{N}$ *which is recursive, such that for all $p \in \mathcal{N}$:*

$$k \geq e(p) \ \textit{implies} \ \left| x - \frac{n_k}{d_k} \right| < 2^{-p}$$

(Definition from [PER89]).

This forms the basis of most implementations of exact arithmetic. We note that it is possible to make the representation more dense by considering the integer n_k to be a function $n(k)$ with an implicit denominator of 2^k [WK87,Wei87].

4 Standard Floating Point Arithmetic

Floating point arithmetic is of course how computers deal with the real numbers and a great deal of effort has gone into developing them. The IEEE floating point standards have been adopted almost universally, giving a portable fixed precision representation of the real numbers. It is a compromise between the need for a fast efficient representation and the need for accuracy. The IEEE 754 standard [IEE85] was developed in 1985 and has been updated to the more generalised 854 standard [IEE94] which removes the dependencies on radix and wordlength [Pop95].

The standard float consists of 32 bits, 24 bits for the sign and mantissa, 8 bits for the exponent. There are actually four floating point precisions defined in the standard: single (24 bit mantissa/significand), single-extended (32 bit), double (53 bit), double-extended (64 bit); the extended type also allows for a bigger exponent range. These extra precisions allow libraries to compute expressions in a higher precision internally and correctly round the result that is outputted. The standard aims to decrease the rounding errors by computing internal calculations to an accuracy of 80 bits and then performing rounding at the final step, this increases the chance of the number being correctly rounded while using the same amount of storage. The philosophy of implementing the operations is described in the statement of the standard:

> *"Every operation is performed as if it first produced an intermediate result correct to infinite precision and with unbounded range and then rounded accordingly."*

There is no scope for dealing with the possibility of incorrectly rounded numbers, however IEEE incorporates other characteristics to over-come this failing, many of which are useful in numerical methods. IEEE features 4 different rounding modes including directed roundings, handling of exceptions and the NaN (Not a Number) special character. NaN's appear when an operation does such things as divide by zero, this special character can either propagate through the code or raise an exception. One standard concept in floating point arithmetic is that of normalised numbers, this is where the leading digit of the mantissa is non-zero. Requiring floats to be normalised makes the representation unique for each real number but it also means that zero cannot be represented.

Rounding error is always a factor in floating point arithmetic as we are often trying to represent an infinite quantity within a finite space, this error can be measured as the number of units in the last place (ulp's) or as the relative error; but ulp's are generally considered to be the most natural quantity. It is impossible to eliminate rounding error totally but it can certainly be decreased, one method used in the floating point standards is the guard digit. This can be used for example when subtracting numbers of different size, if an accuracy of p digits is required then each number is truncated to $p+1$ digits before calculation; thus greatly reducing the error without significantly affecting efficiency. Using one guard digit may still give a different answer than if the result was calculated

exactly and then rounded, so more guard digits can be used without damaging the performance noticeably.

Work has been done on finding the worst case errors that can appear in floating point arithmetic [Krä98]. Algebraic properties of the floating point numbers have also been proved [Mul00a], it is therefore possible to work with floating point arithmetic achieving meaningful results and predicting it's behaviour when problems occur. However as we have seen in Section 2 many applications need a higher level of accuracy and often a variable precision in their calculation. Example 1 shows how completely inadequate floating point arithmetic can be when used for certain mathematical problems.

As one would expect, research has been done on adding to the floating point standard in order to overcome the problems of rounding error, fixed precision and underflow/overflow *etc.* We shall look at some of these in order to get a full overview of the current state of computer arithmetic. [Hul82] gives a proposal of what is required of a floating point arithmetic.

> *"Involving clean arithmetic, complete precision control, and convenient exception handling properties."*

They define some capabilities required by floating point arithmetic in order for it to be used in numerical algorithms, they are as follows:

- *The first capability is to carry out different parts of a calculation in different precisions.*
- *The second capability is to carry out a portion of a calculation in more than one precision. Determining the effect of round-off error on a calculation is just one situation in which this need arises.*
- *The third capability is to have control over what is done in case of exceptions such as overflow or underflow. If underflow occurs we may wish to set the result to zero and carry on with the calculation. On the other hand, we may instead wish to repeat the calculation with a wider exponent range.*

5 Variable Precision Arithmetic

We would like to distinguish between two broad categories of computer implementations of accurate arithmetic. The first is variable precision arithmetic, where an individual calculation is performed to a certain accuracy and then truncated to a rational number. It is then the responsibility of the user to ensure that a sequence of such calculations retains sufficient intermediate accuracy for the final results to be relied upon. An example of such a problem is the logistic map problem used in the exact arithmetic competition at CCA2000 [Bla01].

$x := 1/\pi$**;**
for $i := 1$ **to** 1000 **do**
    $x := 15x(1-x)/4$**;**
println (x)**;**

Although we only desire 10 decimal places of the answer, depending on the algorithms used we will require at least 5000 decimal places of $1/\pi$ to generate the correct answer.

Many symbolic systems *e.g.* Mathematica, Maple, Maxima/Macsyma, and Pari-gp are able to give answers, but require the user to specify the desired accuracy for each of the assignments:

$$x := 15x(1-x)/4.$$

Two early approaches to the provision of a multiple precision Fortran library are those of Brent [Bre78] and Smith [Smi91]. More recently there have been other approaches developed for dealing with the variable precision arithmetic and we will give a brief overview of each.

5.1 ARIΘMOΣ

We start with the ARIΘMOΣ package as describe in [CKVV00]. Version 2 of ARIΘMOΣ is a class of libraries that offers multi-precision floating point (IEEE compliant), floating slash, rational interval and complex arithmetic. Their reasons behind this project are:

> *In short, for the numerical programmer who wants to make use of several of the alternatives above, there is no single platform that offers the best of all worlds. The new rational class library that we describe in this paper must be seen in this context, it accompanies a recently developed and very perfomant multiprecision class library, that fully complies with the principles of the IEEE 754-854 standards for floating point arithmetic.*

This library features:

- Exactly rounded conversions from and to hardware and multiprecision floats and hence more constructors.
- Special values like signed infinities, Non-integer and Non-rational values.
- The unordered relation to compare with Non-integer and Non-rational values.
- The power function for rational operands.
- A rational control word and status word among other things for exception handling and to prepare for rational interval arithmetic.

The library encompasses all types of arithmetic apart from the Cauchy sequence method and provides a useful platform for numerical analysis. The performance tests are impressive although there is only a small improvement over GNU MP 2.0.2 library and version 3.0 of this multiple precision package is now available that should be even faster.

5.2 MPFR

This system is also built on top of the GNU MP library, and is intended to provide a very fast multiprecision floating point library, and also follows the principles of the IEEE 754-854 standards for floating point arithmetic [Zim00]. As can be seen by the competition results obtained at CCA2000 [Bla01], this system is very competitive from an efficiency point of view. It utilizes Brent's AGM algorithm to perform the transcendental functions [Bre76]; further details of the implementation are in [BEIR00]. To perform the tests required by the CCA2000 competition, Zimmermann set up his system to be an interval arithmetic.

6 Exact Arithmetic

Exact arithmetic is an extension to variable precision arithmetic in which the system ensures that sequences of calculations generate correct answers. In this respect we are aiming to provide the primitive operations used by Blum, Shub and Smale [BS86,BSS89] in their 'Real RAM' machine. In this machine rational operations are available on arbitrary real numbers and are considered to take unit time to perform. As previously discussed there are problems with arranging for the storage of arbitrary real numbers, and for this reason one wishes to restrict attention to feasible machines such as those described by Brattka and Hertling [BH98] and Müller [Mül00b], in which only computable reals can be stored in the cells of the RAM. There are many different approaches to the implementation of exact arithmetic, and we begin with that of Boehm.

6.1 Boehm's Functional Approach

[BC90]

Having dismissed the signed digit representation as inefficient, Boehm went on to show how a reasonably efficient exact arithmetic system could be built using a 'functional representation' [BC90]. In this work a computable real was a function from the desired precision of the answer to a rational approximation. There are many optimisations that can be applied to this representation [BC90]:

> *"First, we maintain the best computed approximation for a number as part of it's representation. This information permits less accurate approximations to be computed trivially and provides an estimate (required by the reciprocal operation for example) as a seed for more accurate approximations.*
>
> *Second, we compute floating-point interval bounds as part of the representation of every constructive real number. These bounds eliminate the need to apply the functional representation to the demanded tolerance if the bounded interval is sufficiently small.*
>
> *Third, we represent hereditarily rational numbers explicitly as pairs of unbounded integers. In computations that rely exclusively on the basic arithmetic and relational operations, this optimisation reduces the implementation of the constructive reals to conventional rational arithmetic."*

This work is continued in [LB90], giving a detailed description of how to implement exact arithmetic from a functional point of view. They explain the problems of top down propagation of precision, this is when an expression is evaluated to a certain accuracy then each of it's sub-expressions needs to have a little extra accuracy added to it. If several requests are made of differing accuracy a large amount of computation is necessary. As an alternative bottom up propagation of available accuracy is used by implementing an interval based representation. [LB90] makes 2 statements that have profound effects to the way Exact arithmetic is implemented, these are:

> *1. Doubling the precision of a calculation at least doubles the number of significant digits determined by the result.*
> *2. Doubling the number of significant digits determined by both operands of an operation, and thus by the result, at least doubles the time required to perform the operation.*

The second statement in particular does not bode well for some of the packages described in Section 6. In [BD97,Mül98], the cost of re-calculation is obviously quite high and perhaps another method of evaluating the precision would be more efficient.

6.2 Ménissier-Morain

This research leads onto the work of Ménissier-Morain [MM94], in this thesis a functional approach is taken and implementations based on Cauchy sequences, signed digit and continued fractions are compared. A proof of correctness is also given for the Cauchy sequence approach. A real number r can be represented as a sequence of integers $(c_n)_{n\in N}$ with an implicit denominator β^{-n} such that:

$$\left| r - \frac{c_n}{\beta^n} \right| < \beta^{-n}$$

i.e. satisfying definition 5. The treatment in [MM94] goes on to prove properties about this representation before implementing the standard arithmetical operations. This work is related to that of [GL00]. It would appear, however, that she intended the package to be used as part of a floating point rather than a fixed point implementation. Her algorithms depend on the calculation of:

$$\text{msd}\,(x) \;=\; \min_{n\in\aleph}(|x_n| > 1) \;=\; -\lfloor \log_2(x) \rfloor.$$

This is undefined for $x = 0$ and relies implicitly on the arguments to the arithmetic being in the range

$$x \in (-1, 1),\ x \neq 0.$$

The implementation of [GL00] is based on Cauchy sequences in a similar way to [MM94] however it is not limited to numbers in the range $x \in (-1, 1),\ x \neq 0$

the library is valid for $x \in (-\infty, \infty)$. The implementation is based on the principle of top-down propagation of accuracy, when an operation is evaluated to a precision p each argument is evaluated to a higher accuracy in order to achieve this error bound and this propagates down through the expression tree. The advantage of this is that unlike [Mül86] the calculation will not need to be repeated at any stage to get the correct error bound, thus reducing the time complexity of the algorithms. The downside is that the arguments could be evaluated to a higher precision than might be needed, increasing space consumption occasionally. There is no definite answer yet as to which method is better; it depends mainly on the requirements of the user, is space or time efficiency more important?

[LB90] states that the main problem for top-down propagation of accuracy is that when repeated calls with different levels of accuracy occur, an expression could be evaluated to a high level of accuracy at one point and then time could be wasted evaluating the expression to a lower level of precision. The implementation in [GL00] overcomes this by only re-evaluating an expression if the accuracy is increased, when a low level of precision is required a quick calculation is done on the number stored to return the correct answer.

The approach to implementing the transcendental functions in [GL00] differs to [MM94] in that several functions are implemented separately; $exp(x)$, $sin(x)$, *etc.* are defined and then used in the implementation of other functions. [GL00] instead defines a power series algorithm and builds everything from this. Hence it is necessary to prove the correctness of just one function, making for a less complicated proof and reduced scope for human error.

6.3 Continued Fractions

We move on to a different philosophy of representing the real numbers, that of infinite streams. Although first described by Hurwitz in [Hur88,Hur89], and with basic rational arithmetic operations described by Gosper in [Gos72,Gos77], we quote the definition of continued fractions used in Vuillemin's paper [Vui88].

> *"The first n terms of such a stream are integers, and the last one $r(n)$ is a continuation, also commonly called closure in the Lisp community. This continuation is a function, finitely represented in a language $\mathcal{L}$, which is capable of evaluating the next integer term z_n, as well as the next continuation $r(n+1)$.*
>
> *In the decimal system, the value (interpretation) V of such a stream is given by:*
>
> $$V\langle z_0 z_1 \cdots z_{n-1} r(n)\rangle = \sum_{0 \le i < n} z_i 10^{-i} + \frac{V\langle r(n)\rangle}{10^n}.$$
>
> *With Continued Fractions, we have:*
>
> $$V[z_0 z_1 \cdots z_{n-1} r(n)] = z_0 + \cfrac{1}{z_1 + \cfrac{1}{\ddots + \cfrac{1}{z_{n-1} + \cfrac{1}{V[r_n]}}}}."$$

We rephrase this definition of a Continued Fraction as in [Les01].

Definition 6. *A finite Continued Fraction representation of the number x is a sequence of numbers*

$$[z_0, z_1, \ldots, z_n],$$

satisfying:

$$x = z_0 + \cfrac{1}{z_1 + \cfrac{1}{\ddots_{+\frac{1}{z_n}}}}$$

Infinite streams are also allowed, it is a general rule that a rational number can be represented by a finite stream; whereas a non-rational real number can only be described by an infinite stream. As stated in [Les01]

> *"It is possible to calculate the values of the integers $z_0, z_2 \cdots, z_n$ from x in the following way. Let $x = x_0$; we compute:*
>
> $$x_0 = z_0 + \frac{1}{x_1}, x_1 = z_1 + \frac{1}{x_2} \quad \ldots x_n = z_n$$
>
> *Where at each step we choose an integer z_n that is close to x_n. Obvious candidates for this calculation are the rounding operations floor, round, ceiling."*

Using floor leads to N-fractions and using round leads to Z-fractions, these lead to specific definitions which we will not go in to here; there are also Euclidean fractions (E-fractions). These fractions often lead to elegant representations of certain "difficult" numbers such as $\sqrt{2}$, here are a few examples of N-fractions taken from [Vui88]:

Example 2.

$$\begin{array}{lll} \frac{1+\sqrt{5}}{2} = \phi & = [1\,1\,1\,1\ldots] & = [,1] \\ \sqrt{2} & = [1\,2\,2\,2\cdots] & = [1\,,2] \\ \sqrt{3} & = [1\,1\,2\,1\,2\,1\,2\cdots] & = [1\,,1\,2] \\ e & = [2\,1\,2\,1\,1\,4\,1\,1\,6\,1\cdots] & = [2, 1\,2n+2\,1] \\ \pi & = [3\,7\,15\,1\,292\,1\,1\,1\,2\,1\cdots] & \end{array}$$

Z-fractions also have some useful results e.g. $\sqrt{2} = [1\,,2]$ obviously more useful than the infinite decimal expansion. Some more important properties of N and Z fractions are summed up in Lagrange's theorem:

Theorem 1. *Let $r = [z_0, z_1, \ldots, z_n, \ldots]$ be the N-fraction or Z-fraction expansion of some finite number. Then:*

1. *The continued fraction has only one term if and only if $r \in \mathcal{Z}$ is an integer.*
2. *The continued fraction is finite if and only if the number $r \in \mathcal{Q}$ is rational.*
3. *The continued fraction is eventually periodic if and only if $r \in \sqrt{\mathcal{Q}_{>0}}$ is the irrational square root of a positive rational.*

4. *The continued fraction approximants* $r_n = [z_0, z_1, \ldots, z_n]$ *converge to* r *with speed:*

$$|r - r_n| < \phi^{-n},$$

where $\phi = 1.618\ldots$ *is the golden ratio.*

One of the main problems with continued fractions is that it is very difficult to determine the underlying complexity of their basic operations as can be seen in [Ko86]:

> *"We give some evidence for the common intuition that there is no efficient (polynomial-time) algorithm that maps two Continued Fractions*
>
> $$x = [a_0, a_1, \ldots] \quad \textit{and} \quad y = [b_0, b_1, \ldots]$$
>
> *to a third Continued Fraction* $[c_0, c_1, \ldots]$ *which represents* $x + y$*."*
>
> **Theorem 2.** *There exist two real numbers* x *and* y *such that* LC_x *(Left Cut of* x*) and* LC_y *are polynomial-time computable, but* LC_{x+y} *is not polynomial-time computable.*

There is an equivalence between Dedekind left cuts and continued fractions. Work has been done on improving these efficiency problems [Vui88,Les01], but the algorithms are still not straightforward, unlike other representations. An area in which continued fractions have some pleasing properties is with the transcendental functions as given in [Les01], following Gauss [Gau12] and Vuillemin [Vui88], the stream representation of exp and log can be easily calculated to any accuracy required as can be seen in their definitions:

Definition 7.

$$\exp(r) = \left[1, , \frac{1}{r}, 2, \frac{3}{r}, -2, \frac{5}{r}, 2, \frac{7}{r}, -2, \ldots\right]$$

and

$$\log(r) = \left[0, \frac{1}{r-1}, \frac{2}{1}, \frac{3}{r-1}, \frac{2}{2}, \frac{5}{r-1}, \frac{2}{3}, \ldots\right].$$

A method to implement exact arithmetic using Continued Fractions is described in [Les01]. This uses the lazy functional style of Haskell in order to implement the infinite streams involved. The library in [GL00] was preceded by this implementation of Continued Fractions, however it was decided that a full proof of it's correctness would be difficult, so the current (fast Cauchy sequence) implementation was developed.

6.4 Linear Fractional Transformations

We move on to Linear Fractional Transformations as a representation of the real numbers, this method is described in [EP97]:

A sequence of shrinking nested intervals is then represented by an infinite product of matrices with integer coefficients such that the first so-called sign matrix determines an interval on which the real number lies. The subsequent so-called digit matrices have non-negative integer coefficients and successively refine that interval.

This seems to be a clever way of encoding the real numbers although it is possibly slightly over-complicated. The rational arithmetic operations are implemented in a similar fashion to Gosper's continued fraction algorithms [Gos72,Gos77], with the transcendental functions being implemented using variations on the methods of Gauss [Gau12]. Full details of these features of the system are in [Pot99]. The paper goes on to describe properties of this representation. Furthermore, its stream-based nature appears to be one of the reasons for the system's performance in the CCA2000 competition [Bla01].

6.5 Berthelot and Daumas' Interval Arithmetic

Interval arithmetic is often seen as an efficient useful way of dealing with the real numbers. The basic idea is that the 2 end points of the interval in which the real number lies are stored as exact rationals or floats, and all calculations are done on these 2 numbers. When the results need to be rounded, they are usually rounded away from the centre of the interval so as to ensure that the real number is still confined by these 2 values. We now formulate the interval arithmetic described in [BD97].

"In our library, we define a real number X as a double infinite sequence of numbers

$$(a_n, \beta_n) \in \mathcal{Z} \times \mathcal{R}_+^*$$

such that

$$|a_n| \leq \frac{\beta_{n-1}}{\beta_n} - 1 \quad \textit{and} \quad \sum_{n+1}^{\infty} \beta_i \leq \beta_n.$$

The nth embedded interval X_n is defined from it's midpoint x_n as

$$X_n \;=\; [x_n - \beta_n,\; x_n + \beta_n], \quad \text{where} \quad x_n = \sum_{i=0}^{n} a_n \beta_n."$$

The usual problem with interval arithmetic is that the size of the interval will obviously grow during each calculation and often the resulting interval is too large. When this happens the computations have to be repeated to a greater accuracy, wasting time. In [BD97] the programme backtracks increasing the precision where needed, this is more efficient, but it is still open to debate if this method of re-calculation to a higher precision is more efficient than other methods.

6.6 iRRAM

We now discuss iRRAM [Mül98]. In this approach the program is executed with a with all variables held to a precision (p) and the error in the answer (ε) is determined. In this respect it is much like an interval arithmetic. If the desired maximum error of the answer (ε') is smaller than the actual error (ε), then the program is executed again at a higher precision ($p' = p\frac{\varepsilon'}{\varepsilon}$). This process continued until an answer is generated that is accurate enough. In discussions with Müller, he reports that 'usually' only the second iteration of the program is required.

> *"Instead of using signed digits, the iRRAM is based on a simplified interval arithmetic. Additionally it avoids storing intermediate results as often as possible. Both modifications drastically reduce memory requirements for large computations. E.g. the package is capable to do numerical inversion of (badly conditioned) Hilbert matrices of size* 200×200 *using about 90MB with Gauss' method, where storing the input matrix with the necessary precision needs about 40MB (to compute results with an error of at most* 2^{-50}*)."*

This is very efficient when one considers that using signed digits increases the memory consumption by a factor of 400. iRRAM works in the same way as the package mentioned in [BD97], the result is computed and if the error is too large then the result is repeated with a higher precision. It might be the case that many iterations are needed before the correct accuracy is reached, but the advantage is that each variable is only calculated to the minimum accuracy required.

There are a number of reasons for the impressive display this system produced at the CCA2000 competition. Firstly, the error propagation involves not just bit-at-a-time loss of precision as exhibited by the MAP package, but a more precise measure of the error using two 32 bit machine words, one as a mantissa, and the other as the exponent to give the error in an iRRAM variable. Secondly, elementary transcendental functions are implemented using the AGM method [Bre76], with aggressive range-reduction applied.

6.7 MAP

In this approach, Lester works backwards from the desired accuracy of the answer to the required accuracy of the initial values. To avoid problems at 0, all numbers are held as fixed-point rather than floating-point entities. Underlying the package is the GNU GMP 3.0 library of integer operations. The elementary functions are calculated using Taylor series (rather than the more efficient AGM algorithm [Bre76]), but all arguments are range-reduced to give very small values (typically 2^{-96}) on which the Taylor series expansions are required to work. One small optimization is that a small number of frequently used irrational constants – such as π and $\log(2)$ – are initialized to $30,000$ bits accuracy. It is this that allows the system to perform as well as it does against MPFR and iRRAM.

A comparison of the execution speeds of some of these systems can be found in Jens Blanck's paper [Bla01] in these proceedings.

7 Conclusion

We have discussed the current state of research in exact arithmetic trying to be as thorough as possible but no doubt there are things that have been left out.

The main approaches involve extending current floating point standards to reduce their inherent problems and introducing multiple precision to the format. Or completely separate implementations using infinite streams, rational approximations, linear fractional transforms or floating point interval calculations. No method is perfect, all trying to balance the need for space and time efficiency with the need for accuracy.

There is a lot of research into the theoretical properties of the real numbers but here we have concentrated on implementation rather than theory, particularly concerning the example in Section 2. There is an increasing need for exact arithmetic and increases in computing power to some extent negate the inherent efficiency problems of such packages.

References

BB85. Errett Bishop and Douglas S. Bridges. *Constructive Analysis*, volume 279 of *Grundlehren der mathematischen Wissenschaften*. Springer, Berlin, 1985.

BC90. H. Boehm and R. Cartwright. Exact real arithmetic: Formulating real numbers as functions. In D.A. Turner, editor, *Research Topics in Functional Programming*, University of Texas at Austin Year of Programming, pages 43–64. Addison-Wesley, 1990.

BD97. D. Berthelot and M. Daumas. Computing on sequences of embedded intervals. *Reliable Computing*, 3(3):219–227, 1997.

BEIR00. J.-C. Bajard, M.D. Ercegovac, L. Imbert, and F. Rico. Fast evaluation of elementary functions with combined shift-and-add polynomial methods. In *Fourth Conference on Real Numbers and Computers*, pages 75–87, Schloss Dagstuhl, Saarland, Germany, April 2000.

BH98. Vasco Brattka and Peter Hertling. Feasible real random access machines. *Journal of Complexity*, 14(4):490–526, 1998.

Bla01. J. Blanck. Exact real arithmetic systems: results of competion. In Jens Blanck, Vasco Brattka, and Peter Hertling, editors, *Computability and Complexity in Analysis 2000*, Berlin, 2001. Springer.

Bre76. R.P. Brent. Fast multiple precision evaluation of elementary functions. *Journal of the ACM*, 23:242–251, 1976.

Bre78. R.P. Brent. A Fortran multiple precision package. *ACM Transactions on Mathematical Software*, 4:57–70, 1978.

BS86. L. Blum and M. Shub. Evaluating rational functions: Infinite precision is finite cost and tractable on average. *SIAM Jounal of Computing*, 15(2):384–398, 1986.

BSS89. L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: *NP*-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1), July 1989.

CKVV00. A. Cuyt, P. Kuterna, P.B. Verdonk, and J. Vervloet. The class library for exact rational arithmetic in ARIΘMOΣ. In *Proceedings of the Fourth Real Numbers and Computers*, pages 141 – 160, 2000.

EP97. Abbas Edalat and Peter John Potts. A new representation for exact real numbers. *Electronical Notes in Theoretical Computer Science*, 6:14 pp., 1997. Mathematical foundations of programming semantics (Pittsburgh, PA, 1997).

Gau12. C.F. Gauss. Disquisitiones generales circa seriem infinitam $1 + \frac{\alpha\beta}{1\,.\,\gamma}x + \frac{\alpha(\alpha+1)\beta(\beta+1)}{1\,.\,2\,.\,\gamma(\gamma+1)}xx + \frac{\alpha(\alpha+1)(\alpha+2)\beta(\beta+1)(\beta+2)}{1\,.\,2\,.\,3\,.\,\gamma(\gamma+1)(\gamma+2)}x^3$ etc. pars prior. In *Werke*, volume 3, pages 123–162. Königlichen Gesellschaft der Wissenschaften, Göttingen, 1812.

GL00. P. Gowland and D. Lester. The correctness of an implementation of exact arithmetic. In *Proceedings of the Fourth Real Numbers and Computers Conference*, pages 125–140, 2000.

Gos72. R.W. Gosper. Continued fraction arithmetic. HAKMEM 101b, MIT, 1972.

Gos77. R.W. Gosper. Continued fraction arithmetic. Unpublished Draft Paper, 1977.

Grz55. Andrzej Grzegorczyk. Computable functionals. *Fundamenta Mathematicae*, 42:168–202, 1955.

Grz57. Andrzej Grzegorczyk. On the definitions of computable real continuous functions. *Fundamenta Mathematicae*, 44:61–71, 1957.

Grz59. Andrzej Grzegorczyk. Some approaches to constructive analysis. In A. Heyting, editor, *Constructivity in mathematics*, Studies in Logic and The Foundations of Mathematics, pages 43–61, Amsterdam, 1959. North-Holland. Colloquium at Amsterdam, 1957.

Har98. J. Harrison. *Theorem Proving with the Real Numbers*. Springer-Verlag, London, 1998.

Hul82. T.E. Hull. The use of controlled precision. *The Relationship Between Numerical Computation and Programming Languages*, pages 71–84, 1982.

Hur88. A. Hurwitz. Über die Entwicklung Komplexer Grössen in Kettenbrüche. *Acta Mathematica*, 11:187–200, 1888.

Hur89. A. Hurwitz. Über eine besondere Art der Kettenbruch-Entwicklung reeller Grössen. *Acta Mathematica*, 12:367–405, 1889.

IEE85. IEEE, New York. *IEEE Standard for Binary Floating-point Arithmetic, IEEE Standard 754-1985*, 1985.

IEE94. IEEE, New York. *IEEE Standard for Radix and Format Independent Floating-point Arithmetic, IEEE Standard 854-1994*, 1994.

KM88. P. Kornerup and D.W. Matula. An on-line arithmetic unit for bit-pipelined rational arithmetic. *Journal of Parallel and Distributed Computing*, 5(3):310–330, 1988.

KM90. P. Kornerup and D.W. Matula. An algorithm for redundant binary bit-pipelined rational arithmetic. *IEEE Transactions on Computers*, 39(8):1106–1115, 1990.

Ko86. Ker-I Ko. On the continued fraction representation of computable real numbers. *Theoretical Computer Science*, 47:299–313, 1986. corr. ibid., Vol. 54 (1987), Pages 341–343.

Ko91. Ker-I Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1991.

Krä98. W. Krämer. A priori worst case error bounds for floating-point computations. *IEEE Transactions on Computers*, 47(7):750–756, 1998.

Lac55a. Daniel Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles I. *Comptes Rendus Académie des Sciences Paris*, 240:2478–2480, June 1955. Théorie des fonctions.

Lac55b. Daniel Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles II. *Comptes Rendus Académie des Sciences Paris*, 241:13–14, July 1955. Théorie des fonctions.

Lac55c. Daniel Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles III. *Comptes Rendus Académie des Sciences Paris*, 241:151–153, July 1955. Théorie des fonctions.

Lac58. Daniel Lacombe. Sur les possibilites d'extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles. In *Le raisonnement en mathematiques et en sciences*, pages 67–75, Paris, 1958. Editions du Centre National de la Recherche Scientifique. Colloques Internationaux du Centre National de la Recherche Scientifique, LXX.

LB90. V.A. Lee, Jr and H.J. Boehm. Optimizing programs over the constructive reals. In *Proceedings of The ACM SIGPLAN' 90 Conference on Programming Language, Design and Implementation*, pages 102–111, 1990.

Les01. D. Lester. Effective continued fractions. In *Proceedings of the Fifteenth IEEE Arithmetic Conference*, June 2001.

LL90. Salah Labhalla and Henri Lombardi. Real numbers, continued fractions, and complexity classes. *Annals of Pure and Applied Logic*, 50:1–28, 1990.

Mar54. A.A. Markov. On the continuity of constructive functions (Russian). *Uspekhi Mat. Nauk (N.S.)*, 9:226–230, 1954.

MM94. V. Ménissier-Morain. *Arithmétique Exacte*. PhD thesis, L'Université Paris VII, December 1994.

Mos57. A. Mostowski. On computable sequences. *Fundamenta Mathematicae*, 44:37–51, 1957.

Mül86. Norbert Th. Müller. Subpolynomial complexity classes of real functions and real numbers. In Laurent Kott, editor, *Proceedings of the 13th International Colloquium on Automata, Languages, and Programming*, volume 226 of *Lecture Notes in Computer Science*, pages 284–293, Berlin, 1986. Springer.

Mül98. Norbert Th. Müller. Implementing limits in an interactive RealRAM. In J.-M. Chesneaux, F. Jézéquel, J.-L. Lamotte, and J. Vignes, editors, *Third Real Numbers and Computers Conference*, pages 59–66. Université Pierre et Marie Curie, Paris, 1998. Paris, France, April 27-29, 1998.

Mul00a. J. M. Muller. Some algebraic properties of floating-point arithmetic. In *Proceedings of the Fourth Real Numbers and Computers*, pages 31 – 38, 2000.

Mül00b. Norbert Th. Müller. The iRRAM: Exact arithmetic in C++. In Jens Blanck, Vasco Brattka, Peter Hertling, and Klaus Weihrauch, editors, *Computability and Complexity in Analysis*, volume 272 of *Informatik Berichte*, pages 319–350. FernUniversität Hagen, September 2000. CCA2000 Workshop, Swansea, Wales, September 17–19, 2000.

Myh72. J. Myhill. What is a real number? *The American Mathematical Monthly*, 79:748–754, 1972.

PER89. Marian B. Pour-El and J. Ian Richards. *Computability in Analysis and Physics*. Perspectives in Mathematical Logic. Springer, Berlin, 1989.

Pop95. E.D. Popova. On a formally correct implementation of IEEE computer arithmetic. *Journal of Universal Computer Science*, 1(7):560–569, 1995.

Pot99. P.J. Potts. *Exact Real Arithmetic using Möbius Transformations*. PhD thesis, Imperial College of Science, Technology and Medicine, London University, March 1999.

Ric54. H. Rice. Recursive real numbers. *Proc. Amer. Math. Soc.*, 5:784–791, 1954.

Rob51. R.M. Robinson. Review of "Peter, R., Rekursive Funktionen". *The Journal of Symbolic Logic*, 16:280–282, 1951.

Smi91. D.M. Smith. ALGORITHM 693, A FORTRAN package for floating-point multiple-precision arithmetic. *ACM Transactions on Mathematical Software*, 17(2):273–283, 1991.

Vui88. J. Vuillemin. Exact real computer arithmetic with continued fractions. In *Proceedings of the 1988 ACM Conference on Lisp and Functional Programming*, pages 14–27, Snowbird, Utah, 25–27 July 1988.

Wei87. Klaus Weihrauch. *Computability*, volume 9 of *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, 1987.

Wei00. Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.

WK87. Klaus Weihrauch and Christoph Kreitz. Representations of the real numbers and of the open subsets of the set of real numbers. *Annals of Pure and Applied Logic*, 35:247–260, 1987.

Zim00. P. Zimmermann. MPFR: A library for multiprecision floating-point arithmetic with exact rounding. In *Fourth Conference on Real Numbers and Computers*, pages 89–90, Schloss Dagstuhl, Saarland, Germany, April 2000.

Standard Representations of Effective Metric Spaces

Armin Hemmerling

Ernst-Moritz-Arndt-Universität Greifswald
Institut für Mathematik und Informatik
Friedrich-Ludwig-Jahn-Str. 15a
D 17487 Greifswald, Germany
hemmerli@mail.uni-greifswald.de

Abstract. Representations of effective metric spaces which are equivalent to the normed Cauchy representations are called standard representations. They can be characterized by their computability properties, i.e., by having both computable extensions as well as inversions computable as relations. Another characterization is given by many-sorted so-called indicator functions. Under weak suppositions on the underlying spaces, there are also single-sorted effectively categorical structures characterizing the standard representations. To this purpose, both basic constants and infinitary basic functions of the structures are necessary.

1 Introduction

The approach to approximate computability between effective metric spaces we introduced and applied in [3] is essentially based on the principles of Weihrauch's type two theory of effectivity, cf. [9,11]. Function-oracle Turing machines in the sense of Ko and Friedman [6,7] are used to perform transformations between sequences of natural numbers which are interpreted as index sequences of fast converging Cauchy sequences of skeleton points within the metric spaces under consideration. So we avoid the explicit use of representations by fixing the normed Cauchy representations as the canonically given ones. Standard representations are such ones which are computationally equivalent to the normed Cauchy representations. In our approach, they can be characterized by their computational properties as it has been done in [3] for the metric space of real numbers.

In the present paper, we first generalize these results in order to characterize the standard representations of arbitrary effective metric spaces. Then we deal with the problem of characterizing the standard representations by requiring the computability of finitely many so-called indicator functions. More precisely, we ask for functions $f_1, \ldots, f_n$ which are computable with respect to a representation ϱ iff ϱ is a standard representation. This problem was introduced by Hertling's paper [4] on an effectively categorical structure over the real numbers. Brattka [1] has applied closely related notions and techniques and has got

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 48–68, 2001.

similar results within his many-sorted approach to computability over topological structures. Under rather weak suppositions on the underlying space, one can show that there are effectively categorical single-sorted structures, including basic constants and relations, which characterize the standard representations. It can be shown that the use of constants and infinitary functions is not avoidable in this context. Unfortunately, the question if the task can be solved by a single-sorted algebra, i.e., a structure without basic relations, remains open.

This paper is organized as follows. Section 2 reports the basic notions on effective metric spaces and approximate computability, also for infinitary functions between products of spaces. Section 3 deals with computability w.r.t. representations and generalizes results from [3] to arbitrary spaces. In Section 4, sets of many-sorted indicator functions are given, whereas Section 5 deals with effectively categorical single-sorted structures. Section 6 supplies counterexamples showing that constants and infinitary functions are necessary in structures characterizing the standard representations. Throughout the paper, proofs are only outlined, since they are mainly based on standard techniques of the theory of (approximate) computability. We shall include some hints to related notions and results in the literature. General overviews to computability on metric spaces can be found in [1,3,5,6,9,10,11].

2 Effective Metric Spaces and Approximate Computability

We briefly recall some basic concepts and notations introduced and discussed in more detail in [3].

Definition 1. *An effective metric space (briefly: EMS) is a triple* $\mathcal{X} = (X, d, S)$, *where*
- (X, d) *is a complete metric space and*
- $S = (s_n)_{n\in\mathbb{N}}$ *is a sequence of elements from* X *such that*
  - *the range,* $\mathrm{ran}(S) = \{s_n : n \in \mathbb{N}\}$, *is dense in* (X, d) *and*
  - *the set* $D_< = \{\langle m, n, k\rangle : d(s_m, s_n) < q_k\}$ *is r. e. (i.e., recursively enumerable).*

If, moreover,
- *the set* $D_> = \{\langle m, n, k\rangle : d(s_m, s_n) > q_k\}$ *is r. e.,*

then $\mathcal{X}$ *is said to be a strongly effective metric space (briefly: SEMS).*

As usual, $\langle n_1, \ldots, n_k\rangle$ denotes the Cantor number of a k-tuple $(n_1, \ldots, n_k) \in \mathbb{N}^k$, and $\nu_{\mathbb{Q}} = (q_k)_{k\in\mathbb{N}}$ is a fixed (total) standard numbering of the set $\mathbb{Q}$ of all rational numbers.

The sequence $S = (s_n)_{n\in\mathbb{N}}$ is called the *skeleton* of the EMS $\mathcal{X}$. It gives the basis for the application of recursion theory to the complete separable (i.e., *Polish*) space (X, d).

Examples of SEMSs we shall deal with in the sequel are the following:

1. Discrete EMSs are of the form (X, d_X, S_X), where $d_X(x, x') = 1$ for $x \neq x'$ and $S_X = (x_n)_{n\in\mathbb{N}}$ gives a numbering of the finite or countably infinite set

$X = \mathrm{ran}(S_X)$. In particular, the discrete EMS over the natural numbers is specified as $(\mathbb{N}, d_{\mathbb{N}}, \mathrm{id}_{\mathbb{N}})$.

2. $(\mathbb{R}, d_{\mathbb{R}}, S_{\mathbb{R}})$ is the space of real numbers, with the natural distance function $d_{\mathbb{R}}$ and the standard numbering of the rational numbers, $S_{\mathbb{R}} = \nu_{\mathbb{Q}}$.
3. $(\mathbb{B}, d_{\mathbb{B}}, S_{\mathbb{B}})$ is the Baire space of binary sequences, $\mathbb{B} = \{0,1\}^{\omega}$. The Baire metric $d_{\mathbb{B}}$ is here defined by $d_{\mathbb{B}}((\beta_n)_{n\in\mathbb{N}}, (\beta'_n)_{n\in\mathbb{N}}) = 2^{-\min\{n:\, \beta_n \neq \beta'_n\}}$, for $(\beta_n)_{n\in\mathbb{N}} \neq (\beta'_n)_{n\in\mathbb{N}}$, and the skeleton $S_{\mathbb{B}}$ corresponds to an effective bijective standard numbering of the binary sequences ultimately stationary with 0: $\mathrm{ran}(S_{\mathbb{B}}) = \{(\beta_0, \beta_1, \ldots, \beta_l, 0, 0, \ldots) : \beta_0, \beta_1, \ldots, \beta_l \in \{0,1\}, l \in \mathbb{N}\}$.
4. $(\mathbb{F}, d_{\mathbb{F}}, S_{\mathbb{F}})$ denotes the Baire space of sequences of natural numbers, i.e., $\mathbb{F} = \mathbb{N}^{\omega}$, the Baire metric $d_{\mathbb{F}}$ is defined like $d_{\mathbb{B}}$, and $S_{\mathbb{F}}$ is an effective bijective standard numbering of the ultimately 0-stationary sequences, i.e., $\mathrm{ran}(S_{\mathbb{F}}) = \{(\nu_0, \nu_1, \ldots, \nu_l, 0, 0, \ldots) : \nu_0, \nu_1, \ldots, \nu_l \in \mathbb{N}, l \in \mathbb{N}\}$.

Notice that ω is the first infinite ordinal number, it characterizes the order type of the (ordered set of) natural numbers, $(\mathbb{N}, <)$. In this sense, M^{ω} can be identified with $M^{\mathbb{N}}$, for any set M. The special SEMSs specified at 1., 2., 3. and 4., respectively, will also simply be denoted by $\mathbb{N}$, $\mathbb{R}$, $\mathbb{B}$ and $\mathbb{F}$.

One easily sees that SEMSs are characterized by the approximate computability of the distance functions on their skeletons, whereas EMSs are characterized by the right-cut computability of the distance functions on the skeletons, cf. Section 4. For these and some more details and further examples, we refer to [3] and [10].

A natural notion of (approximate) computability for functions between EMSs is introduced by means of classical recursion theory after representing the elements $x \in X$ by *index sequences* $\sigma = (i_0, i_1, i_2, \ldots) \in \mathbb{F}$ for which the corresponding *S-sequences*, $S \circ \sigma = (s_{i_0}, s_{i_1}, s_{i_2}, \ldots) \in \mathrm{ran}(S)^{\omega}$, converge effectively to x. The sequence $S \circ \sigma = (s_{i_m})_{m\in\mathbb{N}}$ is said to be *effectively converging* to x if $d(x, s_{i_m}) < 2^{-m}$, for all $m \in \mathbb{N}$. Then it is also called a *standard Cauchy function* of x (with respect to the skeleton S). By $\mathrm{CF}_x^{(S)}$, we denote the set of all index sequences σ whose S-sequences, $S \circ \sigma$, converge effectively to x.

Correspondingly, an element $x \in X$ is said to be *computable* on $\mathcal{X}$ if $\mathrm{CF}_x^{(S)}$ contains a recursive sequence (of natural numbers).

Generalizing the approach by Ko and Friedman [6,7] introduced for real functions, approximate computability of partial functions $f : X \rightarrowtail Y$, for EMSs $\mathcal{X} = (X, d, S)$ and $\mathcal{Y} = (Y, d', S')$, can naturally be defined by means of *function-oracle Turing machines* (briefly: *OTMs*). They process natural numbers and access to the arguments $x \in X$ via arbitrary oracle sequences $\sigma \in \mathrm{CF}_x^{(S)}$. More precisely, in the course of its work, a machine can put oracle queries of the form "m?" which are answered by the $(m+1)$st index i_m of the oracle sequence $\sigma = (i_0, i_1, i_2, \ldots)$.

To compute the function f, an OTM $\mathcal{M}$ has to produce an output $\mathcal{M}^{\sigma}(n) = j_n \in \mathbb{N}$, for any input $n \in \mathbb{N}$, in such a way that $(j_n)_{n\in\mathbb{N}} \in \mathrm{CF}_{f(x)}^{(S')}$ if $\sigma \in \mathrm{CF}_x^{(S)}$ and $x \in \mathrm{dom}(f)$. For all $x \in X \setminus \mathrm{dom}(f)$ and all oracles $\sigma \in \mathrm{CF}_x^{(S)}$, there has to be an input n for which the machine never halts, i.e., $\mathcal{M}^{\sigma}(n)$ remains undefined.

Here we give the details of definition for the more general case of partial functions between product sets,

$$f : \prod_{\kappa<\alpha} X_\kappa \rightarrowtail \prod_{\lambda<\beta} Y_\lambda,$$

for (finite or infinite) arities $\alpha, \beta \in \mathbb{N}_+ \cup \{\omega\}$ and EMSs $\mathcal{X}_\kappa = (X_\kappa, d_\kappa, S_\kappa)$ and $\mathcal{Y}_\lambda = (Y_\lambda, d'_\lambda, S'_\lambda)$. Thus, $\prod_{\kappa<\alpha} X_\kappa$ consists of all sequences of length α with elements from X_κ in their $(\kappa+1)$st components:

$$\prod_{\kappa<\alpha} X_\kappa = \{\boldsymbol{x} = (x_\kappa)_{\kappa<\alpha} : x_\kappa \in X_\kappa \text{ for all } \kappa < \alpha\}.$$

For $\alpha \in \mathbb{N}_+$, we simply have the ordinary Cartesian product, i.e., $\prod_{\kappa<\alpha} X_\kappa = X_0 \times X_1 \times \ldots \times X_{\alpha-1}$, and one canonically gets an EMS

$$\mathcal{X}_0 \times \ldots \times \mathcal{X}_{\alpha-1} = (X_0 \times X_1 \times \ldots \times X_{\alpha-1}, d_{\max}, \widehat{S})$$

defined by the component spaces:

$$d_{\max}(\,(x_0,\ldots,x_{\alpha-1}),(x'_0,\ldots,x'_{\alpha-1})\,) = \max_{\kappa=0}^{\alpha-1} d_\kappa(x_\kappa, x'_\kappa),$$
$$\widehat{S} = (\,(s_{0,\pi_1^\alpha(i)},\ldots,s_{\alpha-1,\pi_\alpha^\alpha(i)}) : i \in \mathbb{N}\,)$$

if $S_\kappa = (s_{\kappa,n})_{n\in\mathbb{N}}$ and the π_κ^α denote the projections corresponding to the Cantor numbering of α-tuples ($1 \le \kappa \le \alpha$).

For $\alpha = \omega$ and SEMSs $\mathcal{X}_\kappa$, there is also a canonical way to define a SEMS

$$\prod_{\kappa<\omega} \mathcal{X}_\kappa = (\,\prod_{\kappa<\omega} X_\kappa, \widetilde{d}, \widetilde{S}\,) :$$
$$\widetilde{d}(\,(x_\kappa)_{\kappa\in\mathbb{N}},(x'_\kappa)_{\kappa\in\mathbb{N}}\,) = \sum\nolimits_{\kappa\in\mathbb{N}} 2^{-\kappa}\frac{d_\kappa(x_\kappa,x'_\kappa)}{1+d_\kappa(x_\kappa,x'_\kappa)},$$

and $\widetilde{S}$ enumerates the sequences of skeleton points which are equal to $S_\kappa(0)$, for all $\kappa \ge \kappa_0$, with some $\kappa_0 \in \mathbb{N}$. For details, we refer to [1].

Even if the definition of approximate computability of partial functions of form $f : \prod_{\kappa<\alpha} X_\kappa \rightarrowtail \prod_{\lambda<\beta} Y_\lambda$ could equivalently be reduced to computability w.r.t. these canonically obtained product spaces, we prefer the following explicit version of definition which directly refers to the component spaces.

Let $\boldsymbol{x} = (x_0, x_1, \ldots) \in \prod_{\kappa<\alpha} X_\kappa$. To compute $f(\boldsymbol{x})$, an OTM $\mathcal{M}$ has access to an arbitrary oracle $\boldsymbol{\sigma} = ((i_{0,\mu})_{\mu\in\mathbb{N}}, (i_{0,\mu})_{\mu\in\mathbb{N}}, \ldots) \in \mathbb{F}^\alpha$. The oracle queries have to be of the form $\langle\kappa,\mu\rangle$, $\kappa < \alpha$, $\mu \in \mathbb{N}$, and they are answered by the indices $i_{\kappa,\mu}$. The inputs under consideration are Cantor numbers of pairs, $\langle\lambda,\nu\rangle$, for $\lambda < \beta$ and $\nu \in \mathbb{N}$. The related outputs of the machine, $\mathcal{M}^{\boldsymbol{\sigma}}\langle\lambda,\nu\rangle$, if they exist, have to be indices $j_{\lambda,\nu}$ yielding a sequence $(j_{\lambda,\nu})_{\nu\in\mathbb{N}} \in \mathrm{CF}_{y_\lambda}^{(S'_\lambda)}$ if $f(\boldsymbol{x}) = (y_0, y_1, \ldots)$.

Fig. 1 illustrates this concept which we are going to define precisely now.

Definition 2. *For arities $\alpha, \beta \in \mathbb{N}_+ \cup \{\omega\}$ and EMSs $\mathcal{X}_\kappa = (X_\kappa, d_\kappa, S_\kappa)$ and $\mathcal{Y}_\lambda = (Y_\lambda, d'_\lambda, S'_\lambda)$, a function $f : \prod_{\kappa<\alpha} X_\kappa \rightarrowtail \prod_{\lambda<\beta} Y_\lambda$ is said to be (approximately) computable iff there is an OTM $\mathcal{M}$ such that*

1. *for all* $\boldsymbol{x} = (x_0, x_1, \ldots) \in \mathrm{dom}(f)$, *all* $\boldsymbol{\sigma} \in \prod_{\kappa<\alpha} \mathrm{CF}_{x_\kappa}^{(S_\kappa)}$ *and all* $\lambda < \beta$, $\nu \in \mathbb{N}$, *the outputs* $\mathcal{M}^{\boldsymbol{\sigma}}\langle\lambda,\nu\rangle$ *always exist and fulfill the condition* $((\mathcal{M}^{\boldsymbol{\sigma}}\langle 0,\nu\rangle)_{\nu\in\mathbb{N}}, (\mathcal{M}^{\boldsymbol{\sigma}}\langle 1,\nu\rangle)_{\nu\in\mathbb{N}}, \ldots) \in \prod_{\lambda<\beta} \mathrm{CF}_{y_\lambda}^{(S'_\lambda)}$ *if* $f(\boldsymbol{x}) = (y_0, y_1, \ldots)$;
2. *for all* $\boldsymbol{x} = (x_0, x_1, \ldots) \notin \mathrm{dom}(f)$ *and all* $\boldsymbol{\sigma} \in \prod_{\kappa<\alpha} \mathrm{CF}_{x_\kappa}^{(S_\kappa)}$, *there is an input* $\langle\lambda,\nu\rangle$ *with* $\lambda < \beta$, $\nu \in \mathbb{N}$, *for which* $\mathcal{M}^{\boldsymbol{\sigma}}\langle\lambda,\nu\rangle$ *is undefined.*

For $\alpha = \beta = 1$, the product sets $\prod_{\kappa<\alpha} X_\kappa$ and $\prod_{\lambda<\beta} Y_\lambda$ are identified with X_0 and Y_0, respectively, and the notion of computability of partial functions $f : \prod_{\kappa<\alpha} X_\kappa \rightarrowtail \prod_{\lambda<\beta} Y_\lambda$ according to Def. 2 coincides with the original one for $f : X_0 \rightarrowtail Y_0$ w.r.t. the EMSs $\mathcal{X}_0$ and $\mathcal{Y}_0$.

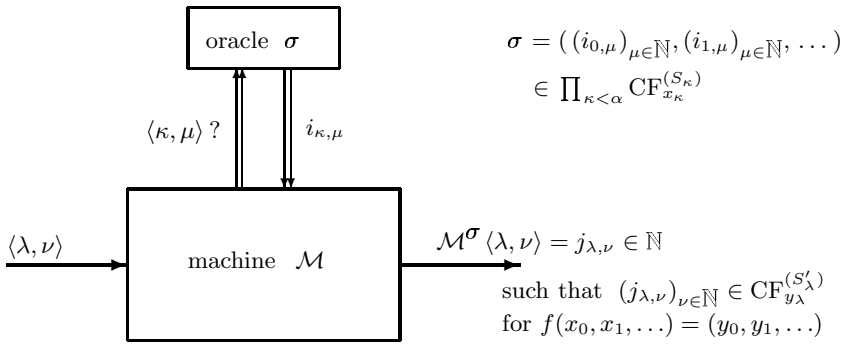


Fig. 1. Computation of a function $f : \prod_{\kappa<\alpha} X_\kappa \rightarrowtail \prod_{\lambda<\beta} Y_\lambda$ by an OTM $\mathcal{M}$.

As usual, one shows that approximate computability is closed under the composition of functions and that computable functions between EMSs are continuous.

Lemma 1. *Let* $f : \prod_{\kappa<\alpha} X_\kappa \rightarrowtail \prod_{\lambda<\beta} Y_\lambda$ *and* $g : \prod_{\lambda<\beta} Y_\lambda \rightarrowtail \prod_{\mu<\gamma} Z_\mu$ *be computable, for related EMSs* $\mathcal{X}_\kappa$, $\mathcal{Y}_\lambda$ *and* $\mathcal{Z}_\mu$. *Then* $g \circ f$ *is computable too. If* $f : X \rightarrowtail Y$ *is computable with respect to the EMSs* $\mathcal{X}$ *and* $\mathcal{Y}$ *over* X *and* Y, *respectively, then* f *is continuous on its domain.* □

3 Representations and Computability

In the previous section, approximate computability has been introduced without speaking about representations explicitly. This just gives the opportunity to investigate the representations of EMSs from the computability point of view, as it has been demonstrated for the space of real numbers in [3].

Definition 3. *Let* $\mathcal{X} = (X, d, S)$ *be an EMS. By a representation of* $\mathcal{X}$, *we mean a surjective partial function* $\varrho : \mathbb{A} \rightarrowtail X$, *where* $\mathbb{A} \in \{\mathbb{B}, \mathbb{F}\}$.

The normed Cauchy representation of $\mathcal{X}$ *is defined as* ${\varrho_{\mathrm{nC}}}^{\mathcal{X}} : \mathbb{F} \rightarrowtail X$ *with* $\mathrm{dom}({\varrho_{\mathrm{nC}}}^{\mathcal{X}}) = \bigcup_{x\in X} \mathrm{CF}_x^{(S)}$ *and* ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}(\sigma) = x$ *iff* $\sigma \in \mathrm{CF}_x^{(S)}$.

Representations ϱ are applied in order to use the sequences from $\varrho^{-1}(x)$ as names for the elements $x \in X$, for example in order to reduce computability on X to computability on the name spaces $\mathbb{B}$ or $\mathbb{F}$. So an element $x \in X$ is said to be ϱ-*computable* iff its name set, $\varrho^{-1}(x)$, contains a computable sequence.

Definition 4. *Let* $\varrho : \mathbb{A} \rightarrowtail X$ *and* $\varrho' : \mathbb{A}' \rightarrowtail Y$ *be representations of EMSs* $\mathcal{X}$ *and* $\mathcal{Y}$*, respectively. A function* $f : X \rightarrowtail Y$ *is said to be* (ϱ, ϱ')*-computable iff there is a function* $g : \mathbb{A} \rightarrowtail \mathbb{A}'$ *computable with respect to the underlying EMSs* $\mathbb{B}$ *or* $\mathbb{F}$ *according to Def. 2 such that* $f \circ \varrho(\sigma) = \varrho' \circ g(\sigma)$*, for all* $\sigma \in \mathrm{dom}(f \circ \varrho)$*, and* $\mathrm{dom}(f \circ \varrho) = \mathrm{dom}(g) \cap \mathrm{dom}(\varrho)$.

Let $\alpha, \beta \in \mathbb{N}_+ \cup \{\omega\}$*, and* $\mathcal{X}_\kappa = (X_\kappa, d_\kappa, S_\kappa)$ *and* $\mathcal{Y}_\lambda = (Y_\lambda, d'_\lambda, S'_\lambda)$ *be EMSs as in Def. 2. For representations* $\varrho_\kappa : \mathbb{A}_\kappa \rightarrowtail X_\kappa$ *and* $\varrho'_\lambda : \mathbb{A}'_\lambda \rightarrowtail Y_\lambda$*, a function* $f : \prod_{\kappa<\alpha} X_\kappa \rightarrowtail \prod_{\lambda<\beta} Y_\lambda$ *is called* $(\prod_{\kappa<\alpha} \varrho_\kappa, \prod_{\lambda<\beta} \varrho'_\lambda)$*-computable iff there is a computable function* $g : \prod_{\kappa<\alpha} \mathbb{A}_\kappa \rightarrowtail \prod_{\lambda<\beta} \mathbb{A}'_\lambda$ *such that* $f \circ (\prod_{\kappa<\alpha} \varrho_\kappa)((\sigma_\kappa)_{\kappa<\alpha}) = (\prod_{\lambda<\beta} \varrho'_\lambda) \circ g((\sigma_\kappa)_{\kappa<\alpha})$*, for all sequences* $(\sigma_\kappa)_{\kappa<\alpha} \in \mathrm{dom}(f \circ (\prod_{\kappa<\alpha} \varrho_\kappa))$*, and* $\mathrm{dom}(f \circ (\prod_{\kappa<\alpha} \varrho_\kappa)) = \mathrm{dom}(g) \cap \mathrm{dom}(\prod_{\kappa<\alpha} \varrho_\kappa)$.

Herein the products of mappings like $\prod_{\kappa<\alpha} \varrho_\kappa$ and $\prod_{\lambda<\beta} \varrho'_\lambda$ are straightforwardly understood: they have to be applied component-wise to tuples or sequences of related lengths, their values have the same lengths and are defined just if the values in all components exist. In the related way, notations like $\varrho_1 \times \varrho_2$ or ϱ^α will be used.

We remark that our definition of computability w.r.t. representations corresponds to the stronger variant of this notion according to [9,1]. This relates better to partial functions than the weaker version preferably used in type two computability, see [4,11]. It is essential for the proof of the following result.

Proposition 1. *For* $\alpha, \beta \in \mathbb{N}_+ \cup \{\omega\}$*, and EMSs* $\mathcal{X}_\kappa$ *and* $\mathcal{Y}_\lambda$ *as in Def. 2, a function* $f : \prod_{\kappa<\alpha} X_\kappa \rightarrowtail \prod_{\lambda<\beta} Y_\lambda$ *is computable in the sense of Def. 2 iff it is* $(\prod_{\kappa<\alpha} {\varrho_{\mathrm{nC}}}^{\mathcal{X}_\kappa}, \prod_{\lambda<\beta} {\varrho_{\mathrm{nC}}}^{\mathcal{Y}_\lambda})$*-computable.*

Proof. We give a sketch of proof for $\alpha = \beta = 1$, $\mathcal{X}_0 = \mathcal{X} = (X, d, S)$ and $\mathcal{Y}_0 = \mathcal{Y} = (Y, d', S')$.

First let the OTM $\mathcal{M}_f$ compute the function $f : X \rightarrowtail Y$ according to Def. 2. In order to compute a function $g : \mathbb{F} \rightarrowtail \mathbb{F}$ which proves that the function f is $({\varrho_{\mathrm{nC}}}^{\mathcal{X}}, {\varrho_{\mathrm{nC}}}^{\mathcal{Y}})$-computable, an OTM $\mathcal{M}_g$ can work as follows on inputs n and oracle sequences $\sigma' \in \mathrm{CF}_\sigma^{(S_\mathbb{F})}$ with $\sigma \in ({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^{-1}(x) = \mathrm{CF}_x^{(S)}$, $x \in X$: By computing $\mathcal{M}_f^\sigma(m)$, for $m = 0, 1, \ldots, n'$ with sufficiently great n', an initial part of some $\sigma'' \in \mathrm{CF}_{f(x)}^{(S')}$ is obtained which is sufficiently large to compute the element $\sigma'''(n)$, for some sequence $\sigma''' \in \mathrm{CF}_{\sigma''}^{(S_\mathbb{F})} = ({\varrho_{\mathrm{nC}}}^{\mathcal{Y}})^{-1}(f(x))$. Now put $g(\sigma') = \sigma'''$, which is defined just if $\mathcal{M}_f^\sigma(m)$ exists for all $m \in \mathbb{N}$, i.e., iff $x \in \mathrm{dom}(f)$. Thus, ${\varrho_{\mathrm{nC}}}^{\mathcal{Y}} \circ g(\sigma) = f \circ {\varrho_{\mathrm{nC}}}^{\mathcal{X}}(\sigma)$ for all $\sigma \in \mathrm{dom}(f \circ {\varrho_{\mathrm{nC}}}^{\mathcal{X}}) = \mathrm{dom}(g) \cap \mathrm{dom}({\varrho_{\mathrm{nC}}}^{\mathcal{X}})$.

Notice that σ can successively be obtained from $\sigma' \in \mathrm{CF}_{\sigma}^{(S_{\mathbb{F}})}$ and $\sigma''' \in \mathrm{CF}_{\sigma''}^{(S_{\mathbb{F}})}$ can be obtained from σ'', since the skeleton $S_{\mathbb{F}}$ is supposed to be effectively computable.

Conversely, let be given an OTM $\mathcal{M}_g$ computing a function $g : \mathbb{F} \rightarrowtail \mathbb{F}$ such that ${\varrho_{\mathrm{nC}}}^{\mathcal{Y}} \circ g(\sigma) = f \circ {\varrho_{\mathrm{nC}}}^{\mathcal{X}}(\sigma)$, for all $\sigma \in \mathrm{dom}(f \circ {\varrho_{\mathrm{nC}}}^{\mathcal{X}})$, and $\mathrm{dom}(f \circ {\varrho_{\mathrm{nC}}}^{\mathcal{X}}) = \mathrm{dom}(g) \cap \mathrm{dom}({\varrho_{\mathrm{nC}}}^{\mathcal{X}})$. Then an OTM $\mathcal{M}_f$ computing f on input n and oracle $\sigma \in \mathrm{CF}_{\sigma}^{(S_{\mathbb{F}})}$ can compute $\mathcal{M}_g^{\sigma'}(m)$, for all $m = 0, 1, \ldots, n'$, with a sufficiently great n', which yields a related initial part of $\sigma''' \in \mathrm{CF}_{\sigma''}^{(S_{\mathbb{F}})}$, for $\sigma'' \in \mathrm{CF}_{f(x)}^{(S')}$, just if $f(x)$ exists. Finally, it can output the element $\sigma''(n)$. □

One easily shows

Lemma 2. *Computability w.r.t. representations is closed under the composition of functions: if $f_1 : \mathcal{X}_1 \rightarrowtail \mathcal{X}_2$ is (ϱ_1, ϱ_2)-computable and $f_2 : \mathcal{X}_2 \rightarrowtail \mathcal{X}_3$ is (ϱ_2, ϱ_3)-computable w.r.t. EMSs $\mathcal{X}_i = (X_i, d_i, S_i)$ and their representations ϱ_i $(i = 1, 2, 3)$, then $f_2 \circ f_1$ is (ϱ_1, ϱ_3)-computable.*
The analogue holds for functions between products of EMSs. □

Now let ϱ and ϱ' be representations of an EMS $\mathcal{X} = (X, d, S)$. To require that any (ϱ_1, ϱ)-computable function $f : X_1 \rightarrowtail X$, for any representation ϱ_1 of an arbitrary EMS $\mathcal{X}_1 = (X_1, d_1, S_1)$, is also (ϱ_1, ϱ')-computable, is equivalent to the condition that id_X is (ϱ, ϱ')-computable. On the other hand, any (ϱ, ϱ_1)-computable function $f : X \rightarrowtail X_1$ is also (ϱ', ϱ_1)-computable iff id_X is (ϱ', ϱ)-computable. This leads to the following concepts.

Definition 5. *For representations ϱ and ϱ' of an EMS $\mathcal{X} = (X, d, S)$, ϱ is said to be reducible to ϱ' (briefly: $\varrho \leq \varrho'$) iff id_X is (ϱ, ϱ')-computable. ϱ and ϱ' are called equivalent (briefly: $\varrho \equiv \varrho'$) iff both $\varrho \leq \varrho'$ and $\varrho' \leq \varrho$.*

It should be noticed that, for representations $\varrho : \mathbb{A} \rightarrowtail X$, $\varrho' : \mathbb{A}' \rightarrowtail X$, $\varrho \leq \varrho'$ simply means that there is a computable function $g : \mathbb{A} \rightarrowtail \mathbb{A}'$ satisfying

$$\varrho(\sigma) = \varrho' \circ g(\sigma), \quad \text{for all } \sigma \in \mathrm{dom}(\varrho).$$

Lemma 2 shows that computability w.r.t. representations for functions between products of (carrier sets of) EMSs remains valid if representations ϱ_λ of image spaces are replaced by representations ϱ'_λ with $\varrho_\lambda \leq \varrho'_\lambda$. Analogously, representations ϱ_κ of preimage spaces can be replaced by ϱ'_κ if $\varrho'_\kappa \leq \varrho_\kappa$. In particular, equivalent representations can be replaced each by the other without changing the validity of computability assertions.

Since the normed Cauchy representations are the most natural ones from the viewpoint of approximate computability, cf. Proposition 1, it becomes a crucial problem to characterize just those representations which are equivalent to ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$.

Definition 6. *A representation ϱ of an EMS $\mathcal{X}$ is said to be a standard representation iff $\varrho \equiv {\varrho_{\mathrm{nC}}}^{\mathcal{X}}$.*

First we show that the characterization of the standard representations of the space of real numbers which has been obtained in [3] can be generalized to arbitrary EMSs. To this purpose, we have to recall the notion of computability for relations between EMSs.

Definition 7. *For EMSs $\mathcal{X} = (X, d, S)$ and $\mathcal{Y} = (Y, d', S')$, a relation $R \subseteq X \times Y$ is said to be computable iff there is an OTM $\mathcal{M}$ such that*

1. *for all $x \in \mathrm{dom}(R)$ and all index sequences $\sigma \in \mathrm{CF}_x^{(S)}$, $\mathcal{M}^\sigma(n)$ always exists and it holds $(\mathcal{M}^\sigma(n))_{n\in\mathbb{N}} \in \mathrm{CF}_y^{(S')}$, for some y with $(x, y) \in R$;*
2. *for all $(x, y) \in R$ there is a sequence $\sigma \in \mathrm{CF}_x^{(S)}$ such that $(\mathcal{M}^\sigma(n))_{n\in\mathbb{N}} \in \mathrm{CF}_y^{(S')}$;*
3. *for all $x \notin \mathrm{dom}(R)$ and all $\sigma \in \mathrm{CF}_x^{(S)}$, there is an input $n \in \mathbb{N}$ for which $\mathcal{M}^\sigma(n)$ is undefined.*

If R is a function, i.e., for any $x \in X$ there is at most one y with $(x, y) \in R$, then it is computable according to Def. 7 iff it is computable in the sense of Def. 2. Unfortunately, for computable relations $R \subseteq X \times Y$ and $Q \subseteq Y \times Z$, the composition $Q \circ R = \{(x, z) :$ there is an $y \in Y$ s.t. $(x, y) \in R$ and $(y, z) \in Q\}$ is not necessarily computable.

By an *inversion* of a relation $R \subseteq X \times Y$, we understand a right–inverse with respect to composition, i.e., a relation $R^{\leftarrow} \subseteq Y \times X$ such that $R \circ R^{\leftarrow} = \mathrm{id}_Y$. It follows that $\mathrm{dom}(R^{\leftarrow}) = \mathrm{ran}(R) = Y$.

Lemma 3. *For any EMS $\mathcal{X}$, ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ is computable with respect to $\mathbb{F}$ and $\mathcal{X}$, and there is an inversion $({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^{\leftarrow}$ of ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ which is computable as relation from $\mathcal{X}$ to $\mathbb{F}$.*

Proof. To compute ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$, let an OTM $\mathcal{M}$ produce the output sequence of the form $(\mathcal{M}^\sigma(n))_{n\in\mathbb{N}} = \sigma' \in \mathrm{CF}_x^{(S)}$, for any oracle sequence $\sigma \in \mathrm{CF}_{\sigma'}^{(S_{\mathbb{F}})}$. This is possible since the skeleton $S_{\mathbb{F}}$ is supposed to be effectively computable. The only difficulty is to guarantee that, for all oracles $\sigma \in \mathbb{F} \setminus \bigcup\{\mathrm{CF}_{\sigma'}^{(S_{\mathbb{F}})} : \sigma' \in \mathrm{CF}_x^{(S)}$ for some $x \in X\}$, $\mathcal{M}^\sigma(n)$ remains undefined for at least one input n. To this purpose, on any input n, $\mathcal{M}$ ensures additionally that $(\sigma(k))_{k\le n}$ is the initial part of a normed Cauchy sequence of some sequence from $\mathbb{F}$, by checking if $\bigcap_{k=0}^{n} \mathrm{Ball}^{\mathbb{F}}_{2^{-k}}(S_{\mathbb{F}}(\sigma(k))) \neq \emptyset$. This non-emptiness holds iff there is a number $m_n \in \mathbb{N}$ satisfying $d_{\mathbb{F}}(S_{\mathbb{F}}(m_n), S_{\mathbb{F}}(\sigma(k))) < 2^{-k}$ for all $k \in \{0, 1, \ldots, n\}$. Using the effective enumeration of the set $D_<$ for the EMS $\mathbb{F}$, $\mathcal{M}$ can confirm this. Otherwise, $\mathcal{M}^\sigma(n)$ remains undefined.

To compute an inversion $({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^{\leftarrow}$ of ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$, an OTM $\mathcal{M}'$ has simply to generate an index sequence $(\mathcal{M}'^{\sigma'}(n))_{n\in\mathbb{N}} = \sigma \in \mathrm{CF}_{\sigma'}^{(S_{\mathbb{F}})}$, for an oracle $\sigma' \in \mathrm{CF}_x^{(S)}$, $x \in X$. □

With a slight modification, the properties of Lemma 3 characterize the standard representations.

Proposition 2. *For every representation $\varrho : \mathbb{A} \rightarrowtail X$ of an EMS $\mathcal{X}$, it holds:*

i) $\varrho \leq {\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ iff there is an extension $\overline{\varrho} \supseteq \varrho$ which is a computable function (from $\mathbb{A}$ to $\mathcal{X}$);
ii) ${\varrho_{\mathrm{nC}}}^{\mathcal{X}} \leq \varrho$ iff there is an inversion $\varrho^{\leftarrow}$ of ϱ which is a computable relation (from $\mathcal{X}$ to $\mathbb{A}$).

This has been proved for $\mathcal{X} = \mathbb{R}$ in [3], Proposition 7. Since the proof given there doesn't use any particular property of $\mathbb{R}$, it applies to the general assertion with only slight modifications. So it is omitted here. □

It immediately follows

Corollary 1. *A representation ϱ of an EMS $\mathcal{X}$ is a standard representation iff there are both an extension $\overline{\varrho} \supseteq \varrho$ computable as function and an inversion $\varrho^{\leftarrow}$ of ϱ which is computable as relation.* □

In [3] we have discussed the possibility to restrict the notion of standard representation in such a way that they are required to be computable functions. Then there would be only $\aleph_0$ standard representations of any $\mathcal{X}$, whereas in the more general sense applied so far there are $2^{2^{\aleph_0}}$ standard representations if the EMS $\mathcal{X}$ contains an accumulation point.
By means of Lemma 1, Proposition 2 also yields

Corollary 2. *If $\varrho \leq {\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ for a representation ϱ of $\mathcal{X}$, then ϱ is continuous on its domain.* □

As a main tool in the proof of Proposition 2, the existence of so-called *funnel machines* for the EMSs $\mathbb{A} \in \{\mathbb{F}, \mathbb{B}\}$ is used. These are OTMs $\mathcal{M}$ computing the identical functions $\mathrm{id}_{\mathbb{A}}$ in such a way that, for any element $\sigma \in \mathbb{A}$, only one distinguished output sequence $(\mathcal{M}^{\sigma'}(n))_{n\in\mathbb{N}}$ occurs, independently on the oracle $\sigma' \in \mathrm{CF}_{\sigma}^{(S_{\mathbb{A}})}$. One can prove that an EMS admits a funnel machine iff it has an injective standard representation. Here we only show

Lemma 4. *The identical functions $\mathrm{id}_{\mathbb{F}}$ and $\mathrm{id}_{\mathbb{B}}$ are standard representations of the EMS $\mathbb{F}$ and $\mathbb{B}$, respectively.*

Proof. Let $\mathbb{A} \in \{\mathbb{F}, \mathbb{B}\}$. To reduce $\mathrm{id}_{\mathbb{A}}$ to ${\varrho_{\mathrm{nC}}}^{\mathbb{A}}$, an OTM $\mathcal{M}$ has to produce an output sequence $(\mathcal{M}^{\sigma'}(n))_{n\in\mathbb{N}} = \sigma'' \in \mathrm{CF}_{\sigma}^{(S_{\mathbb{F}})}$, for any $\sigma' \in \mathrm{CF}_{\sigma}^{(S_{\mathbb{A}})}$, $\sigma \in \mathbb{A}$. To confirm that ${\varrho_{\mathrm{nC}}}^{\mathbb{A}} \leq \mathrm{id}_{\mathbb{A}}$, an OTM $\mathcal{M}'$ has to produce $(\mathcal{M}'^{\sigma'}(n))_{n\in\mathbb{N}} = \sigma'' \in \mathrm{CF}_{\sigma}^{(S_{\mathbb{A}})}$, for any $\sigma' \in \mathrm{CF}_{\sigma}^{(S_{\mathbb{F}})}$, $\sigma \in \mathbb{F}$. Both these transformations are possible because of the effectiveness of the skeletons $S_{\mathbb{F}}$ and $S_{\mathbb{A}}$. □

Thus, a function $f : \mathbb{F} \rightarrowtail \mathbb{F}$ is computable according to Def. 2 iff it is $(\mathrm{id}_{\mathbb{F}}, \mathrm{id}_{\mathbb{F}})$-computable, and analogously for $\mathbb{B}$. Quite similarly one sees that approximate computability of functions $f : \mathbb{F} \rightarrowtail \mathbb{F}$ could equivalently be defined by OTMs getting the elements $\sigma \in \mathbb{F}$ themselves as oracle sequences and putting out the function values directly, i.e., $(\mathcal{M}^{\sigma}(n))_{n\in\mathbb{N}} = f(\sigma)$. In this direct way, the

notion of computability in the name spaces $\mathbb{F}$ and $\mathbb{B}$ is usually defined in type two theory of effectivity, and computability over $\mathbb{R}$ or other EMSs is reduced to that over $\mathbb{F}$ or $\mathbb{B}$ via representations then, cf. [9,11]. In the present approach, we prefer our slightly more complicated definition in order to get a notion of approximate computability which is uniformly applicable to arbitrary EMSs.

By Corollary 1, we obtain

Corollary 3. *An injective representation ϱ of an EMS is a standard representation iff both ϱ and its inverse ϱ^{-1} are computable functions.*

Proof. If an injective representation ϱ is standard, both an extension $\overline{\varrho}$ and the only possible inversion ϱ^{-1} must be computable functions. Then a computation of $\overline{\varrho}$ can easily be restricted to $\mathrm{dom}(\varrho) = \{\sigma : \varrho^{-1} \circ \overline{\varrho}(\sigma) = \sigma\}$. The converse direction follows immediately. □

Since computable functions are continuous, injective standard representations can only exist if the EMS $\mathcal{X}$ is homeomorphic to a subset of $\mathbb{F}$ or $\mathbb{B}$ w.r.t. the restriction of its metric. In particular, $\mathcal{X}$ has to be totally disconnected then. Brattka and Hertling [2] have shown that the set of elements $x \in X$ that have infinite name sets $\varrho^{-1}(x)$, with respect to a standard representation ϱ, is fat and dense in (X, d) if it is a locally connected perfect Polish space. This holds by topological reasons, even for all so-called admissible representations ϱ.

4 Indicator Functions for Standard Representations

Now we are going to deal with the problem to specify a set of functions such that a representation ϱ of an EMS $\mathcal{X}$ is a standard representation if and only if all these functions are computable w.r.t. ϱ. Thus, they are *indicator functions* of the property of representations to be standard.

It would be nice to have a finite set $\{f_1, \ldots, f_n\}$ of indicator functions f_i, each of a finite arity $k_i \in \mathbb{N}_+$ and operating on the carrier X of $\mathcal{X}$. This would define a constant–free, single-sorted algebra over the universe X, $(X; f_1, \ldots, f_n)$, which is *effectively categorical* in the sense of Hertling [4]. This means that it is effective w.r.t. a representation ϱ iff ϱ is a standard representation. Unfortunately, the problem is not solvable in this strong sense. So we have also to accept 0-ary indicator functions, i.e., constants of the structure, and even infinitary functions. Moreover, relations are allowed whose computability w.r.t. representations is defined in a rather special way. Such single-sorted effectively categorical structures characterizing the standard representations will be specified in the next section.

In the present section, as a first attempt, we deal with many-sorted indicator functions. This weak version of the problem is closely related to Brattka's [1] approach to an abstract characterization of computability in structures. He just considered many-sorted structures with operations that can be infinitary and/or even indeterministic.

By the remarks after Def. 5, standard representations have to share all computability properties of the normed Cauchy representations. Thus, in order to

solve the problem, one has to search for some typical functions which are computable w.r.t. ${\varrho_{\rm nC}}^{\mathcal{X}}$.

First we observe that ${\varrho_{\rm nC}}^{\mathcal{X}}$ allows to compute an index sequence $\sigma' \in {\rm CF}_x^{(S)}$, for any $x \in X$: it even holds $({\varrho_{\rm nC}}^{\mathcal{X}})^{-1}(x) = {\rm CF}_x^{(S)}$. Of course, it depends on the given name $\sigma \in ({\varrho_{\rm nC}}^{\mathcal{X}})^{-1}(x)$ which σ' is obtained. Thus, one has also to consider the computability of relations w.r.t. representations.

Definition 8. *Let $\varrho : \mathbb{A} \rightarrowtail X$ and $\varrho' : \mathbb{A}' \rightarrowtail Y$ be representations of EMSs $\mathcal{X}$ and $\mathcal{Y}$, respectively. A relation $R \subseteq X \times Y$ is said to be (ϱ, ϱ')-computable iff there is a relation $Q \subseteq \mathbb{A} \times \mathbb{A}'$ computable with respect to the underlying EMSs $\mathbb{B}$ or $\mathbb{F}$ according to Def. 7 such that $R \circ \varrho(\sigma) = \varrho' \circ Q(\sigma)$, for all $\sigma \in {\rm dom}(R \circ \varrho)$, and $Q({\rm dom}(\varrho)) \subseteq {\rm dom}(\varrho')$.*

Herein let $R \circ \varrho(\sigma)$ denote the set $\{y : (\sigma, y) \in R \circ \varrho\}$, in the same way $\varrho' \circ Q(\sigma)$ stands for a subset of Y. Correspondingly, in equations between values of functions and relations within the following sketch of proof for Proposition 3, functions have to be considered as relations. $Q({\rm dom}(\varrho))$ stands for the set $\{\sigma' : (\sigma, \sigma') \in Q \text{ for some } \sigma \in {\rm dom}(\varrho)\}$. From $Q({\rm dom}(\varrho)) \subseteq {\rm dom}(\varrho')$ and $R \circ \varrho(\sigma) = \varrho' \circ Q(\sigma)$, for all $\sigma \in {\rm dom}(R \circ \varrho)$, it follows ${\rm dom}(R \circ \varrho) = {\rm dom}(Q) \cap {\rm dom}(\varrho)$. Thus, if R and Q in Def. 8 are functions, we have the situation from Def. 4.

Proposition 3. *For any representation $\varrho : \mathbb{A} \rightarrowtail X$ of an EMS $\mathcal{X}$, it holds:*

i) $\varrho \leq {\varrho_{\rm nC}}^{\mathcal{X}}$ iff $({\varrho_{\rm nC}}^{\mathcal{X}})^{-1}$ is $(\varrho, {\rm id}_{\mathbb{F}})$-computable (as relation from $\mathcal{X}$ into $\mathbb{F}$);
ii) ${\varrho_{\rm nC}}^{\mathcal{X}} \leq \varrho$ iff ${\varrho_{\rm nC}}^{\mathcal{X}}$ is $({\rm id}_{\mathbb{F}}, \varrho)$-computable (as function from $\mathbb{F}$ into $\mathcal{X}$).

Proof. We show assertion i), the proof for ii) goes by standard methods.

If $\varrho \leq {\varrho_{\rm nC}}^{\mathcal{X}}$, then $\varrho(\sigma) = {\varrho_{\rm nC}}^{\mathcal{X}} \circ g(\sigma)$, for all $\sigma \in {\rm dom}(\varrho)$, with a computable function $g : \mathbb{A} \rightarrowtail \mathbb{F}$. It follows $({\varrho_{\rm nC}}^{\mathcal{X}})^{-1} \circ \varrho(\sigma) = ({\varrho_{\rm nC}}^{\mathcal{X}})^{-1} \circ {\varrho_{\rm nC}}^{\mathcal{X}} \circ g(\sigma) = {\rm id}_{\mathbb{F}} \circ (({\varrho_{\rm nC}}^{\mathcal{X}})^{-1} \circ {\varrho_{\rm nC}}^{\mathcal{X}} \circ g)(\sigma)$. The relation $Q = ({\varrho_{\rm nC}}^{\mathcal{X}})^{-1} \circ {\varrho_{\rm nC}}^{\mathcal{X}} \circ g \subseteq \mathbb{A} \times \mathbb{F}$ turns out to be computable. $Q({\rm dom}(\varrho)) \subseteq {\rm dom}({\rm id}_{\mathbb{F}}) = \mathbb{F}$ holds trivially.

Conversely, let $({\varrho_{\rm nC}}^{\mathcal{X}})^{-1}$ be $(\varrho, {\rm id}_{\mathbb{F}})$-computable, i.e., $({\varrho_{\rm nC}}^{\mathcal{X}})^{-1} \circ \varrho(\sigma) = {\rm id}_{\mathbb{F}} \circ Q(\sigma)$, for all $\sigma \in {\rm dom}(({\varrho_{\rm nC}}^{\mathcal{X}})^{-1} \circ \varrho) = {\rm dom}(\varrho)$, with a computable relation $Q \subseteq \mathbb{A} \times \mathbb{F}$. Then $\varrho(\sigma) = {\varrho_{\rm nC}}^{\mathcal{X}} \circ ({\varrho_{\rm nC}}^{\mathcal{X}})^{-1} \circ \varrho(\sigma) = {\varrho_{\rm nC}}^{\mathcal{X}} \circ Q(\sigma)$, for all $\sigma \in {\rm dom}(\varrho)$. Unfortunately, Q is not necessarily a function. Using a funnel machine for the space $\mathbb{A}$, however, similarly to the proof of Proposition 2, one gets a function $g \subseteq Q$ such that $\varrho(\sigma) = {\varrho_{\rm nC}}^{\mathcal{X}} \circ g(\sigma)$ for all $\sigma \in {\rm dom}(\varrho)$ too. □

Corollary 4. *A representation ϱ of an EMS $\mathcal{X}$ is a standard representation iff both ${\varrho_{\rm nC}}^{\mathcal{X}}$ is $({\rm id}_{\mathbb{F}}, \varrho)$-computable and $({\varrho_{\rm nC}}^{\mathcal{X}})^{-1}$ is $(\varrho, {\rm id}_{\mathbb{F}})$-computable, the latter as a relation from $\mathcal{X}$ into $\mathbb{F}$.* □

After this counterpart of Proposition 2, we are going to consider operations canonically related to any EMS $\mathcal{X} = (X, d, S)$. Recall that the skeleton $S = (s_n)_{n \in \mathbb{N}}$ is a sequence of points, i.e., it is a function $S : \mathbb{N} \longrightarrow X$. Since the constant sequences are ${\varrho_{\rm nC}}^{\mathcal{X}}$-names of the skeleton points, ${\varrho_{\rm nC}}^{\mathcal{X}}(n, n, n, \ldots) = s_n$,

one easily shows that S is $({\varrho_{\mathrm{nC}}}^{\mathbb{N}}, {\varrho_{\mathrm{nC}}}^{\mathcal{X}})$-computable. Notice that, by our definition of the EMS $\mathbb{N}$, $S_{\mathbb{N}} = \mathrm{id}_{\mathbb{N}}$ and ${\varrho_{\mathrm{nC}}}^{\mathbb{N}}$ is essentially an injective numbering: $({\varrho_{\mathrm{nC}}}^{\mathbb{N}})^{-1}(n) = \{(n, n, n, \ldots)\}$. So the $({\varrho_{\mathrm{nC}}}^{\mathbb{N}}, \varrho)$-computability of the skeleton S is a candidate of a many-sorted indicator condition.

As a second natural indicator condition, we are now going to consider the right–cut computability of the distance function $d : X^2 \longrightarrow \mathbb{R}$. Right–cut computability of real–valued functions can fair be defined via a suitable nonstandard representation $\varrho^{\mathbb{R}}_{\mathrm{rcut}}$ of the reals: let

$$\varrho^{\mathbb{R}}_{\mathrm{rcut}}((k_0, k_1, k_2, \ldots)) = \inf(\{q_{k_i} : i \in \mathbb{N}\})$$

if $(k_0, k_1, k_2, \ldots) \in \mathrm{dom}(\varrho^{\mathbb{R}}_{\mathrm{rcut}}) = \{(k_0, k_1, k_2, \ldots) \in \mathbb{F} : \inf(\{q_{k_i} : i \in \mathbb{N}\}) \in \mathbb{R}\}$. Remember that $\nu_{\mathbb{Q}} = (q_k)_{k\in\mathbb{N}}$ is the fixed standard numbering of $\mathbb{Q}$.

The property that $D_< = \{\langle m, n, k\rangle : d(s_m, s_n) < q_k\}$ is r. e. is just equivalent to the right–cut computability of function d restricted to the points of the skeleton. More precisely, the function $d^0 : \mathbb{N}^2 \longrightarrow \mathbb{R}$, defined by

$$d^0(m, n) = d(s_m, s_n),$$

is $({\varrho_{\mathrm{nC}}}^{\mathbb{N}} \times {\varrho_{\mathrm{nC}}}^{\mathbb{N}}, \varrho^{\mathbb{R}}_{\mathrm{rcut}})$-computable. If $\mathcal{X}$ is even a SEMS, then d^0 is typically $({\varrho_{\mathrm{nC}}}^{\mathbb{N}} \times {\varrho_{\mathrm{nC}}}^{\mathbb{N}}, {\varrho_{\mathrm{nC}}}^{\mathbb{R}})$-computable, cf. [3].

The distance function d is $({\varrho_{\mathrm{nC}}}^{\mathcal{X}} \times {\varrho_{\mathrm{nC}}}^{\mathcal{X}}, \varrho^{\mathbb{R}}_{\mathrm{rcut}})$-computable: for given names $\sigma_j = (n_{j0}, n_{j1}, \ldots) \in ({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^{-1}(x_j)$, $j = 1, 2$, the sequence of distances between the related skeleton points, $(d(s_{n_{1k}}, s_{n_{2k}}))_{k\in\mathbb{N}}$ and the recursive enumerability of $D_<$ (or the right–cut computability of d^0) can be used to generate a sequence from $(\varrho^{\mathbb{R}}_{\mathrm{rcut}})^{-1}(d(x_1, x_2))$. Indeed, it holds $d(x_1, x_2) \leq d(x_1, s_{n_{1k}}) + d(s_{n_{1k}}, s_{n_{2k}}) + d(s_{n_{2k}}, x_2) < d^0(n_{1k}, n_{2k}) + 2 \cdot 2^{-k}$, for all $k \in \mathbb{N}$.

So we have seen that, w.r.t. the normed Cauchy representation, both the skeleton is computable and the distance is right–cut computable. These two conditions of computability, however, give a natural characterization of effectivity of an EMS w.r.t. a representation.

Definition 9. *For a representation ϱ, the EMS $\mathcal{X} = (X, d, S)$ is said to be ϱ-effective iff both the skeleton S is $({\varrho_{\mathrm{nC}}}^{\mathbb{N}}, \varrho)$-computable and the distance d is $(\varrho \times \varrho, \varrho^{\mathbb{R}}_{\mathrm{rcut}})$-computable.*

By Lemma 2 and the remarks above, we have

Lemma 5. *For any standard representation ϱ, the EMS $\mathcal{X}$ is ϱ-effective.* □

On the other hand, if $\mathcal{X}$ is ϱ-effective, ϱ is not necessarily a standard representation. For example, let $\mathcal{X} = \mathbb{R}$ and take as ϱ-names the index sequences which belong to uncomputably fast converging sequences of points of the skeleton, cf. [4]. Then $\mathbb{R}$ is ϱ-effective, but ϱ is not a standard representation.

Lemma 6. *If $\mathcal{X}$ is ϱ-effective, then $\varrho \leq {\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ and ϱ is continuous.*

Proof. Given a ϱ-name σ, by computing S successively and using the right–cut computability of d w.r.t. ϱ, the element $x = \varrho(\sigma)$ can effectively be approximated by a sequence of skeleton points, with a computable rate of convergence. This yields a ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$-name of x. The continuity follows by Proposition 2 i). □

The converse of Lemma 6 doesn't hold: for every restriction $\varrho \subseteq {\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ with $\mathrm{ran}(\varrho) = X$, we have $\varrho \leq {\varrho_{\mathrm{nC}}}^{\mathcal{X}}$. If (X, d) is a perfect Polish space, however, any preimage set $({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^{-1}(x)$, $x \in X$, contains nonrecursive sequences. But if all ϱ-names of a point of the skeleton $x \in \mathrm{ran}(S)$ are nonrecursive, S cannot be $({\varrho_{\mathrm{nC}}}^{\mathbb{N}}, \varrho)$-computable.

Lemmas 5 and 6 yield the following characterization.

Proposition 4. *The standard representations are just the (w.r.t. $\leq$) greatest of all those representations ϱ for which the underlying EMS is ϱ-effective.*

Proof. If ϱ is a standard representation of $\mathcal{X}$, then $\mathcal{X}$ is ϱ-effective by Lemma 5. Moreover, by Lemma 6, for any representation ϱ' such that $\mathcal{X}$ is ϱ'-effective, we have $\varrho' \leq {\varrho_{\mathrm{nC}}}^{\mathcal{X}} \equiv \varrho$. Conversely, if $\mathcal{X}$ is ϱ-effective, then $\varrho \leq {\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ by Lemma 6. If ϱ is even the greatest of all representations which make $\mathcal{X}$ effective, it also holds ${\varrho_{\mathrm{nC}}}^{\mathcal{X}} \leq \varrho$. □

Coming back to the problem of finding a set of indicator functions, we notice that the condition of $(\varrho, \mathrm{id}_{\mathbb{F}})$-computability of the relation $({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^{-1}$ in Corollary 4 can equivalently be replaced by the ϱ-effectivity of $\mathcal{X}$. So we have

Proposition 5. *A representation ϱ of an EMS $\mathcal{X}$ is a standard representation iff both $\mathcal{X}$ is ϱ-effective and ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ is $(\mathrm{id}_{\mathbb{F}}, \varrho)$-computable.* □

Thus, the functions $S : \mathbb{N} \longrightarrow X, d : X^2 \longrightarrow \mathbb{R}$ and ${\varrho_{\mathrm{nC}}}^{\mathcal{X}} : \mathbb{F} \rightarrowtail X$ build a finite set of indicator functions if their computabilities are required in the ways described above. Unfortunately, they are many-sorted and establish relationships between $\mathcal{X}$ and the EMSs $\mathbb{N}$, $\mathbb{R}$ and $\mathbb{F}$, respectively. In the general case of an EMS $\mathcal{X}$, we have even to consider the nonstandard representation $\varrho_{\mathrm{rcut}}^{\mathbb{R}}$ of $\mathbb{R}$. If $\mathcal{X}$ is a SEMS, $\varrho_{\mathrm{rcut}}^{\mathbb{R}}$ can equivalently be replaced by ${\varrho_{\mathrm{nC}}}^{\mathbb{R}}$ however.

5 Effectively Categorical Single-Sorted Structures

In their axiomatic approach to computability structures over Banach spaces, Pour–El and Richards [8] required the computability of limits of computable double sequences. This relates to Hertling's [4] and Brattka's [1] use of the function NormLim as an indicator function for the real number space and recursive metric spaces, respectively.

Over an arbitrary EMS $\mathcal{X} = (X, d, S)$, NormLim is defined as an infinitary partial function, $\mathrm{NormLim} : X^{\omega} \rightarrowtail X$, such that

$$\mathrm{dom}(\mathrm{NormLim}) = \{(x_n)_{n\in\mathbb{N}} : d(x_m, x_n) < 2^{-\min(m,n)}\},$$
$$\mathrm{NormLim}((x_n)_{n\in\mathbb{N}}) = \lim_{n\to\infty} x_n \quad \text{if} \quad (x_n)_{n\in\mathbb{N}} \in \mathrm{dom}(\mathrm{NormLim}).$$

Lemma 7. *For any EMS* $\mathcal{X}$, NormLim *is* $((\varrho_{\mathrm{nC}}{}^{\mathcal{X}})^{\omega}, \varrho_{\mathrm{nC}}{}^{\mathcal{X}})$*-computable. If, for some representation* ϱ, $\mathcal{X}$ *is* ϱ*-effective and* NormLim *is* $(\varrho^{\omega}, \varrho)$*-computable, then* $\varrho_{\mathrm{nC}}{}^{\mathcal{X}}$ *is* $(\mathrm{id}_{\mathbb{F}}, \varrho)$*-computable.*

Proof. The $((\varrho_{\mathrm{nC}}{}^{\mathcal{X}})^{\omega}, \varrho_{\mathrm{nC}}{}^{\mathcal{X}})$-computability of NormLim is easily shown. An OTM can successively confirm that the oracle sequence $\boldsymbol{\sigma} = (\sigma_0, \sigma_1, \ldots)$ belongs to dom(NormLim), and it can generate a $\varrho_{\mathrm{nC}}{}^{\mathcal{X}}$-name of $\lim_{n\to\infty} x_n$ for $\boldsymbol{\sigma} \in \prod_{n\in\mathbb{N}} \mathrm{CF}_{x_n}^{(S)}$.

Now let $\mathcal{X}$ be ϱ-effective and NormLim be $(\varrho^{\omega}, \varrho)$-computable. Given $\sigma' \in \mathrm{CF}_{\sigma}^{S\mathbb{F}}$, $\sigma \in (\varrho_{\mathrm{nC}}{}^{\mathcal{X}})^{-1}(x)$, $x \in X$, an OTM can successively generate a sequence of ϱ-names of a sequence $(x_n)_{n\in\mathbb{N}}$ such that $x = \mathrm{NormLim}((x_n)_{n\in\mathbb{N}})$. This is done by means of the $(\varrho_{\mathrm{nC}}{}^{\mathbb{N}}, \varrho)$-computability of S and the $(\varrho\times\varrho, \varrho_{\mathrm{rcut}}^{\mathbb{R}})$-computability of d. Then, using the $(\varrho^{\omega}, \varrho)$-computability of NormLim, a ϱ-name of x can be generated. □

By means of Proposition 5, it follows

Proposition 6. *A representation* ϱ *of an EMS* $\mathcal{X}$ *is a standard representation iff both* $\mathcal{X}$ *is* ϱ*-effective and* NormLim *is* $(\varrho^{\omega}, \varrho)$*-computable.* □

So instead of the many-sorted function $\varrho_{\mathrm{nC}}{}^{\mathcal{X}} : \mathbb{F} \rightarrowtail X$, one can use the function NormLim operating only on X as an indicator function. It just expresses the ϱ-effective completeness of $\mathcal{X}$ in a natural way. Unfortunately, it is an infinitary function. In Section 6, we shall however see that finitary functions, also together with constants and relations in the universe, are not sufficient to characterize the standard representations.

Over the SEMS $\mathbb{R}$ of the real numbers, the $(\varrho_{\mathrm{nC}}{}^{\mathbb{N}}, \varrho)$-computability of the skeleton $S_{\mathbb{R}} = (q_k)_{k\in\mathbb{N}}$ can be replaced by requirements concerning pure ϱ-computabilities. If the number 1 is ϱ-computable and the arithmetical basic operations $+, -, \cdot, /$ are $(\varrho\times\varrho, \varrho)$-computable, then $S_{\mathbb{R}}$ is $(\varrho_{\mathrm{nC}}{}^{\mathbb{N}}, \varrho)$-computable. Conversely, any standard representation ϱ has to fulfill these requirements. Moreover, the values $d_{\mathbb{R}}(x, y)$ are computable w.r.t. $\varrho_{\mathrm{nC}}{}^{\mathbb{R}}$ from ϱ-names of the arguments x, y. So we have

Proposition 7. *A representation* ϱ *of the real numbers is a standard representation iff the number* 1 *is* ϱ*-computable, the basic operations* $+, -, \cdot, /$ *are* $(\varrho\times\varrho, \varrho)$*-computable, the distance function* $d_{\mathbb{R}}$ *is* $(\varrho\times\varrho, \varrho_{\mathrm{nC}}{}^{\mathbb{R}})$*-computable, and* NormLim *is* $(\varrho^{\omega}, \varrho)$*-computable.* □

The proof shows that the number 1 and the functions $+, -, \cdot, /$ as indicator functions in Proposition 7 can equivalently be replaced by any set of (w.r.t. $\varrho_{\mathrm{nC}}{}^{\mathbb{R}}$) computable numbers and functions sufficient to generate all the points of a skeleton equivalent to $S_{\mathbb{R}}$ (cf. [3] for this equivalence notion). For example, the number 1 and the operations $+, -, \cdot\frac{1}{2}$, the latter considered as a unary function, would yield the equivalent skeleton of dyadic numbers.

Corollary 5. *A representation ϱ of the real numbers is a standard representation iff the number 1 is ϱ-computable, the basic operations $+$ and $-$ are $(\varrho \times \varrho, \varrho)$-computable, the unary function $\cdot\frac{1}{2}$ is (ϱ, ϱ)-computable, the distance $d_{\mathbb{R}}$ is $(\varrho \times \varrho, {\varrho_{\mathrm{nC}}}^{\mathbb{R}})$-computable, and* NormLim *is $(\varrho^{\omega}, \varrho)$-computable.* □

Also in arbitrary EMSs $\mathcal{X} = (X, d, S)$, the condition that the skeleton S is $({\varrho_{\mathrm{nC}}}^{\mathbb{N}}, \varrho)$-computable can often be replaced by computability conditions with respect to ϱ only. This is the case if the existence of an injective embedding

$$\mathrm{emb}_{\mathbb{N}} : \mathbb{N} \longrightarrow X,$$

which is $({\varrho_{\mathrm{nC}}}^{\mathbb{N}}, \varrho)$-computable and whose inverse is $(\varrho, {\varrho_{\mathrm{nC}}}^{\mathbb{N}})$-computable, can be expressed in this way. Indeed, by Lemma 2, then S is $({\varrho_{\mathrm{nC}}}^{\mathbb{N}}, \varrho)$-computable iff $S \circ {\mathrm{emb}_{\mathbb{N}}}^{-1}$ is (ϱ, ϱ)-computable.

We are going to show that, for any EMS which possesses an effective discrete sequence, such a function $\mathrm{emb}_{\mathbb{N}}$ exists.

Definition 10. *By an effectively discrete sequence (briefly: EDS) of an EMS $\mathcal{X} = (X, d, S)$, we mean a pair (η, δ) of recursive total functions $\eta, \delta : \mathbb{N} \longrightarrow \mathbb{N}$ such that $d(s_{\eta(n)}, s_{\eta(n')}) > 2^{-\delta(n)}$, for all $n, n' \in \mathbb{N}$ with $n \neq n'$.*

In [3] we have shown that all SEMSs, which are unbounded or have an accumulation point, possesses an EDS.

Now we suppose that $\mathcal{X}$ has an EDS (η, δ) fixed in the sequel. Then let

$$\mathrm{emb}_{\mathbb{N}} = S \circ \eta.$$

This is an injection of $\mathbb{N}$ into X. To express the needed computability properties w.r.t. ϱ, we define the function $\mathrm{succ} : X \rightarrowtail X$ by $\mathrm{dom}(\mathrm{succ}) = \mathrm{ran}(S \circ \eta)$,

$$\mathrm{succ}(\, S \circ \eta(n)\,) = S \circ \eta\,(n+1), \quad \text{for all } n \in \mathbb{N}.$$

Lemma 8. *Let $\mathcal{X}$ possess an EDS (η, δ), and let the functions* $\mathrm{emb}_{\mathbb{N}}$ *and* succ *be defined as described above. Then, for any representation ϱ of $\mathcal{X}$ such that the distance d is $(\varrho \times \varrho, \varrho_{\mathrm{rcut}}^{\mathbb{R}})$-computable, the following conditions are equivalent:*

1. *$\mathrm{emb}_{\mathbb{N}}$ is $({\varrho_{\mathrm{nC}}}^{\mathbb{N}}, \varrho)$-computable, and its inverse is $(\varrho, {\varrho_{\mathrm{nC}}}^{\mathbb{N}})$-computable;*
2. *both the element $S \circ \eta(0)$ is ϱ-computable, and the function* succ *is (ϱ, ϱ)-computable.*

Proof. If Condition 1. holds, the element $S \circ \eta(0) = \mathrm{emb}_{\mathbb{N}}(0)$ is ϱ-computable, and the function $\mathrm{succ} = \mathrm{emb}_{\mathbb{N}} \circ (+1) \circ {\mathrm{emb}_{\mathbb{N}}}^{-1}$ is (ϱ, ϱ)-computable, where $(+1)$ denotes the successor function on $\mathbb{N}$.

Conversely, let Condition 2. be fulfilled. For the $({\varrho_{\mathrm{nC}}}^{\mathbb{N}}, \varrho)$-computation of $\mathrm{emb}_{\mathbb{N}}(n)$, an OTM can successively generate sufficiently large initial parts of ϱ-names of the elements $\mathrm{emb}_{\mathbb{N}}(0)$, $\mathrm{emb}_{\mathbb{N}}(1), \ldots, \mathrm{emb}_{\mathbb{N}}(n)$.
For the $(\varrho, {\varrho_{\mathrm{nC}}}^{\mathbb{N}})$-computation of ${\mathrm{emb}_{\mathbb{N}}}^{-1}(x)$, let an OTM first search for a

number $n \in \mathbb{N}$ such that $d(\mathrm{succ}^n(S \circ \eta(0)), x) < 2^{-\delta(n)-1}$. This n is uniquely determined if it exists, and will be found, since d is $(\varrho \times \varrho, \varrho_{\mathrm{rcut}}^{\mathbb{R}})$-computable. Then let the machine successively verify that $x = \mathrm{succ}^n(S \circ \eta(0))$ by confirming that $d(\mathrm{succ}^n(S \circ \eta(0)), x) < 2^{-m}$, for all $m \in \mathbb{N}$. As long as this is possible, it outputs n as an element of the constant sequence $(n, n, \ldots) = ({\varrho_{\mathrm{nC}}}^{\mathbb{N}})^{-1}(n)$; in the case of violation, the machine never halts. □

Proposition 8. *Suppose that the EMS $\mathcal{X}$ possesses an EDS (η, δ) and the functions* $\mathrm{emb}_{\mathbb{N}}$ *and* succ *are defined as above. A representation ϱ of $\mathcal{X}$ is a standard representation of $\mathcal{X}$ iff*

- *the element $S \circ \eta(0)$ is ϱ-computable,*
- *the functions* succ *and* ${\mathrm{emb}_{\mathbb{N}}}^{-1} \circ S$ *are (ϱ, ϱ)-computable,*
- *the distance function d is $(\varrho \times \varrho, \varrho_{\mathrm{rcut}}^{\mathbb{R}})$-computable, and*
- NormLim *is $(\varrho^{\omega}, \varrho)$-computable.*

Proof. If $S \circ \eta(0)$ is ϱ-computable, the functions succ and ${\mathrm{emb}_{\mathbb{N}}}^{-1} \circ S$ are (ϱ, ϱ)-computable and the distance d is $(\varrho \times \varrho, \varrho_{\mathrm{rcut}}^{\mathbb{R}})$-computable, then Condition 1 of Lemma 8 follows, and the skeleton S is $({\varrho_{\mathrm{nC}}}^{\mathbb{N}}, \varrho)$-computable. If ϱ admits the computation of NormLim too, it is a standard representation by Proposition 6. On the other hand, for standard representations ϱ the required computability conditions have to be fulfilled, since in this case Condition 2 of Lemma 8 holds true. □

For a SEMS $\mathcal{X}$, the $(\varrho \times \varrho, \varrho_{\mathrm{rcut}}^{\mathbb{R}})$-computability of the distance d in Proposition 8 can equivalently be replaced by the $(\varrho \times \varrho, {\varrho_{\mathrm{nC}}}^{\mathbb{R}})$-computability. By means of the embedding $\mathrm{emb}_{\mathbb{N}}$, this can be enforced by a computability condition w.r.t. ϱ if the enumerability of relations w.r.t. ϱ is also taken into account.

Definition 11. *For a representation ϱ of an EMS $\mathcal{X}$, a k-ary relation $R \subseteq X^k$, $k \in \mathbb{N}_+$, is said to be enumerable w.r.t. ϱ iff there exists a recursively open set $M \subseteq \mathbb{F}^k$ such that $(\varrho^k)^{-1}(R) \subseteq M$ and $(\varrho^k)^{-1}(X^k \setminus R) \cap M = \emptyset$.*

As usual, by a recursively open set, we mean an effective union of basic open sets, the details are given in [3]. For an equivalent characterization of enumerability w.r.t. a representation, we refer to [4]. For example, the recursive enumerability of $D_>$ implies that, for any SEMS $\mathcal{X}$, the binary inequality relation $\neq$ is enumerable w.r.t. ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$.

One easily shows

Lemma 9. *If $\varrho_1 \leq \varrho_2$ and R is enumerable w.r.t. ϱ_2, for representations ϱ_1, ϱ_2 and a relation R of an EMS $\mathcal{X}$, then R is enumerable w.r.t. ϱ_1 too.* □

Thus, the enumerability w.r.t. ϱ of the inequality $\neq$ turns out to be a necessary condition for the standardness of a representation ϱ.

For a SEMS $\mathcal{X}$ with an EDS (η, δ) and an embedding function $\mathrm{emb}_{\mathbb{N}} = S \circ \eta$, we consider the partial function $d_{\mathrm{ap}} : X^3 \rightarrowtail X$ defined by

$$d_{\mathrm{ap}}(x_1, x_2, x_3) = \begin{cases} \mathrm{emb}_{\mathbb{N}}(0) & \text{if } x_3 \in \mathrm{ran}(\mathrm{emb}_{\mathbb{N}}) \text{ and } d(x_1, x_2) < q_{\mathrm{emb}_{\mathbb{N}}{}^{-1}(x_3)}, \\ \mathrm{emb}_{\mathbb{N}}(1) & \text{if } x_3 \in \mathrm{ran}(\mathrm{emb}_{\mathbb{N}}) \text{ and } d(x_1, x_2) > q_{\mathrm{emb}_{\mathbb{N}}{}^{-1}(x_3)}, \\ \text{undefined} & \text{if } x_3 \notin \mathrm{ran}(\mathrm{emb}_{\mathbb{N}}) \text{ or } d(x_1, x_2) = q_{\mathrm{emb}_{\mathbb{N}}{}^{-1}(x_3)}. \end{cases}$$

Lemma 10. *Let $\mathcal{X}$ be a SEMS with an EDS (η, δ), and $\mathrm{emb}_{\mathbb{N}} = S \circ \eta$. Then the function d_{ap} is (ϱ^3, ϱ)-computable, for any standard representation ϱ of $\mathcal{X}$. Conversely, if the element $\mathrm{emb}_{\mathbb{N}}(0)$ is ϱ-computable, the functions $S \circ \mathrm{emb}_{\mathbb{N}}{}^{-1}$ and d_{ap} are (ϱ, ϱ)-computable resp. (ϱ^3, ϱ)-computable and the inequality relation $\neq$ is enumerable w.r.t. ϱ, then the distance d is $(\varrho \times \varrho, {\varrho_{\mathrm{nC}}}^{\mathbb{R}})$-computable.*

Proof. For the first assertion, we have to show that d_{ap} is $(({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^3, {\varrho_{\mathrm{nC}}}^{\mathcal{X}})$-computable. Let be given ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$-names of elements x_1, x_2, x_3. Then an OTM can try to compute successively both the numbers $k = \mathrm{emb}_{\mathbb{N}}{}^{-1}(x_3)$ and $d(x_1, x_2)$. If $d(x_1, x_2) < q_k$ or $d(x_1, x_2) > q_k$, this is realized at some time and a ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$-name of $\mathrm{emb}_{\mathbb{N}}(0)$ resp. $\mathrm{emb}_{\mathbb{N}}(1)$ can successively be generated confirming at the same time that $x_3 \in \mathrm{ran}(\mathrm{emb}_{\mathbb{N}})$. If $x_3 \notin \mathrm{ran}(\mathrm{emb}_{\mathbb{N}})$ or $d(x_1, x_2) = q_{\mathrm{emb}_{\mathbb{N}}{}^{-1}(x_3)}$, then the machine does never yield a result.

Now let the suppositions of the second assertion be fulfilled, and let x_1, x_2 be given by ϱ-names as oracle sequences of an OTM. For any $n \in \mathbb{N}$, the machine can find a rational number q_m such that $|d(x_1, x_2) - q_m| \leq 2^{-(n+1)}$. This is done by simulating sufficiently large initial parts of computations of $d_{\mathrm{ap}}(x_1, x_2, k)$, for $q_k \in \{q_m - 2^{-(n+1)}, q_m + 2^{-(n+1)}\}$, $m \in \mathbb{N}$, up to having confirmed that $q_m - 2^{-(n+1)} \leq d(x_1, x_2) \leq q_m + 2^{-(n+1)}$, for some number m. Then this m can be put out as an index of a 2^{-n}-approximation of $d(x_1, x_2)$. □

It immediately follows

Proposition 9. *Let $\mathcal{X}$ be a SEMS with an EDS (η, δ), the functions $\mathrm{emb}_{\mathbb{N}}$, succ and d_{ap} be defined as above. Then a representation ϱ of $\mathcal{X}$ is a standard representation of $\mathcal{X}$ iff*

- *the element $S \circ \eta(0)$ is ϱ-computable,*
- *the functions succ and $S \circ \mathrm{emb}_{\mathbb{N}}{}^{-1}$ are (ϱ, ϱ)-computable,*
- *the function d_{ap} is (ϱ^3, ϱ)-computable,*
- NormLim *is $(\varrho^\omega, \varrho)$-computable, and*
- *the inequality relation $\neq$ is enumerable w.r.t. ϱ.*

In other words, the structure $(X; S \circ \eta(0); \mathrm{succ}, S \circ \mathrm{emb}_{\mathbb{N}}{}^{-1}, d_{\mathrm{ap}}, \mathrm{NormLim}; \neq)$ is effectively categorical. □

For the SEMS $\mathbb{R}$ of real numbers, we can suppose that $\eta(n) = n$ and $S_{\mathbb{R}}(0) = 0$. Then $\mathrm{emb}_{\mathbb{N}} = \mathrm{id}_{\mathbb{N}}$, and succ is the addition of 1, $\mathrm{succ}(n) = n + 1$, restricted to

the natural numbers as a subset of $\mathbb{R}$. Moreover, $S_{\mathbb{R}} \circ \mathrm{emb}_{\mathbb{N}}{}^{-1}(n) = S_{\mathbb{R}}(n) = q_n$. The function d_{ap} behaves as follows on $\mathbb{R}$:

$$d_{\mathrm{ap}}(x_1, x_2, x_3) = \begin{cases} 0 & \text{if } d_{\mathbb{R}}(x_1, x_2) < q_{x_3},\ x_3 \in \mathbb{N}, \\ 1 & \text{if } d_{\mathbb{R}}(x_1, x_2) > q_{x_3},\ x_3 \in \mathbb{N}, \\ \text{undefined} & \text{if } x_3 \notin \mathbb{N} \text{ or } d_{\mathbb{R}}(x_1, x_2) = q_{x_3}. \end{cases}$$

Corollary 6. *A representation ϱ of the real numbers is a standard representation iff*

- *the number* 0 *is ϱ-computable,*
- *the partial real functions* succ *and $S_{\mathbb{R}}$ are (ϱ, ϱ)-computable,*
- *the function d_{ap} is (ϱ^3, ϱ)-computable,*
- NormLim *is $(\varrho^\omega, \varrho)$-computable, and*
- *the inequality $\neq$ is enumerable w.r.t. ϱ.*

Thus, the real number structure $(\mathbb{R}; 0; \mathrm{succ}, S_{\mathbb{R}}, d_{\mathrm{ap}}, \mathrm{NormLim}; \neq)$ as well as $(\mathbb{R}; 1; +, -, \cdot, /, d_{\mathrm{ap}}, \mathrm{NormLim}; \neq)$ and $(\mathbb{R}; 1; +, -, \cdot\frac{1}{2}, d_{\mathrm{ap}}, \mathrm{NormLim}; \neq)$ are effectively categorical. □

This result is closely related to Hertling's [4] original one yielding the effectively categorical structure $(\mathbb{R}; 0, 1; +, -, \cdot, /, \mathrm{NormLim}; <)$.

So, under rather weak suppositions on the SEMS $\mathcal{X}$, we have specified a finite set of indicator functions and relations operating on the carrier of $\mathcal{X}$ only. It should be noticed, however, that the notion of enumerability of relations w.r.t. representations refers to recursively open sets and is not really based on the concept of approximate computability of functions. The question if there are effectively categorical single-sorted algebras, for example over the real numbers, which characterize the standard representations remains open.

6 Some Counterexamples

In this section, we deal with some special nonstandard representations showing that certain modifications of our conditions from the previous section are not sufficient to characterize the standard representations.

The first example proves that the requirement of computability of some constants of the EMSs cannot be avoided in this context.

Lemma 11. *To any EMS $\mathcal{X} = (X, d, S)$, there is a nonstandard representation ϱ such that every $((\varrho_{\mathrm{nC}}{}^{\mathcal{X}})^\alpha, \varrho_{\mathrm{nC}}{}^{\mathcal{X}})$-computable function $f : X^\alpha \rightarrowtail X$, for $\alpha \in \mathbb{N}_+ \cup \{\omega\}$, is $(\varrho^\alpha, \varrho)$-computable too, and every relation enumerable w.r.t. $\varrho_{\mathrm{nC}}{}^{\mathcal{X}}$ is also enumerable w.r.t. ϱ.*

Proof. Let $\varphi : \mathbb{N} \longrightarrow \mathbb{N}$ be a nonrecursive total function. For any $\varrho_{\mathrm{nC}}{}^{\mathcal{X}}$-name $\sigma \in \mathrm{CF}_x^{(S)}$, $x \in X$, let $\widehat{\sigma} \in \mathbb{F}$ be defined by

$$\widehat{\sigma}(2n) = \sigma(n), \quad \widehat{\sigma}(2n+1) = \varphi(n) \qquad (\text{for all } n \in \mathbb{N}).$$

The representation ϱ is determined by

$$\mathrm{dom}(\varrho) = \{\widehat{\sigma} : \sigma \in \mathrm{dom}({\varrho_{\mathrm{nC}}}^{\mathcal{X}})\}, \quad \text{and} \quad \varrho(\widehat{\sigma}) = {\varrho_{\mathrm{nC}}}^{\mathcal{X}}(\sigma).$$

In other words, the ϱ-names consist of the ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$-names on a first trace (of even numbered places) and of the sequence φ on a second trace (of odd places).

Any OTM computing w.r.t. ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ an α-ary function f on X, $\alpha \in \mathbb{N}_+ \cup \{\omega\}$, can easily be modified to a machine computing f w.r.t. ϱ. Moreover, it holds $\varrho \leq {\varrho_{\mathrm{nC}}}^{\mathcal{X}}$, but not ${\varrho_{\mathrm{nC}}}^{\mathcal{X}} \leq \varrho$, since φ is not recursive. Also there is no ϱ-computable element of $\mathcal{X}$. So ϱ is nonstandard. By Lemma 9, any relation which is enumerable w.r.t. ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ is also enumerable w.r.t. ϱ. □

The following result shows that the standard representations cannot be characterized by effectively categorical structures, possibly of infinite signature, which have only finitary basic functions.

Proposition 10. *Let $\mathcal{X} = (X, d, S)$ be an EMS possessing a noncomputable element. Then there is a nonstandard representation $\widehat{\varrho}$ such that:*

- *every ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$-computable element $x \in X$ is $\widehat{\varrho}$-computable,*
- *every function $f : X^k \rightarrowtail X^l$, $k, l \in \mathbb{N}_+$, which is $(({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^k, ({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^l)$-computable is $(\widehat{\varrho}^k, \widehat{\varrho}^l)$-computable too, and*
- *every relation $R \subseteq X^k$, $k \in \mathbb{N}_+$, which is enumerable w.r.t. ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ is also enumerable w.r.t. $\widehat{\varrho}$.*

Proof. The construction of $\widehat{\varrho}$ is similar to that of the representation from the previous proof. Let be fixed a noncomputable element $y \in X$ and a sequence $\sigma_y \in \mathrm{CF}_y^{(S)} = ({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^{-1}(y)$. By reasons of cardinality, there is a nonrecursive total function $\varphi_y : \mathbb{N} \longrightarrow \mathbb{N}$ such that, for every OTM $\mathcal{M}$, either $\mathcal{M}^{\sigma_y}(n)$ remains undefined for some input $n \in \mathbb{N}$, or $(\mathcal{M}^{\sigma_y}(2n+2))_{n\in\mathbb{N}} \neq (\varphi_y(n))_{n\in\mathbb{N}}$, i.e., the sequences don't coincide.

For any ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$-name $\sigma \in \bigcup_{x\in X} \mathrm{CF}_x^{(S)}$, let $\widehat{\sigma}_0, \widehat{\sigma}_1 \in \mathbb{F}$ be defined by

$$\begin{aligned} \widehat{\sigma}_0(0) &= 0, \\ \widehat{\sigma}_0(n+1) &= \sigma(n), \quad \text{for } n \in \mathbb{N}; \\ \widehat{\sigma}_1(0) &= 1, \\ \widehat{\sigma}_1(2n+1) &= \sigma(n), \\ \widehat{\sigma}_1(2n+2) &= \varphi_y(n), \quad \text{for } n \in \mathbb{N}. \end{aligned}$$

Thus, the first element $\widehat{\sigma}_i(0)$ gives the index $i \in \{0, 1\}$. Moreover, $\widehat{\sigma}_0$ corresponds to σ, where all values are shifted by one place, whereas $\widehat{\sigma}_1$ consists of the sequence σ on a first trace of the odd numbered places and of the values of function φ_y on the second trace given by the places $2n+2$, for $n \in \mathbb{N}$.

For $\sigma \in \mathrm{dom}({\varrho_{\mathrm{nC}}}^{\mathcal{X}})$, let

$$\begin{aligned} \widehat{\varrho}(\widehat{\sigma}_1) &= {\varrho_{\mathrm{nC}}}^{\mathcal{X}}(\sigma) \quad \text{if } {\varrho_{\mathrm{nC}}}^{\mathcal{X}}(\sigma) \text{ is not } {\varrho_{\mathrm{nC}}}^{\mathcal{X}}\text{-computable}, \\ \widehat{\varrho}(\widehat{\sigma}_i) &= {\varrho_{\mathrm{nC}}}^{\mathcal{X}}(\sigma) \quad \text{if } {\varrho_{\mathrm{nC}}}^{\mathcal{X}}(\sigma) \text{ is } {\varrho_{\mathrm{nC}}}^{\mathcal{X}}\text{-computable and } i \in \{0, 1\}, \end{aligned}$$

i.e., the domain of the representation $\widehat{\varrho}$ is defined as

$$\begin{aligned}\mathrm{dom}(\widehat{\varrho}) = \{\widehat{\sigma}_i :\ & \sigma \in \mathrm{dom}({\varrho_{\mathrm{nC}}}^{\mathcal{X}}) \text{ and} \\ & i \in \{0,1\} \text{ if } {\varrho_{\mathrm{nC}}}^{\mathcal{X}}(\sigma) \text{ is } {\varrho_{\mathrm{nC}}}^{\mathcal{X}}\text{-computable}, \\ & i = 1 \text{ if } {\varrho_{\mathrm{nC}}}^{\mathcal{X}}(\sigma) \text{ is not } {\varrho_{\mathrm{nC}}}^{\mathcal{X}}\text{-computable}\,\}.\end{aligned}$$

One easily sees that $\widehat{\varrho} \leq {\varrho_{\mathrm{nC}}}^{\mathcal{X}}$. By our supposition on the function φ_y, there is no OTM transforming the special sequence $\sigma_y \in ({\varrho_{\mathrm{nC}}}^{\mathcal{X}})^{-1}(y)$ into some element from $\widehat{\varrho}^{-1}(y)$. Thus, ${\varrho_{\mathrm{nC}}}^{\mathcal{X}} \not\leq \widehat{\varrho}$, and $\widehat{\varrho}$ is nonstandard.

On the other hand, for any recursive ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$-name σ, the related $\widehat{\varrho}$-name $\widehat{\sigma}_0$ is recursive too.

Moreover, any finitary function $f : X^k \rightarrowtail X^l$, $k, l \in \mathbb{N}_+$, which is computable w.r.t. ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$, is also computable w.r.t. $\widehat{\varrho}$. Indeed, for computable arguments $x_1, \ldots, x_k$ from the domain of f, also the components $x'_1, \ldots, x'_l$ of the value $(x'_1, \ldots, x'_l) = f(x_1, \ldots, x_k)$ are computable (w.r.t. ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$). Thus, an OTM $\mathcal{M}$ computing f w.r.t. ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$ can easily be modified to a machine $\mathcal{M}'$ computing f w.r.t. $\widehat{\varrho}$. To this purpose, given $\widehat{\varrho}$-names of the arguments, $\mathcal{M}'$ successively puts here corresponding ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$-names and simulates $\mathcal{M}$ on them. If at least one input name starts with 1 at place 0 (i.e., the table of function φ_y is communicated by this name), let $\mathcal{M}'$ output the modifications $\widehat{\sigma}_1$ of the output sequences σ produced by $\mathcal{M}$; if all input names start with number 0 (i.e., all arguments are computable), let the output sequences have the forms $\widehat{\sigma}_0$.

Finally, since $\widehat{\varrho} \leq {\varrho_{\mathrm{nC}}}^{\mathcal{X}}$, any k-ary relation on X, $k \in \mathbb{N}_+$, which is enumerable w.r.t. ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$, is enumerable w.r.t. $\widehat{\varrho}$ too. □

In particular, Proposition 10 applies to the SEMS $\mathbb{R}$ of real numbers as well as to the sequence spaces $\mathbb{B}$ and $\mathbb{F}$. Thus, it negatively answers the question if there is an effectively categorical structure over the reals with only finitary basic functions. This problem was left open in [4]. Unfortunately, the question discussed at the end of the previous section, if the relations can be avoided in effectively categorical structures, is still open. We remark without proof that in the case of total structures, i.e., if all basic functions have to be total, as well as if the weak definition of computability w.r.t. representations is used, one can show that relations cannot be avoided in effectively categorical structures characterizing the standard representations.

With respect to the representation $\widehat{\varrho}$, notice that an infinitary operation like NormLim can yield uncomputable results $\mathrm{NormLim}((x_n)_{n\in\mathbb{N}})$, even if all arguments x_n are ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$-computable. Since the skeleton gives a dense set $\mathrm{ran}(S)$, every element of X can even be obtained in this way. Thus, if $\mathcal{X}$ contains elements which are not ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$-computable, the function NormLim cannot be $(\widehat{\varrho}^{\omega}, \widehat{\varrho})$-computable.

We close with some remarks on the *general* or naive *Cauchy representation* ${\varrho_{\mathrm{gC}}}^{\mathcal{X}}$. It is defined as ${\varrho_{\mathrm{gC}}}^{\mathcal{X}} : \mathbb{F} \rightarrowtail X$, $\mathrm{dom}({\varrho_{\mathrm{gC}}}^{\mathcal{X}}) = \{\sigma \in \mathbb{F} : \lim_{n\to\infty} S \circ \sigma(n) \text{ exists}\}$, ${\varrho_{\mathrm{gC}}}^{\mathcal{X}}(\sigma) = \lim_{n\to\infty} S \circ \sigma(n)$, for $\sigma \in \mathrm{dom}({\varrho_{\mathrm{gC}}}^{\mathcal{X}})$.

There are several reasons that approximate computability cannot be based on ${\varrho_{\mathrm{gC}}}^{\mathcal{X}}$ instead of ${\varrho_{\mathrm{nC}}}^{\mathcal{X}}$, cf. [9,2,11]. They illustrate the differences between the nonconstructive point of view in classical analysis and the demands in effective analysis, respectively. A rather simple reason, perhaps surprising on a first view, consists in the discontinuity of ${\varrho_{\mathrm{gC}}}^{\mathcal{X}}$. One easily shows

Lemma 12. *For any two elements $x, y \in X$, $x \neq y$, there is a converging sequence of ${\varrho_{\mathrm{gC}}}^{\mathcal{X}}$-names of x, $(\sigma_n)_{n\in\mathbb{N}}$, ${\varrho_{\mathrm{gC}}}^{\mathcal{X}}(\sigma_n) = x$ for all $n \in \mathbb{N}$, such that ${\varrho_{\mathrm{gC}}}^{\mathcal{X}}(\lim_{n\to\infty} \sigma_n) = y$.* □

Thus, ${\varrho_{\mathrm{gC}}}^{\mathcal{X}}$ is not a standard representation if $\mathrm{card}(X) \geq 2$. Nevertheless, as it has been remarked by Brattka and Hertling [2], continuity and $({\varrho_{\mathrm{gC}}}^{\mathcal{X}}, {\varrho_{\mathrm{gC}}}^{\mathcal{X}})$-continuity of functions over $\mathcal{X}$ coincide.

Within our framework, the situation is different. Even if $\mathrm{dom}({\varrho_{\mathrm{gC}}}^{\mathcal{X}})$ is not closed for $\mathrm{card}(X) \geq 2$, one can see that $\mathrm{dom}(f)$ is closed for any $({\varrho_{\mathrm{gC}}}^{\mathcal{X}}, {\varrho_{\mathrm{gC}}}^{\mathcal{X}})$-computable function $f : X \rightarrowtail X$. Thus, there are $({\varrho_{\mathrm{nC}}}^{\mathcal{X}}, {\varrho_{\mathrm{nC}}}^{\mathcal{X}})$-computable functions which are not $({\varrho_{\mathrm{gC}}}^{\mathcal{X}}, {\varrho_{\mathrm{gC}}}^{\mathcal{X}})$-computable, as well as $({\varrho_{\mathrm{gC}}}^{\mathcal{X}}, {\varrho_{\mathrm{gC}}}^{\mathcal{X}})$-computable functions which are not $({\varrho_{\mathrm{nC}}}^{\mathcal{X}}, {\varrho_{\mathrm{nC}}}^{\mathcal{X}})$-computable.

Acknowledgement

I would like to thank an anonymous referee for some useful hints.

References

1. V. Brattka: Recursive and computable operations over topological structures. Dissertation. Informatik–Berichte 255 - 7/1999, FernUniversität Hagen
2. V. Brattka, P. Hertling: Topological properties of real number representations. to appear in: Theoretical Computer Science (CCA'99 Special Issue)
3. A. Hemmerling: Effective metric spaces and representations of the reals. to appear in: Theoretical Computer Science (CCA'99 Special Issue)
4. P. Hertling: A real number structure that is effectively categorical. Math. Log. Quart., v. 45, 1999, 147 – 182
5. P. Hertling: Effectivity and effective continuity of functions between computable matric spaces. in: D. S. Bridges et al. (eds.), Combinatorics, Complexity & Logic. Proc. of the DMTCS'96, Springer–Verlag, Singapore 1997, 264–275
6. K.–I. Ko: Complexity theory of real functions. Birkhäuser, Boston et al., 1991
7. K.–I. Ko, H. Friedman: Computational complexity of real functions. Theoretical Computer Science 20, 1982, 323–352
8. M. B. Pour-El, J. I. Richards: Computability in analysis and physics. Springer–Verlag, Berlin et al. 1989
9. K. Weihrauch: Computability. Springer–Verlag, Berlin et al., 1987
10. K. Weihrauch: Computability on computable metric spaces. Theoretical Computer Science, v. 113, 1993, 191 – 210
11. K. Weihrauch: Computable analysis. Springer–Verlag, Berlin et al., 2000

Banach–Mazur Computable Functions on Metric Spaces

Peter Hertling

Theoretische Informatik I, FernUniversität Hagen, 58084 Hagen, Germany
peter.hertling@fernuni-hagen.de

Abstract. We present Mazur's continuity results for Banach–Mazur computable functions on computable real numbers in the slightly more general setting of metric spaces satisfying suitable computability conditions. Additionally, we prove that the image of a computable, computably convergent sequence under a Banach–Mazur computable function is again computably convergent.

1 Introduction

Among the first people who studied computability over the real numbers in the sense of recursion theory were Banach and Mazur, as is documented in [1]. The book [10], edited by A. Grzegorczyk and H. Rasiowa and based on lecture notes "S. Mazur 'Computable Analysis' in the academic year 1949–1950" [10, Foreword by Grzegorczyk and Rasiowa], contains a detailed exposition of their results. Some of them had already been presented by Grzegorczyk [5].

Among other things, Mazur [10] considered functions f mapping computable real numbers to computable real numbers satisfying the following condition: if a sequence $(x_n)_n$ of real numbers is computable and contained in the domain of f then its image under f, the sequence $(f(x_n))_n$, is again a computable sequence of real numbers. We call such functions *Banach–Mazur computable* or, shorter, *BM–computable*. One of the most interesting properties proved by Mazur about Banach–Mazur computable functions on the computable real numbers is that any function of this kind which is defined on all computable real numbers in some interval must be continuous in this interval. Since, according to [10, Foreword], the results presented there "concerning general recursive mathematical objects are obtained by Mazur himself after the war" and are contained in the above mentioned lecture notes from 1949–1950, it is possible that this is historically the first continuity result for functions satisfying an effectivity condition which does not obviously imply continuity, and that it precedes the results by Markov [8,9] and the effective continuity results by Kreisel, Lacombe, Shoenfield [6], by Ceĭtin [2,3], and by Moschovakis [11]. But it should be noticed that in [6,2,3,11] on the one hand functions are considered which are computable in a slightly stronger sense than the Banach–Mazur computable functions, and that on the other hand in these papers not only continuity but even effective continuity in various senses is derived for such functions.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 69–81, 2001.

Let us have a look at the two main results by Mazur concerning continuity for Banach–Mazur computable functions over the real numbers. Let us fix a (possibly partial) Banach–Mazur computable function mapping computable real numbers to computable real numbers.

The first result says that, for any computable sequence in the domain of the function converging to a point in the domain its image under the function converges to the image of the point. This result can be generalized straightforwardly to computable metric spaces.

The second result says that the function is even continuous if the domain is equal to the set of all computable real numbers in an interval. This can also be expressed by saying that, under this additional condition on the domain, for any sequence in the domain of the function converging to a point in the domain its image under the function converges to the image of the point, i.e. one can drop the condition that the first sequence is computable. This result can also be generalized straightforwardly to computable metric spaces. Furthermore, instead of assuming that the domain of the function is an interval of computable real numbers it is sufficient to assume that the domain contains a dense computable sequence.

After presenting some preliminary material and the versions of these two results for metric spaces including proofs, we shall prove a third, complementing result: for any computable sequence in the domain of the function which converges *computably* to a point in the domain its image under the function converges *computably* to the image of the point.

Finally, we shall discuss shortly the relation between the notions of Banach–Mazur computability and the other computability notion mentioned above and considered in [6,2,3,11], where it was shown to be equivalent to effective continuity if the considered functions have a sufficiently simple domain of definition.

2 Notions

2.1 Basic Notation and Computability

By $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$, we denote the set of natural numbers, i.e. nonnegative integers, the set of rational numbers, and the set of real numbers, respectively. A sequence $x_0, x_1, x_2, \ldots$ will be denoted by $(x_n)_n$ or $(x_i)_i$, etc. For two sets X and Y, by $f :\subseteq X \to Y$ we denote a possibly partial function whose domain of definition is a subset of X, and whose range is a subset of Y. We denote the domain of definition of f by $\operatorname{dom} f$ and the range of f by $\operatorname{range} f$. If $\operatorname{dom} f = X$, we call the function f *total* and may indicate this by writing $f : X \to Y$ instead of $f :\subseteq X \to Y$. For an integer $k \geq 1$, we denote by $P^{(k)}$ the set of all computable —in the sense of recursion theory— functions $f :\subseteq \mathbb{N}^k \to \mathbb{N}$. We use the standard bijection $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N}$ defined by $\langle i, j \rangle = (i+j)(i+j+1)/2 + j$, for all $i, j \in \mathbb{N}$. Inductively, we define $\langle i_1, \ldots, i_k, i_{k+1} \rangle := \langle \langle i_1, \ldots, i_k \rangle, i_{k+1} \rangle$, for $k \geq 2$. We fix a total standard numbering φ of all computable natural number functions, i.e. a total surjective function $\varphi : \mathbb{N} \to P^{(1)}$ satisfying the following two conditions:

(1) (universality) the function $u :\subseteq \mathbb{N}^2 \to \mathbb{N}$ defined by $u(i,j) := \varphi(i)(j)$, for all $i, j \in \mathbb{N}$, is computable; (2) (smn–property) for any computable function $f :\subseteq \mathbb{N}^2 \to \mathbb{N}$ there exists a total computable function $r : \mathbb{N} \to \mathbb{N}$ with $f(i,j) = \varphi(r(i))(j)$, for all $i, j \in \mathbb{N}$. Often we write φ_i instead of $\varphi(i)$, and $\varphi_i(j)$ instead of $\varphi(i)(j)$. A set $A \subseteq \mathbb{N}^k$ is *computably enumerable*, iff there is a computable function $f :\subseteq \mathbb{N}^k \to \mathbb{N}$ with $\operatorname{dom} f = A$. It is *decidable*, iff both it and its complement are computably enumerable.

2.2 Numbered Sets and Computability

A *numbering* of a countable set X is a surjective function $\nu :\subseteq \mathbb{N} \to X$. A *numbered set* is a pair (X, ν) consisting of a countable set X and a numbering $\nu :\subseteq \mathbb{N} \to X$.

Definition 1. Let (X, ν) be a numbered set.

1. A number $i \in \mathbb{N}$ is called a *ν–index* of an element $x \in X$, iff $\nu(i) = x$. We often write ν_i for $\nu(i)$.
2. A sequence $(x_n)_n$ in X is called *ν–computable*, iff there is a total computable function $f : \mathbb{N} \to \mathbb{N}$ with $x_n = \nu(f(n))$ for all n. A number i is called a *ν–index* for the sequence $(x_n)_n$, iff the function φ_i is total and $x_n = \nu(\varphi_i(n))$ for all n.
3. Let $(X^{(i)}, \nu^{(i)})$ be numbered sets for $i = 1, \ldots n$, $n \geq 2$. One defines a numbering $(\nu^{(1)}, \ldots, \nu^{(n)})$ of $X^{(1)} \times \ldots \times X^{(n)}$ by

$$\operatorname{dom}(\nu^{(1)}, \ldots, \nu^{(n)}) := \{\langle i_1, \ldots, i_n\rangle \mid i_j \in \operatorname{dom} \nu^{(j)}, \text{ for all } j = 1, \ldots, n\},$$

and by $(\nu^{(1)}, \ldots, \nu^{(n)})(\langle i_1, \ldots, i_n\rangle) := (\nu^{(1)}(i_1), \ldots, \nu^{(n)}(i_n))$, for all $i_j \in \operatorname{dom} \nu^{(j)}$, $j = 1, \ldots, n$.

In the following definition the letters BM stand for "Banach–Mazur". The computability notion for functions was introduced by them. The computability notion for sets is taken from Lachlan [7].

Definition 2. Let (X, ν) and (X', ν') be numbered sets.

1. A subset $Y \subseteq X$ is called *BM–ν–computably enumerable*, iff for any ν–computable sequence $(x_n)_n$ in X the set $\{n \in \mathbb{N} \mid x_n \in Y\}$ is computably enumerable.
2. A function $f :\subseteq X \to X'$ is called *BM–(ν, ν')–computable*, iff for any ν–computable sequence $(x_n)_n$ with $x_n \in \operatorname{dom} f$, for all n, the sequence $(f(x_n))_n$ is ν'–computable.
3. Assume that $X \subseteq X'$. The numbering ν is *BM–reducible* to ν', written: $\nu \leq_{\mathrm{BM}} \nu'$, iff any ν–computable sequence is also ν'–computable. We write $\nu \equiv_{\mathrm{BM}} \nu'$, iff $\nu \leq_{\mathrm{BM}} \nu'$ and $\nu' \leq_{\mathrm{BM}} \nu$.

While in this paper we will deal mostly with these computability notions, the following notions are more common and defined analogously.

Definition 3. Let (X, ν) and (X', ν') be numbered sets.

1. A subset $Y \subseteq X$ is called *ν–computably enumerable*, iff there exists a computably enumerable set $A \subseteq \mathbb{N}$ with $\nu^{-1}(Y) = A \cap \operatorname{dom} \nu$.
2. A function $f :\subseteq X \to X'$ is called *(ν, ν')–computable*, iff there is a computable function $F :\subseteq \mathbb{N} \to \mathbb{N}$ with $f\nu(i) = \nu' F(i)$, for all $i \in \operatorname{dom}(f\nu)$.
3. Assume that $X \subseteq X'$. The numbering ν is *reducible* to ν', written: $\nu \leq \nu'$, iff there is a computable function $F :\subseteq \mathbb{N} \to \mathbb{N}$ with $\nu(i) = \nu'(F(i))$, for all $i \in \operatorname{dom} \nu$. We write $\nu \equiv \nu'$, iff $\nu \leq \nu'$ and $\nu' \leq \nu$.

The second set of definitions is slightly stronger than the first.

Lemma 4. *Let (X, ν) and (X', ν') be numbered sets. Any ν–computably enumerable set $Y \subseteq X$ is BM–ν–computably enumerable. Any (ν, ν')–computable function is BM–(ν, ν')–computable. And $\nu \leq \nu'$ implies $\nu \leq_{\mathrm{BM}} \nu'$.*

We omit the simple proof. We make some comments on the reverse of Lemma 4 at the end of Section 3.

Very often we will omit the prefix ν resp. (ν, ν') etc. indicating the numberings when these are clear from the context. Then we speak simply of *computable* sequence, of an *index* of an element or a computable sequence, of a *(BM–)computably enumerable* set, and of a *(BM–)computable* function.

2.3 Computable Metric Spaces

Often one considers numbered sets with additional properties. For example, the set X could be a metric space. In that case we speak of a *numbered metric space*, understanding that besides the numbering ν of the set X we have also fixed a metric $d : X \times X \to \mathbb{R}$ on the set X. We will usually denote any metric on any metric space simply by d, with one exception: the distance of two rational or real numbers x and y is denoted $|x - y|$. It should always be clear from the context on which space the considered metric d is defined.

Example 5. The function $\nu_{\mathbb{Q}} : \mathbb{N} \to \mathbb{Q}$ defined by $\nu_{\mathbb{Q}}(\langle i, j, k \rangle) := (i - j)/(1 + k)$, for $i, j, k \in \mathbb{N}$, is a total numbering of $\mathbb{Q}$. Whenever we speak about computability for objects involving rational numbers, we always mean the computability notions defined with respect to the numbered set $(\mathbb{Q}, \nu_{\mathbb{Q}})$. This pair is a numbered metric space, with the usual Euclidean metric. In fact, it is even a computable metric space according to the following definition.

Definition 6. A numbered metric space (X, ν) is called

1. *BM–semi–computable*, iff the set $\{(x, y, q) \in X^2 \times \mathbb{Q} \mid d(x, y) < q\}$ is BM–$(\nu, \nu, \nu_{\mathbb{Q}})$–computably enumerable,
2. *BM–computable*, iff it is BM–semi–computable and the set $\{(x, y, q) \in X^2 \times \mathbb{Q} \mid d(x, y) > q\}$ is BM–$(\nu, \nu, \nu_{\mathbb{Q}})$–computably enumerable.

The notions *semi–computable* and *computable* for a numbered metric space are defined analogously.

If a numbered metric space is semi–computable (resp. computable) then it is also BM–semi–computable (resp. BM–computable).

We need precise notions what it means to approximate an element efficiently by a sequence of elements.

Definition 7. Let (X, ν) be a numbered metric space, and let $(x_n)_n$ be a sequence of elements in X.

1. The sequence $(x_n)_n$ is *computably Cauchy*, iff there exists a total computable function $e : \mathbb{N} \to \mathbb{N}$ such that $\forall n, m, l \in \mathbb{N}.\ (m, l \geq e(n) \Rightarrow d(x_m, x_l) \leq 2^{-n})$. Such a function e is called a *computable modulus of convergence for* $(x_n)_n$. The sequence $(x_n)_n$ *converges computably*, iff it is computably Cauchy and converges to an element in X.
2. The sequence $(x_n)_n$ is a *fast Cauchy* sequence, iff $\forall n, m \in \mathbb{N}.\ d(x_n, x_m) \leq 2^{-\min\{n,m\}}$. The sequence $(x_n)_n$ *converges fast*, iff it is a fast Cauchy sequence and converges to an element in X.

Obviously, any fast Cauchy sequence is also computably Cauchy. Note that, if $e : \mathbb{N} \to \mathbb{N}$ is a computable modulus of convergence for a sequence $(x_n)_n$, then the function $\widetilde{e} : \mathbb{N} \to \mathbb{N}$ defined by $\widetilde{e}(n) := \max\{e(0), \ldots, e(n)\}$ is a nondecreasing computable modulus of convergence for $(x_n)_n$.

Definition 8. Let (Y, ν) be a numbered metric space, and let $X \supseteq Y$ be a metric space containing Y such that every element in X is the limit of some fast Cauchy, computable sequence of elements of Y. We define the *derived* numbering $\partial_X \nu :\subseteq \mathbb{N} \to X$ of X by

$$\partial_X \nu(i) = x \iff \begin{array}{l} i \text{ is a } \nu\text{–index of some computable,} \\ \text{fast Cauchy sequence in } Y \text{ with limit } x, \end{array}$$

for $i \in \mathbb{N}$ and $x \in X$. If $X = Y$, we usually write $\partial\nu$ for $\partial_X \nu$.

It is clear that this is indeed a definition of a numbering of X. Since from an index of an element $x \in Y$ one can easily compute an index of the constant sequence in Y whose components are all equal to x, we observe $\nu \leq \partial_X \nu$. For the numbering $\partial_X \nu$ also the converse holds true: $\partial_X(\partial_X \nu) \leq \partial_X \nu$, as one checks by taking a diagonal sequence and using some triangle inequalities. Hence, $\partial_X(\partial_X \nu) \equiv \partial_X \nu$,

Definition 9. A numbered metric space (X, ν) is called *BM–weakly complete*, iff $\partial\nu \leq_{\mathrm{BM}} \nu$. It is called *weakly complete*, iff $\partial\nu \leq \nu$.

Hence, (X, ν) is BM–weakly complete, iff $\partial\nu \equiv_{\mathrm{BM}} \nu$, and (X, ν) is weakly complete, iff $\partial\nu \equiv \nu$. Any weakly complete space is also BM–weakly complete.

Example 10. A real number x is called *computable*, iff there exists a computable, fast Cauchy sequence of rational numbers with limit x. Let $\mathbb{R}_c$ denote the set of all computable real numbers. The pair $(\mathbb{R}_c, \nu_{\mathbb{R}_c})$ with the numbering $\nu_{\mathbb{R}_c} := \partial_{\mathbb{R}_c}\nu_{\mathbb{Q}}$ and endowed with the Euclidean metric is a computable, weakly complete metric space. Whenever we speak about computability for objects involving computable real numbers, we always mean the computability notions defined with respect to the numbered set $(\mathbb{R}_c, \nu_{\mathbb{R}_c})$.

Using the space $(\mathbb{R}_c, \nu_{\mathbb{R}_c})$ one can give another useful characterization of BM–computable metric spaces and of computable metric spaces.

Lemma 11. *A numbered metric space (X, ν) is BM–computable (resp. computable), iff its metric $d : X \times X \to \mathbb{R}$ satisfies* $\operatorname{range} d \subseteq \mathbb{R}_c$ *and is BM–$((\nu,\nu), \nu_{\mathbb{R}_c})$–computable (resp. $((\nu,\nu), \nu_{\mathbb{R}_c})$–computable).*

We omit the proof.

We will have to encode the halting problem into a computable sequence. The following lemma gives two possibilities how to do that.

Lemma 12. *Let (X, ν) be a BM–weakly complete metric space, and let $(x_n)_n$ be a computable and computably convergent sequence with limit $x_\infty \in X$.*

1. *Let $g : \mathbb{N} \to \mathbb{N}$ be a total computable function, $A := \operatorname{range} g$. Then the sequence $(z_n)_n$ defined by:*

$$z_n := \begin{cases} x_\infty & \text{if } n \notin A, \\ x_m & \text{if } n \in A \text{ and } m = \min\{l \mid g(l) = n\} \end{cases}$$

   is computable.
2. *Let $h :\subseteq \mathbb{N} \to \mathbb{N}$ be a computable function with the property that the set*

$$\{\langle n, k\rangle \mid n \in \operatorname{dom} h \quad \text{and} \quad h(n) \leq k\}$$

   is decidable. Then the sequence $(z_n)_n$ defined by:

$$z_n := \begin{cases} x_\infty & \text{if } n \notin \operatorname{dom} h, \\ x_{h(n)} & \text{if } n \in \operatorname{dom} h \end{cases}$$

   is computable.

Proof. Let $e : \mathbb{N} \to \mathbb{N}$ be a nondecreasing, computable modulus of convergence for $(x_n)_n$. For the proof of the first assertion, we define a computable function $f : \mathbb{N}^2 \to \mathbb{N}$ by

$$f(n,i) := \begin{cases} e(i) & \text{if } n \notin \{g(0), \ldots, g(e(i))\}, \\ m & \text{if } n \in \{g(0), \ldots, g(e(i))\} \text{ and } m = \min\{l \mid g(l) = n\}. \end{cases}$$

For the proof of the second assertion, we define a computable function $f : \mathbb{N}^2 \to \mathbb{N}$ by

$$f(n,i) := \begin{cases} e(i) & \text{if } n \notin \operatorname{dom} h \text{ or } h(n) > e(i), \\ h(n) & \text{if } n \in \operatorname{dom} h \text{ and } h(n) \leq e(i). \end{cases}$$

In both cases, the sequence $(x_{f(n,i)})_{\langle n,i\rangle}$ is ν–computable, and, for each $n \in \mathbb{N}$, the sequence $(x_{f(n,i)})_i$ converges fast to z_n. Thus, by the smn–property, the sequence $(z_n)_n$ is $\partial\nu$–computable. Since we assume (X,ν) to be BM–weakly complete, we obtain that $(z_n)_n$ is also ν–computable. That was to be shown. □

In the following well–known example we show how one can apply the lemma in order to encode the halting problem into a computable sequence.

Example 13. In Example 5 we have defined the numbered metric space $(\mathbb{Q}, \nu_{\mathbb{Q}})$, and we mentioned that it is computable. Here we show that it is not BM–weakly complete. We show this by exhibiting a sequence $(z_n)_n$ of rational numbers which is $\partial\nu_{\mathbb{Q}}$–computable, but not $\nu_{\mathbb{Q}}$–computable. Let $K \subseteq \mathbb{N}$ be a computably enumerable but not decidable set, and let $g : \mathbb{N} \to \mathbb{N}$ be a total computable function with range $g = K$. Note that the sequence $(2^{-n})_n$ is a $\nu_{\mathbb{Q}}$–computable, fast converging sequence of rational numbers with limit 0. Then it is also $\partial\nu_{\mathbb{Q}}$–computable. We define a sequence $(z_n)_n$ as in Lemma 12.1 by

$$z_n := \begin{cases} 0 & \text{if } n \notin K, \\ 2^{-m} & \text{if } n \in K \text{ and } m = \min\{l \mid g(l) = n\}. \end{cases}$$

(Alternatively, one could take a computable function $h :\subseteq \mathbb{N} \to \mathbb{N}$ with $\operatorname{dom} h = K$ and such that the set $\{\langle n,k\rangle \mid n \in \operatorname{dom} h \text{ and } h(n) \leq k\}$ is decidable, and define $(z_n)_n$ as in Lemma 12.2.) Since $(\mathbb{Q}, \partial\nu_{\mathbb{Q}})$ is weakly complete, by Lemma 12.1 the sequence $(z_n)_n$ is $\partial\nu_{\mathbb{Q}}$–computable. But, since the set $\{0\}$ is a $\nu_{\mathbb{Q}}$–computably enumerable subset of $\mathbb{Q}$, the sequence $(z_n)_n$ cannot be $\nu_{\mathbb{Q}}$–computable. Otherwise the complement of K would be computably enumerable.

3 Banach–Mazur Computable Functions and Converging Sequences

In this section we first present Mazur's continuity results for Banach–Mazur computable functions in the slightly more general context of BM–computable metric spaces. Then we prove a result on computably convergent sequences and Banach–Mazur computable functions, which seems to be new even for the special case of functions on computable real numbers or on total computable number functions. In the end we make some comments on the relation between Banach–Mazur computable functions and computable functions.

The following three results, Lemma 14, Theorem 15, and Theorem 16 are straightforward generalizations of corresponding results by Mazur [10]; see also

Grzegorczyk [5] and Pour–El [12]. Mazur formulated the results for the case of Banach–Mazur computable functions mapping computable real numbers to computable real numbers. Pour–El considered functions defined on a subset of $P^{(1)}$ and with range either in $\mathbb{N}$ or in $R^{(1)} := \{f \in P^{(1)} \mid f \text{ is total}\}$. Later we shall strengthen Lemma 14.

Lemma 14. *Let (X, ν) be a BM–weakly complete metric space, and let (X', ν') be a BM–computable metric space. Let $f :\subseteq X \to X'$ be a BM–computable function. Let $(x_n)_n$ be a ν–computable sequence in X with $x_n \in \operatorname{dom} f$ for all n. If $(x_n)_n$ converges computably to an element $x_\infty \in \operatorname{dom} f$, then the sequence $(f(x_n))_n$ converges to $f(x_\infty)$.*

Proof. Assume the statement is not true, that is, assume the sequence $(x_n)_n$ converges computably to an element $x_\infty \in \operatorname{dom} f$, but the sequence $(f(x_n))_n$ does not converge to $f(x_\infty)$. Then there is a rational number $\varepsilon > 0$ such that there are infinitely many n with $d(f(x_n), f(x_\infty)) > \varepsilon$. Since f is BM–computable and X' is BM–computable, the set $\{n \in \mathbb{N} \mid d(f(x_n), f(x_\infty)) > \varepsilon\}$ is computably enumerable. Hence, there is a total computable function $h : \mathbb{N} \to \mathbb{N}$ with $h(k) \geq k$ and $d(f(x_{h(k)}), f(x_\infty)) > \varepsilon$ for all k. Furthermore, the sequence $(x_{h(k)})_k$ is computable and converges computably to x_∞.

Now, let $K \subseteq \mathbb{N}$ be a computably enumerable but not decidable set, and let $g : \mathbb{N} \to \mathbb{N}$ be a total computable function with $\operatorname{range} g = K$. We define a sequence $(z_n)_n$ by

$$z_n := \begin{cases} x_\infty & \text{if } n \notin K, \\ x_{h(m)} & \text{if } n \in K \text{ and } m = \min\{l \mid g(l) = n\}. \end{cases}$$

By Lemma 12.1 the sequence $(z_n)_n$ is computable. Since the function f is BM–computable, also the sequence $(f(z_n))_n$ is computable. Finally, since the metric space X' is BM–computable, the sequence $(d(f(x_\infty), f(z_n))_n$ is a computable sequence of real numbers. But this sequence satisfies:

$$d(f(x_\infty), f(z_n)) \begin{cases} = 0 & \text{if } n \notin K, \\ > \varepsilon & \text{if } n \in K, \end{cases}$$

implying that K is decidable. Contradiction! □

Theorem 15. *Let (X, ν) be a BM–semi–computable and BM–weakly complete metric space, and let (X', ν') be a BM–computable metric space. Let $f :\subseteq X \to X'$ be a BM–computable function. Let $(x_n)_n$ be a ν–computable sequence in X with $x_n \in \operatorname{dom} f$ for all n. If $(x_n)_n$ converges to an element $x_\infty \in \operatorname{dom} f$, then the sequence $(f(x_n))_n$ converges to $f(x_\infty)$.*

Proof. Assume the statement is not true, that is, assume the sequence $(x_n)_n$ converges to an element $x_\infty \in \operatorname{dom} f$, but the sequence $(f(x_n))_n$ does not converge to $f(x_\infty)$. Then there is a rational number $\varepsilon > 0$ such that for each k the set

$$A_k := \{n \mid d(x_n, x_\infty) < 2^{-k} \text{ and } d(f(x_n), f(x_\infty)) > \varepsilon\}$$

is nonempty. Since X is BM–semi–computable and X' is BM–computable, the set $\{\langle k, n\rangle \mid n \in A_k\}$ is computably enumerable. Hence, there is a total computable function $h : \mathbb{N} \to \mathbb{N}$ with $h(k) \in A_k$ for all k. The sequence $(x_{h(n)})_n$ is computable, converges computably to x_∞, and satisfies $d(f(x_{h(n)}), f(x_\infty)) > \varepsilon$ for all n. This contradicts Lemma 14. □

Let (X, ν) be a numbered metric space. We call a subset $Y \subseteq X$ *ν–computably separable*, iff there is a ν–computable sequence $(a_n)_n$ in X such that the set $\{a_n \mid n \in \mathbb{N}\}$ is a dense subset of Y.

Theorem 16. *Let (X, ν) be a BM–semi–computable and BM–weakly complete metric space, and let (X', ν') be a BM–computable metric space. Any BM–computable function $f :\subseteq X \to X'$ with a ν–computably separable domain of definition is continuous.*

Proof. Let $f :\subseteq X \to X'$ be a BM–computable function, and let $(a_n)_n$ be a computable sequence in X such that the set $\{a_n \mid n \in \mathbb{N}\}$ is a dense subset of $\operatorname{dom} f$. We wish to show that f is continuous. Therefore we will apply Lemma 14 twice. First, we use it to prove the following claim:

$$\begin{array}{l}\text{For any } b \in \operatorname{dom} f \text{ and any } \delta > 0 \text{ and } \varepsilon > 0\\ \text{there exists some } n \text{ with } d(b, a_n) < \delta \text{ and } d(f(b), f(a_n)) < \varepsilon.\end{array} \tag{1}$$

In order to prove the claim, let us fix some $b \in \operatorname{dom} f$. Since arbitrarily close to b there are elements in the sequence $(a_n)_n$ and since the space X is BM–semi–computable, there is a total computable function $g : \mathbb{N} \to \mathbb{N}$ such that $d(b, a_{g(n)}) < 2^{-n}$, for all n. The sequence $(a_{g(n)})_n$ is computable and converges computably to b. By Lemma 14, the sequence $(f(a_{g(n)}))_n$ converges to $f(b)$. Hence, the claim (1) follows.

We come to the second part of the proof. In order to show that f is continuous, we have to show that f is continuous in every point of its domain. Let us fix some $c \in \operatorname{dom} f$ and, for the sake of a contradiction, let us assume that f is not continuous in c. Then there is a rational number $\varepsilon > 0$ such that arbitrarily close to c there are points $b \in \operatorname{dom} f$ with $d(f(c), f(b)) > 2\varepsilon$. Since, due to the claim (1) proved above, arbitrarily close to any such b there is an element a_m of the sequence $(a_n)_n$ with $d(f(b), f(a_m)) < \varepsilon$, we conclude that arbitrarily close to c there are elements a_m in the sequence $(a_n)_n$ with $d(f(a_m), f(c)) > \varepsilon$. That means that for each k the set

$$A_k := \{n \mid d(a_n, c) < 2^{-k} \text{ and } d(f(a_n), f(c)) > \varepsilon\}$$

is nonempty. From this, one derives a contradiction to Lemma 14 similarly as in the proof of Theorem 15. □

Let us have a look at the previous three results. Let us fix spaces X, X' as in Theorems 15 and 16, a BM–computable function $f :\subseteq X \to X'$, a sequence $(x_n)_n$ in X with $x_n \in \operatorname{dom} f$, for all n, and an element $x_\infty \in \operatorname{dom} f$. Note that under these assumptions Theorem 15 is a strengthening of Lemma 14. Theorem

15 says that if the sequence $(x_n)_n$ is computable and converges to x_∞, then the sequence $(f(x_n)_n$ must also converge, and, in fact, to the point $f(x_\infty)$. Theorem 16 says that in the previous statement one can omit the condition that the sequence $(x_n)_n$ is computable, if there is a dense, computable sequence in the domain of f. In Lemma 14 we considered computable, computably convergent sequences $(x_n)_n$ and derived convergence of the sequence $(f(x_n)_n$. Can one say something about the speed of convergence in this case? The next theorem, which seems to be new, says that indeed then the sequence $(f(x_n))_n$ must converge computably as well. Note that Theorems 15 and 16 were statements of the type that by adding an assumption to the assumptions in Lemma 14 one could deduce a stronger conclusion. The following theorem is a strengthening of Lemma 14 without any additional assumption.

Theorem 17. *Let (X, ν) be a BM–weakly complete metric space, and let (X', ν') be a BM–computable metric space. Let $f :\subseteq X \to X'$ be a BM–computable function. Let $(x_n)_n$ be a ν–computable sequence in X with $x_n \in \operatorname{dom} f$ for all n. If $(x_n)_n$ converges computably to an element $x_\infty \in \operatorname{dom} f$, then the sequence $(f(x_n))_n$ converges computably to $f(x_\infty)$.*

For the proof we need the following lemma.

Lemma 18. *Let $(r_n)_n$ be a computable sequence of real numbers converging to 0, and let $(q_n)_n$ be a computable sequence of rational numbers with $|r_n - q_{\langle n,i\rangle}| \leq 2^{-i}$, for all $n, i \in \mathbb{N}$. Then, the set $A \subseteq \mathbb{N}$ defined by*

$$A := \{\langle i, m\rangle \mid \exists n \geq m.\ |q_{\langle n,i\rangle}| > 2 \cdot 2^{-i}\}$$

is computably enumerable. It is decidable, iff the sequence $(r_n)_n$ converges computably.

Proof. It is clear that the set A is computably enumerable.
First, let us assume that $(r_n)_n$ converges computably. Then it has a computable modulus of convergence $e : \mathbb{N} \to \mathbb{N}$. In order to decide whether a number $\langle i, m\rangle$ is in A, first one calculates $e(i)$. Then, one checks whether there is a number n with $m \leq n < e(i)$ and $|q_{\langle n,i\rangle}| > 2 \cdot 2^{-i}$. This is sufficient, because from $|r_n| \leq 2^{-i}$ for all $n \geq e(i)$, one concludes $|q_{n,i}| \leq 2 \cdot 2^{-i}$ for all $n \geq e(i)$.
For the other direction, let us assume that A is decidable. We define $e : \mathbb{N} \to \mathbb{N}$ by

$$e(i) := \min\{m \mid \langle i + 3, m\rangle \notin A\}.$$

This function is well–defined because the sequence $(r_n)_n$ converges to 0. It is computable because A is decidable. It is a computable modulus of convergence for the sequence $(r_n)_n$ because $|q_{\langle n,i+3\rangle}| \leq 2 \cdot 2^{-i-3}$ for all $n \geq e(i)$, hence,

$$\begin{aligned} |r_k - r_l| &\leq |r_k - q_{\langle k,i+3\rangle}| + |q_{\langle k,i+3\rangle} - q_{\langle l,i+3\rangle}| + |q_{\langle l,i+3\rangle} - r_l| \\ &\leq 2^{-i-3} + 2 \cdot 2 \cdot 2^{-i-3} + 2^{-i-3} \\ &< 2^{-i}, \end{aligned}$$

for all $k, l \geq e(i)$. □

Proof (of Theorem 17). Let us assume that $(x_n)_n$ converges computably to an element $x_\infty \in \operatorname{dom} f$. By Lemma 14, the sequence $(f(x_n))_n$ converges to $f(x_\infty)$. Since f is BM–computable and the metric space X' is BM–computable, the sequence $(r_n)_n$ defined by $r_n := d(f(x_\infty), f(x_n))$ is a computable sequence of real numbers. It converges to 0. For $(r_n)_n$, fix a computable sequence $(q_n)_n$ of rational numbers as in Lemma 18 and define the set $A \subseteq \mathbb{N}$ as in Lemma 18. By Lemma 18, the set A is computably enumerable. We define a function $h :\subseteq \mathbb{N} \to \mathbb{N}$ by $\operatorname{dom} h := A$, and for $\langle i, m\rangle \in A$,

$$h(\langle i, m\rangle) := \min\{n \geq m \mid |q_{\langle n,i\rangle}| > 2 \cdot 2^{-i}\}$$

This function is computable. Furthermore, the set

$$\{\langle n, k\rangle \mid n \in \operatorname{dom} h \text{ and } h(n) \leq k\}$$

is decidable. By Lemma 12.2 the sequence $(z_n)_n$ defined by:

$$z_n := \begin{cases} x_\infty & \text{if } n \notin \operatorname{dom} h, \\ x_{h(n)} & \text{if } n \in \operatorname{dom} h \end{cases}$$

is a computable sequence in X. Since the function f is BM–computable, also the sequence $(f(z_n))_n$ must be computable. Since X' is a BM–computable metric space, the sequence $(d(f(x_\infty), f(z_n)))_n$ is a computable sequence of real numbers. We observe:

$$d(f(x_\infty), f(z_{\langle i,m\rangle})) \begin{cases} = 0 & \text{if } \langle i, m\rangle \notin A, \\ > 2^{-i} & \text{if } \langle i, m\rangle \in A. \end{cases}$$

This implies that A is decidable. Hence, by Lemma 18 the sequence $(r_n)_n$ converges computably to 0. This implies that the sequence $(f(x_n))_n$ converges computably to $f(x_\infty)$. □

Note that in the previous three theorems the first space, (X, ν), did not need to be BM–computable. It was only assumed to be BM–weakly complete and, in Theorems 15 and 16, BM–semi–computable.

Finally, let us make some comments on the relation between BM–computable functions and computable functions. Let (X, ν) and (X', ν') be numbered sets. In Lemma 4 we already mentioned that any (ν, ν')–computable function is also BM–(ν, ν')–computable. In fact, any of the computability notions in Definition 3 implies the corresponding BM–computability notion in Definition 2. For the converse, let us first assume that the numbering ν has a computably enumerable domain of definition.

Proposition 19. *Let (X, ν) and (X', ν') be numbered sets, and let $\operatorname{dom}\nu$ be computably enumerable. A set $Y \subseteq X$ is BM–ν–computably enumerable iff it is ν–computably enumerable. A total function $f : X \to X'$ is BM–(ν, ν')–computable iff it is (ν, ν')–computable. And $\nu \leq_{\mathrm{BM}} \nu'$ is equivalent to $\nu \leq \nu'$.*

Proof. One direction of the assertions has been stated already in Lemma 4. For the other direction, first note that we can assume without loss of generality that $\operatorname{dom}\nu$ is nonempty. Let $g : \mathbb{N} \to \mathbb{N}$ be a total computable function with $\operatorname{range} g = \operatorname{dom}\nu$. Then the sequence $(x_n)_n$ with $x_n := \nu(g(n))$, for all n, is ν–computable. Let $Y \subseteq X$ be BM–ν–computably enumerable. Then the set $A := \{n \in \mathbb{N} \mid x_n \in Y\}$ is computably enumerable. Also the set $g(A)$ is computably enumerable. It is equal to $\nu^{-1}(Y)$. Hence, Y is ν–computably enumerable. Let $f : X \to X'$ be a total BM–(ν,ν')–computable function. Then the sequence $(f(x_n))_n$ is ν'–computable. Hence, there is a total computable function $h : \mathbb{N} \to \mathbb{N}$ with $f(x_n) = \nu'(h(n))$ for all n. We define a function $F :\subseteq \mathbb{N} \to \mathbb{N}$ with $\operatorname{dom} F = \operatorname{dom}\nu$ by

$$F(n) := h(\min\{i \in \mathbb{N} \mid g(i) = n\})$$

for all $n \in \operatorname{dom}\nu$. The function F is computable and satisfies $f\nu(n) = \nu' F(n)$ for all $n \in \operatorname{dom}(\nu)$. Hence, f is (ν,ν')–computable. Finally, assume that $\nu \leq_{\mathrm{BM}} \nu'$. This means that X is a subset of X' and that the embedding of X into X' is a BM–(ν,ν')–computable function. According to the assertion just shown, it is a (ν,ν')–computable function. This, in turn, is equivalent to $\nu \leq \nu'$. □

But a partial BM–(ν,ν')–computable function does not need to be (ν,ν')–computable. The following counterexample is due to Pour–El [12]. We consider the numbered, computable, weakly complete metric space $(\mathbb{N}, \mathrm{id}_{\mathbb{N}})$. Consider an immune subset $A \subseteq \mathbb{N}$ (a subset $A \subseteq \mathbb{N}$ is called *immune*, iff it is infinite and does not contain any infinite computably enumerable subset; see Rogers [13]). Any function $f :\subseteq \mathbb{N} \to \mathbb{N}$ with $\operatorname{dom} f = A$ is BM–$(\mathrm{id}_{\mathbb{N}}, \mathrm{id}_{\mathbb{N}})$–computable. But only countably many of these uncountably many functions can be $(\mathrm{id}_{\mathbb{N}}, \mathrm{id}_{\mathbb{N}})$–computable.

Finally, we note that in case $\operatorname{dom}\nu$ is not computably enumerable, it can happen that the two computability notions for functions do not even coincide for total functions. A result by Friedberg [4] combined with the continuity result by Kreisel et. al. [6] and Ceĭtin [2,3] shows that there exists a total BM–$(\varphi|^{R^{(1)}}, \mathrm{id}_{\mathbb{N}})$–computable but not $(\varphi|^{R^{(1)}}, \mathrm{id}_{\mathbb{N}})$–computable function $f : R^{(1)} \to \mathbb{N}$, defined on the computable, weakly complete, and computably separable metric space $(R^{(1)}, \varphi|^{R^{(1)}})$ and with range in the computable, weakly complete and computably separable metric space $(\mathbb{N}, \mathrm{id}_{\mathbb{N}})$; compare also Rogers [13, §15.3]. Note that $\operatorname{dom}(\varphi|^{R^{(1)}})$ is not computably enumerable.

Acknowledgements

I would like to thank Vasco Brattka and Klaus Weihrauch for interesting discussions on topics related to this paper, and Jeffery Zucker for valuable comments on an earlier version of the paper.

References

1. S. Banach and S. Mazur. Sur les fonctions calculables. *Ann. Soc. Pol. de Math.*, 16:223, 1937.
2. G.S. Ceĭtin. Algorithmic operators in constructive complete separable metric spaces. *Doklady Akad. Nauk*, 128:49–52, 1959. (in Russian).
3. G.S. Ceĭtin. Algorithmic operators in constructive metric spaces. *Tr. Mat. Inst. Steklov*, 67:295–361, 1962. (in Russian, English trans. in AMS Trans. 64, 1967).
4. R.M. Friedberg. 4–quantifier completeness: A Banach–Mazur functional not uniformly partial recursive. *Bulletin de l'Academie Polonaise des Sciences, Série des sci. math., astr. et phys.*, 6(1):1–5, 1958.
5. A. Grzegorczyk. Some approaches to constructive analysis. In A. Heyting, editor, *Constructivity in mathematics*, Studies in Logic and The Foundations of Mathematics, pages 43–61, Amsterdam, 1959. North-Holland. Colloquium at Amsterdam, 1957.
6. G. Kreisel, D. Lacombe, and J.R. Shoenfield. Partial recursive functionals and effective operations. In A. Heyting, editor, *Constructivity in Mathematics*, Studies in Logic and The Foundations of Mathematics, pages 290–297, Amsterdam, 1959. North-Holland. Proc. Colloq., Amsterdam, Aug. 26–31, 1957.
7. A.H. Lachlan. Effective operations in a general setting. *The Journal of Symbolic Logic*, 29:163–178, 1964.
8. A.A. Markov. On the continuity of constructive functions (Russian). *Uspekhi Mat. Nauk (N.S.)*, 9:226–230, 1954.
9. A.A. Markov. On constructive functions. *Trudy Mat. Inst. Steklov.*, 52:315–348, 1958. (in Russian, English trans. in AMS Trans. (2) 29, 1963).
10. S. Mazur. *Computable Analysis*, volume 33. Rozprawy Matematyczne, Warsaw, 1963.
11. Y.N. Moschovakis. Recursive metric spaces. *Fundamenta Mathematicae*, 55:215–238, 1964.
12. M.B. Pour-El. A comparison of five "computable" operators. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6:325–340, 1960.
13. H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

A Generic Root Operation for Exact Real Arithmetic

Namhyun Hur* and James H. Davenport

Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, UK
mapnh@maths.bath.ac.uk, jhd@maths.bath.ac.uk

Abstract. We describe a generic root operation for exact real arithmetic.

1 Introduction

Numerical exact real arithmetic models such as the Lazy B-adic model [9] and the Linear Fractional Transformation (LFT) model [10] provide us ways of computing real numbers exactly. But these models normally provide only algorithms for the square root and the n-th root except the work reported in [2]. This paper describes a method for computing general radical expressions using the lazy B-adic model. We represent real algebraic numbers as a pair of a numerical representation and a symbolic representation as described in [6], which gave an equality algorithm for real algebraic numbers. In summary, [6] showed how to decide equality for real algebraic numbers using exact numerical information. This time we reverse the situation and use the symbolic information to derive arbitrary numerical accuracy of general radical expressions. In this paper we chose the lazy B-adic model (with $B = 2$, hence dyadic) for the numerical part but obviously one can also use the LFT.

This paper is organised as follows. In section 2 we give a brief summary of the lazy dyadic exact real arithmetic. In section 3 we mention the pair model and its arithmetic operations briefly. In section 4 we define a generic root operation (which we will call `rootOf`) of which special cases are the square root and the n-th root function. We then apply `rootOf` to few special cases: finite summation and product of radicals. We finish with few remarks.

2 Dyadic Exact Real Arithmetic

The lazy dyadic exact real arithmetic is an exact real arithmetic based on the concept of computable real numbers and their finite representations. The set of all real numbers is uncountable while the set of computable real numbers ($\mathbb{R}_C$) is countably infinite. The set of all real algebraic numbers is a subset of $\mathbb{R}_C$. The

* The first author was partially supported during the writing of this paper by the OpenMath project (Esprit project 24969).

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 82–87, 2001.

dyadic exact real arithmetic represents a number $x \in \mathbb{R}_C$ by a recursive function $f_x : \mathbb{N} \to \mathbb{Z}$ such that

$$|f_x(n) - 2^n x| < 1.$$

One can also define all the usual operations on them including many transcendental functions [9][5]. For example, addition (note $B = 2$ in our case) can be defined as below which satisfies the bound condition above (note that $\lfloor\ \rceil$ below denotes a certain rounding to a nearest integer):

$$f_{x+y}(n) := \left\lfloor \frac{f_x(n+2) + f_y(n+2)}{4} \right\rceil .$$

and the square root operation can be defined as

$$f_{\sqrt{x}}(n) := \begin{cases} \lfloor \sqrt{f_x(2n)} \rfloor & \text{if } f_x(2n) \geq 0 \\ \text{fail} & \text{otherwise} \end{cases}$$

where the $\sqrt{}$ on the right-hand side is a predefined square root operation on integers. Note that we may have to evaluate the $\sqrt{}$ function up to $2n$ precision to get n accurate digits.

3 Real Algebraic Numbers as Pairs

Let $\mathbb{R}_A$ be the set of all real algebraic numbers. We represent a real algebraic number $x \in \mathbb{R}_A$ by a pair of its dyadic real $f_x(n) \in R_C^{DR}$ and the corresponding square-free[1] polynomial $p(x) \in \mathbb{Z}[x]$.

Definition 1. *A real algebraic number is a pair $[f_x(n),\ p(x)]$ where $f_x(n) \in \mathbb{R}_C^{DR}$ is a dyadic representation of x and $p(x)$ is a square-free polynomial $\in \mathbb{Z}[x]$ such that $p(x) = 0$.*

We will write $\bar{x}$ to denote a pair representation of $x \in \mathbb{R}_A$. For example, an integer $k \in \mathbb{R}_A$ is represented by $\bar{k} = [f_k(n), x - k]$ where $f_k(n)$ is a dyadic representation of k (in fact $f_k(n) := 2^n k$) and $x - k$ is a square-free polynomial corresponding to k.

3.1 Elementary Operations

The elementary operations are quite straightforward to define. If $\bar{x} = [f_x(n), p(x)]$ and $\bar{y} = [g_y(n), q(y)]$ are two real numbers in our model, then the pair operations (denoted by $\oplus, \ominus, \otimes$ and $\oslash$) are defined as below. Note that $\hat{+}$, $\hat{-}$, $\hat{\times}$ and $\hat{/}$ denote the operations of dyadic real arithmetic and *res* denotes *resultant*. The resultant

[1] This is a deliberate choice on our part. Not insisting on square-free polynomials makes root isolation harder (essentially the root isolation has to do square-free decomposition) while insisting on irreducible polynomials can require full factorisation, and we may well have badly behaved (generalised Swinnerton-Dyer) polynomials [7].

of two polynomials is defined as the determinant of the Sylvester matrix formed from them and is useful for checking whether they have a non-constant common factor [3]. The resultant calculations do not necessarily give minimal or even square-free polynomials so we need a *refinement* operation which we denoted as R.

$$\begin{aligned}
\bar{x} \oplus \bar{y} &:= [f_x(n)\hat{+}g_y(n), \\
&\quad R(res(res(z-(x+y), p(x), x), q(y), y))], \\
\bar{x} \ominus \bar{y} &:= [f_x(n)\hat{-}g_y(n), \\
&\quad R(res(res(z-(x-y), p(x), x), q(y), y))], \\
\bar{x} \otimes \bar{y} &:= [f_x(n)\hat{\times}g_y(n), \\
&\quad R(res(res(z-xy, p(x), x), q(y), y))], \\
\bar{x} \oslash \bar{y} &:= [f_x(n)\hat{/}g_y(n), \\
&\quad R(res(res(yz-x, p(x), x), q(y), y))].
\end{aligned}$$

Note that the resultant calculation and square-free factorisation are standard parts of almost all algebraic number packages [8] and we implemented our pair model in Axiom[2]

Below is a simple example of $\sqrt{2}+\sqrt{3}$ in Axiom. Note that $\sqrt{2} = [f_2(n), x^2-2]$ and $\sqrt{3} = [f_3(n), y^2-3]$. Note that we only show five decimal digits of the numerical value for convenience. The **?** is the Axiom's symbol for a variable.

```
(1) -> a:= sqrt(2)$PAIR
                      2
   (1)  ["+1.41421",?  - 2]                        Type: PAIR

(2) -> b:= sqrt(3)$PAIR
                      2
   (2)  ["+1.73205",?  - 3]                        Type: PAIR

(3) -> a + b
                      4      2
   (3)  ["+3.14627",?  - 10?  + 1]                 Type: PAIR
```

4 A Generic Root Operation

As pointed out in [6] we can replace the numerical part of a pair by a numerical algorithm directly approximating it as the corresponding root of the symbolic part. For example, the numerical part $3.14627\ldots$ of $\sqrt{2}+\sqrt{3}$, is the same as `rootOf`$(x^4-10x^2+1, 3)$ assuming that we have such operation[3]. We believe that we need this generic root operation for following reasons:

[2] Axiom is a generic computer algebra system. Axiom is a trademark of NAG Limited.
[3] The second argument of `rootOf` is necessary as the initial value of Newton iteration.

- (importance) As Abel showed, we can not solve polynomial equations of order bigger than 4 in terms of radicals. For example, `rootOf`$(x^5+x+1,-1)$ can't be done any other way.
- (simplicity) It is often troublesome to input a complicated radical expressions. As a simple example, `rootOf`$(x^2-210,14)$ is much simpler to type than `(sqrt(2)*sqrt(3)*sqrt(5)*sqrt(7))$PAIR`.
- (efficiency) For those radical expressions where its defining polynomial's total degree and the size of coefficients are small, it is much faster than the lazy dyadic approach, although we have to be more precise what we mean by *those radical expressions where its defining polynomial's degree and the number of coefficients are small.*

We can implement `rootOf` operation in two ways: one using bisection + Sturm sequences and the other using the Newton's iteration. Here we describe the version using the Newton's method. Note that one of the key problem in applying Newton's iteration (in numerical mathematics) is to find the initial starting point. But we don't have this difficulty thanks to our pair representation. Basing on the observation that $3.14627\ldots$ is the same as `rootOf`$(x^4-10x^2+1,3)$ we chose the type of `rootOf` as $(\mathbb{Z}[x], \text{LazyDyadicReal}) \to \text{LazyDyadicReal}$[4] and the algorithm is quite simple. Given a pair of polynomial and an approximation $(p(x), y)$, `rootOf` does:

1. checks whether the $p(x)$ has one, and only one, root in the interval given by the second argument, i.e., $(y(0)-e, y(0)+e)$, where e is the allowed error bound which we can choose, say 1. If no, then report error, otherwise proceed.
2. Newton-iterate with
   - starting value: $y(0)$.
   - stop condition: $|(y(0) - \frac{p(y(0))}{p'(y(0))}) - y(0)| < \frac{1}{2^n}$.
3. return $\lfloor 2^n \times$ (the result of step 2)$\rfloor$.

5 Finite Product of Simple Radicals

Here we are interested in finite products of simple radicals, i.e., $\prod_{i=0}^{m-1} \sqrt{k_i}$ (case 1), $\prod_{i=0}^{m-1} \sqrt[n]{k_i}$ (case 2) or combination of these two.

If we evaluate, say $\sqrt{2}*\sqrt{3}*\sqrt{5}*\sqrt{7}$, using the usual `sqrt` operation of the lazy dyadic arithmetic, then we need to perform four (lazy dyadic) multiplications, which is costly, and also we may have to evaluate up to twice of the given precisions for each of the `sqrt` operation.

- (case 1) $\prod_{i=0}^{m-1} \sqrt{k_i}$, where for each i, k_i are positive integers. In this case the polynomial is of the form $x^2 - c$ where $c = \prod_{i=0}^{m-1} k_i$. In this case, `rootOf` is much faster than the lazy dyadic approach.

[4] In our implementation, the polynomial is of Axiom type `SUP Z`, SUP being the abbreviation of SparseUnivariatePolynomial, `Z` being Integer, and the `LazyDyadicReal` is the domain name for the implementation of lazy dyadic real arithmetic in Axiom.

– (case 2) $\prod_{i=0}^{m-1} \sqrt[n]{k_i}$, where $n > 2$ and for each i, k_i are positive integers. Again `rootOf` performs much better than the lazy dyadic approach.

6 Finite Summation of Simple Radicals

This is a thorny case and our `rootOf` operation seems worse than the ordinary combination of the square root and n-th root. The main reason for this is that the resulting polynomial corresponding to the numerical part usually has a sizable total degree and also has several non-zero coefficients. In these cases it seems that the original lazy dyadic approach is much faster than our `rootOf`. Below is an example of Axiom session evaluating $\sqrt{2}+\sqrt{3}+\sqrt{5}+\sqrt{7}$. Notice the enormous time difference between the two approaches.

– (Lazy Dyadic Approach)

```
(30) -> (sqrt(2)+sqrt(3)+sqrt(5)+sqrt(7))$LDR

   (30)  "+8.02808"                                    Type: LDR
```

Evaluation (in lazy dyadic approach) of the same expression to many precisions, say hundreds, does not cost much extra.

– (Pair Approach) To see what kind of polynomial we are talking about we evaluate the same expression in pair model.

```
(31) -> (sqrt(2)+sqrt(3)+sqrt(5)+sqrt(7))$PAIR

   (31)
   ["+8.02808",

        16       14        12          10           8
       ?   - 136?   + 6476?   - 141912?   + 1513334?
                         6            4           2
              - 7453176?  + 13950764?  - 5596840?   + 46225]

                                                    Type: PAIR
        Time: 0.07 (IN) + 0.08 (EV) + 0.05 (OT) = 0.20 sec
```

The polynomial is of total degree 73 and the coefficients are quite large. This tends to make the intermediate rational numbers huge in the Newton iteration. How to make intermediate calculations less is a topic for future research.

```
(34) -> rootOf(sp %% 28,8::LDR)

   (34)  "+8.02808"
                                                    Type: LDR
        Time: 0.07 (EV) + 174.90 (OT) + 0.64 (GC) = 175.61 sec
```

7 Conclusion

We gave an algorithm for a generic root operation for exact real arithmetic. The generic `rootOf` operation shows much better timing performance in the case of finite products of simple radicals than the lazy dyadic approach. Certainly we need to refine our implementation in several ways. First we may be able to reduce the evaluation precision inside the Newton iteration procedure. Second we can make the implementation more user-friendly by allowing the user to specify the error bound. Also, using bisection, instead of Newton iteration, might be useful in splitting the intervals.

Acknowledgement

We'd like to thank the anonymous referees for constructive comments.

References

1. H.J. Boehm, R. Cartwright, M.J. O'Donnel, and M. Riggle, *Exact real arithmetic: A case study in higher order programming*, Proceedings of the 1986 ACM Conference on LISP and Functional Programming, ACM, 1986, pp. 162-173.
2. A. Edalat and F. Rico, *Two algorithms for root finding in exact real arithmetic*, Third Real Numbers and Computers Conference, pp. 27-44, 1998.
3. J.H. Davenport, *Computer algebra for cylindrical algebraic decomposition*, Technical Report 88-10, Department of Mathematical Sciences, University of Bath, Claverton Down, Bath BA2 7AY, England.
4. J. H. Davenport, Y. Siret, and E. Tournier, *Computer Algebra*, 2nd edition, Academic Press, 1993.
5. John R. Harrison, *Introduction to functional programming*, Available from the web: `http://www.cl.cam.ac.uk/~jrh`.
6. N. Hur and J.H. Davenport, *An Exact Real Algebraic Arithmetic with Equality Determination*, Proceedings of ISSAC 2000, pp. 169-174.
7. Kaltofen, E., Musser, D.R., and Saunders, B.D., *A Generalized Class of Polynomials That are Hard to Factor*, SIAM J. Comp., pp. 473-483, 12(1983).
8. R. Loos, *Computing in algebraic extensions*, Computing, **4**(suppl.), pp. 173-187.
9. V. Ménissier-Morain, *Arithmétique exacte: conception, algorithmique et performances d'une impleméntation informatique en précision arbitraire*, PhD Thesis, Université Paris 7, December 1994.
10. P.J. Potts, *Exact real arithmetic using Möbius transformations*, PhD Thesis, Imperial College, July 1998.
11. A.W. Strzeboński, *Computing in the field of complex algebraic numbers*, Journal of Symbolic Computation, **24** 1997, pp. 647-656.

Effective Contraction Theorem and Its Application

Hiroyasu Kamo*

Nara Women's University
wd@ics.nara-wu.ac.jp

Abstract. A contraction on a complete metric space has a unique fixed point. This fact is called the contraction theorem and of wide application. In this paper, we present an effective version of the contraction theorem. We show that if the contraction is a computable function on an effectively locally compact metric space, then the fixed point is a computable point on the space. Many facts on computability can be proved by using the effective contraction theorem. We give, in this paper, three examples, the effective implicit function theorem, the result on computability of self-similar sets by Kamo and Kawamura, and computability of the Takagi function.

1 Introduction

In this paper, we introduce an effective version of the contraction theorem.

Let (X, d) be a metric space. A function $f : X \to X$ is called a contraction if there exists a real L with $0 < L < 1$ such that $d(f(x), f(y)) \leq Ld(x, y)$ for any $x, y \in X$. If (X, d) is a complete metric space and $f : X \to X$ is a contraction, then there exists a unique fixed point of f. This theorem is known as the *contraction theorem* and of wide application.

Mori, Tsujii, and Yasugi investigated effective total boundedness, an effective counterpart of total boundedness, and reached to effectively compact metric spaces and effectively σ-compact metric spaces [6][11]. An effectively compact metric space is a complete and effectively totally bounded metric space. An effectively σ-compact metric space is roughly an effective union of countably many effectively compact metric spaces. We introduce an effectively locally compact metric space as an effectively σ-compact metric space with an additional condition. Details are explained in §2.

We will show in §3 that if the contraction is a computable function on an effectively locally compact metric space, then the fixed point is a computable point on the space.

Many facts on computability can be proved by using the effective contraction theorem. We will give three examples in §4.

The first example is the effective implicit function theorem. The implicit function theorem is one of the answers to the problem: for a given binary function F, find a unary function f such that $F(x, f(x)) = 0$. We will introduce

* This work has been supported in part by the Scientific Grant of Japan No. 12640120.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 88–100, 2001.

an effective version of the implicit function theorem and prove it by using the effective contraction theorem.

The second example is the result on computability of self-similar sets by Kamo and Kawamura. Hutchinson [4] proved a theorem on existence of self-similar sets and Hata [3] generalized it. Kamo and Kawamura [5] added the viewpoint of computability to it. We will rewrite Kamo and Kawamura's proof by using the effective contraction theorem.

The third example is computability of the Takagi function. The Takagi function is a continuous but nowhere differentiable function discovered by Takagi [9] in 1903. The van der Waerden function (discovered by van der Waerden in 1930) is an analogous function. It is easy to prove the computability directly from the definition of the Takagi function. We, however, will prove the computability by using the effective contraction theorem. What we wish to show as the third example is not only the computability itself but also the fact that the computability is an application of the effective contraction theorem.

2 Preliminary

We follow the terminology and the notation in [11] except for some minor modifications. Especially, we start natural numbers with 0. We refer to [7] for computability of reals and real functions. We refer to [10] for Type 2 computability.

We abbreviate an ω-sequence to a sequence, an ω^2-sequence to a double sequence, an ω^k-sequence to a k-tuple sequence, etc. We often identify a double sequence $(x_{m,n})$ with a sequence (x_n) such that $x_{m,n} = x_{\langle\langle m,n\rangle\rangle}$ where $\langle\langle m,n\rangle\rangle = m+(m+n)(m+n+1)/2$. This identification is applicable to k-tuple sequences by using a standard construction of tupling from a pairing, $\langle\langle n_1\rangle\rangle = n_1$, $\langle\langle n_1,\ldots,n_k,n_{k+1}\rangle\rangle = \langle\langle\langle\langle n_1,\ldots,n_k\rangle\rangle,n_{k+1}\rangle\rangle$.

If $\varphi : \mathbb{N} \to \mathbb{N}$ is a recursive function, $(x_{\varphi(n)})$ is said to be an effectively selected subsequence of (x_n).

Let (M,d) be a metric space. We write $B(a,\varepsilon) = \{x \in M \mid d(a,x) < \varepsilon\}$ and $\bar{B}(a,\varepsilon) = \{x \in M \mid d(a,x) \leq \varepsilon\}$. A double sequence $(x_{n,k})$ is said to converge to (x_n) effectively in n and k as $k \to \infty$ if there exists a recursive function $\psi : \mathbb{N}^2 \to \mathbb{N}$ such that, for any $p,n,k \in \mathbb{N}$, it holds that $k \geq \psi(n,p)$ implies $d(x_{n,k},x_n) \leq 2^{-p}$. A double sequence $(x_{n,k})$ is said to converge to (x_n) uniformly in n and exponentially in k as $k \to \infty$ if for any $n,k \in \mathbb{N}$, it holds that $d(x_{n,k},x_n) \leq 2^{-k}$. A double sequence $(x_{n,k})$ is said to be an effective Cauchy sequence as $k \to \infty$ if there exists a recursive function $\varphi : \mathbb{N}^2 \to \mathbb{N}$ such that, for any $p,n,k,k' \in \mathbb{N}$, it holds that $\varphi(n,p) \leq k < k'$ implies $d(x_{n,k},x_{n,k'}) \leq 2^{-p}$.

If $(x_{n,k})$ converges to (x_n) uniformly in n and exponentially in k as $k \to \infty$, then $(x_{n,k})$ converges to (x_n) effectively in n and k as $k \to \infty$. If $(x_{n,k})$ converges to (x_n) effectively in n and k as $k \to \infty$, then there exists an effectively selected subsequence $(x'_{n,k})$ of $(x_{n,k})$ such that $(x'_{n,k})$ converges to (x_n) uniformly in n and exponentially in k as $k \to \infty$.

If $(x_{n,k})$ converges effectively, then $(x_{n,k})$ is an effective Cauchy sequence. If an effective Cauchy sequence converges, it converges effectively. However an effective Cauchy sequence does not always converge.

We summarize here the definitions we will use in this paper.

Definition 1 (Computability Structure [6, Definition 5], [11, Definition 1.4]). *Let (M, d) be a metric space. A set $\mathcal{S}$ of sequences of points on M is a* computability structure *on (M, d) if the following three conditions hold.*

1. (Metric axiom) *If $(x_m), (y_n) \in \mathcal{S}$, then $(d(x_m, y_n))_{m,n}$ forms a computable double sequence of reals.*
2. (Subsequence axiom) *If $(x_n) \in \mathcal{S}$ and (x'_n) is an effectively selected subsequence of (x_n), then $(x'_n) \in \mathcal{S}$.*
3. (Limit axiom) *If $(x_{n,k}) \in \mathcal{S}$, $(x'_n) \in M^\omega$, and $(x_{n,k})$ converges to (x'_n) effectively in n and k as $k \to \infty$, then $(x'_n) \in \mathcal{S}$.*

A sequence in $\mathcal{S}$ is said to be a computable sequence. A point $x \in M$ is said to be a computable point if $(x)_n$, the sequence such that all of its elements equal to x, is a computable sequence.

Definition 2 (Effectively Locally Compact Metric Space). *A metric space with a computability structure $(M, d, \mathcal{S})$ is an* effectively locally compact metric space *if the following two conditions hold.*

1. (Completeness) *d is a complete metric.*
2. *There exist a sequence (K_k) of compact subsets and a computable double sequence $(\xi_{k,i}) \in \mathcal{S}$ such that the following three conditions holds.*
   (a) *There exist computable functions $\kappa : (\mathbb{N}^{\mathbb{N}})^2 \to \mathbb{N}$ and $\rho : (\mathbb{N}^{\mathbb{N}})^2 \to \mathbb{N}$ such that: for any $\boldsymbol{k}, \boldsymbol{i} \in \mathbb{N}^{\mathbb{N}}$, if $(\xi_{\boldsymbol{k}[n],\boldsymbol{i}[n]})$ converges exponentially in n as $n \to \infty$, then $\bar{B}(\lim_{n\to\infty} \xi_{\boldsymbol{k}[n],\boldsymbol{i}[n]}, 2^{-\rho(\boldsymbol{k},\boldsymbol{i})}) \subseteq K_{\kappa(\boldsymbol{k},\boldsymbol{i})}$.*
   (b) (Effective separability) *$\overline{\{\xi_{k,i} \mid i \in \mathbb{N}\}} = K_k$ for any $k \in \mathbb{N}$.*
   (c) (Effective σ-total boundedness) *There exists a recursive function $\sigma : \mathbb{N}^2 \to \mathbb{N}$ such that $\bigcup_{i<\sigma(k,p)} B(\xi_{k,i}, 2^{-p}) \supseteq K_k$ for any $k \in \mathbb{N}$.*

The definition of an effectively σ-compact metric space [11, Definition 4.2] contains the condition $\bigcup_{k\in\mathbb{N}} K_k = M$ instead of the condition *2a* in Definition 2, which is the only difference between the two definitions. Since the condition *2a* implies $\bigcup_{k\in\mathbb{N}} K_k = M$, an effectively locally compact metric space is an effectively σ-compact metric space.

Yasugi, Mori, and Tsujii have defined a computable function from an effectively σ-compact metric space to $\mathbb{R}$ [11, Definition 4.3]. We adapt their definition to apply to a function from an effectively locally compact metric space to an effectively locally compact metric space.

Definition 3 (Computable Function). *Let $(M, d, \mathcal{S})$ be an effectively locally compact metric space with (K_k) and $(\xi_{k,i})$. Let $(M', d', \mathcal{S}')$ be a metric space with a computability structure. A function $f : M \to M'$ is* computable *if the following two conditions hold.*

1. (Sequential computability) *For any* $(x_n) \in \mathcal{S}$*, it holds that* $(f(x_n)) \in \mathcal{S}'$.
2. (Effective uniform continuity) *There exists a recursive function* $\psi : \mathbb{N}^2 \to \mathbb{N}$ *such that for any* $k \in \mathbb{N}$ *and* $x, y \in K_k$*, it holds that* $d(x, y) \leq 2^{-\psi(k,p)}$ *implies* $d'(f(x), f(y)) \leq 2^{-p}$.

The Euclidean line $\mathbb{R}$ with computable sequences of reals satisfies Definition 2. The computability of real functions by Definition 3 coincides with the usual computability.

For a q-tuple of effectively locally compact metric spaces $(M_1, d_1, \mathcal{S}_1)$, ..., $(M_q, d_q, \mathcal{S}_q)$, construct $(M, d, \mathcal{S})$ by:

$$\begin{gathered} M = M_1 \times \cdots \times M_q, \\ d((x_1, \ldots, x_q), (x'_1, \ldots, x'_q)) = \max\{d_1(x_1, x'_1), \ldots, d_q(x_q, x'_q)\}, \\ \left((x_1^{(n)}, \ldots, x_q^{(n)})\right)_n \in \mathcal{S} \iff (x_1^{(n)})_n \in \mathcal{S}_1 \wedge \cdots \wedge (x_q^{(n)})_n \in \mathcal{S}_q. \end{gathered}$$

Then $(M, d, \mathcal{S})$ forms an effectively locally compact metric space, which we call the product space of $(M_1, d_1, \mathcal{S}_1)$, ..., $(M_q, d_q, \mathcal{S}_q)$. If $(M_1, d_1, \mathcal{S}_1)$, ..., $(M_q, d_q, \mathcal{S}_q)$ are effectively locally compact metric spaces with $(K_1^{(k)})$, $(\xi_1^{(k,i)})$, ρ_1, κ_1, σ_1, ..., $(K_q^{(k)})$, $(\xi_q^{(k,i)})$, ρ_q, κ_1, σ_q, respectively, then $(M, d, \mathcal{S})$ is an effectively locally compact metric space with (K_k), $(\xi_{k,i})$, ρ, κ, σ defined by:

$$\begin{gathered} K_k = K_1^{(\pi_1^q(k))} \times \cdots \times K_q^{(\pi_1^q(k))}, \\ \xi_{k,i} = (\xi_1^{(\pi_1^q(k), \pi_1^q(i))}, \ldots, \xi_q^{(\pi_q^q(k), \pi_q^q(i))}), \\ \rho(\varphi) = \max\{\rho_1((\pi_1^q \times \pi_1^q) \circ \varphi), \ldots, \rho_q((\pi_q^q \times \pi_q^q) \circ \varphi)\}, \\ \kappa(\varphi) = \langle\langle \kappa_1((\pi_1^q \times \pi_1^q) \circ \varphi), \ldots, \kappa_q((\pi_q^q \times \pi_q^q) \circ \varphi)\rangle\rangle, \\ \sigma(k, p) = \max\{\sigma_1(\pi_1^q(k), p), \ldots, \sigma_q(\pi_q^q(k), p)\} \end{gathered}$$

where $\pi_1^q, \ldots, \pi_q^q$ denote the projections.

Definition 4 (Effectively Compact Metric Spaces). *A metric space with a computability structure* $(M, d, \mathcal{S})$ *is an* effectively compact metric space *if the following two conditions hold.*

1. (Completeness) d *is a complete metric.*
2. *There exist a sequence* $(\xi_i) \in \mathcal{S}$ *such that the following two conditions hold.*
   (a) (Effective separability) $\overline{\{\xi_i \mid i \in \mathbb{N}\}} = M$.
   (b) (Effective total boundedness) *There exists a recursive function* $\sigma : \mathbb{N} \to \mathbb{N}$ *such that* $\bigcup_{i<\sigma(p)} B(\xi_i, 2^{-p}) = M$ *for any* $k \in \mathbb{N}$.

If $(M, d, \mathcal{S})$ is an effectively compact metric space with (ξ_i), then it is an effectively locally compact metric space with $(M)_k$ and $(\xi_i)_{k,i}$. We consider an effectively compact metric space a special case of effectively locally compact metric spaces.

3 Effective Contraction Theorem

We denote by $\mathrm{Fix}\, f$ the unique fixed point of a contraction f.

We use the following three lemmata in this section.

Lemma 1 (Effective Completeness [11, Proposition 1.4]). *Let $(M, d, \mathcal{S})$ be an effectively locally compact metric space. If a computable double sequence $(x_{n,k})$ is an effective Cauchy sequence as $k \to \infty$, then there exists a computable sequence (x_n) such that $(x_{n,k})$ converges to (x_n) as $k \to \infty$ effectively in n and k.*

Lemma 2 (Effective Density Lemma [6, Proposition 1.2]). *Let $(M, d, \mathcal{S})$ be an effectively locally compact metric space with (K_k) and $(\xi_{k,i})$. For a sequence $(x_n) \in M^\omega$, the following three conditions are equivalent.*

1. *(x_n) is a computable sequence.*
2. *There exists a recursive function $\varphi : \mathbb{N}^2 \to \mathbb{N}^2$ such that $(\xi_{\varphi(n,l)})$ converges to (x_n) effectively in n and l as $l \to \infty$.*
3. *There exists a recursive function $\varphi : \mathbb{N}^2 \to \mathbb{N}^2$ such that $(\xi_{\varphi(n,l)})$ converges to (x_n) uniformly in n and exponentially in l as $n \to \infty$.*

Lemma 3 (Iteration). *Let $(M, d, \mathcal{S})$ be an effectively locally compact metric space with (K_k) and $(\xi_{k,i})$. If $a \in M$ is a computable point and $f : M \to M$ is a computable function, then $(f^n(a))_{n\in\mathbb{N}}$ forms a computable sequence of points.*

Proof. Since $d(\xi_{k,i}, \xi_{k',i'})$ forms a computable quadruple sequence of reals, there exists a recursive function $\alpha : \mathbb{N}^5 \to \mathbb{Q}$ such that $|d(\xi_{k,i}, \xi_{k',i'}) - \alpha(k, i, k', i', l)| \leq 2^{-l}$ for any $k, i, k', i', l \in \mathbb{N}$. Let $\kappa : (\mathbb{N}^2)^{\mathbb{N}} \to \mathbb{N}$ and $\rho : (\mathbb{N}^2)^{\mathbb{N}} \to \mathbb{N}$ be computable functions such that: for any $\varphi : \mathbb{N} \to \mathbb{N}^2$, if $(\xi_{\varphi(n)})$ converges exponentially in n as $n \to \infty$, then $B(\lim_{n\to\infty} \xi_{\varphi(n)}, 2^{-\rho(\varphi)}) \subseteq K_{\kappa(\varphi)}$.

We use Lemma 2 (*1* $\Rightarrow$ *3*) twice. Since a is a computable point, there exists a recursive function $\varphi' : \mathbb{N} \to \mathbb{N}^2$ such that $d(\xi_{\varphi'(l)}, a) \leq 2^{-l}$ for any $l \in \mathbb{N}$. Since f is sequentially computable, $(f(\xi_{k,i}))$ forms a computable double sequence. Thus there exists a recursive function $\varphi'' : \mathbb{N}^3 \to \mathbb{N}^2$ such that $d(\xi_{\varphi''(k,i,l)}, f(\xi_{k,i})) \leq 2^{-l}$ for any $k, i, l \in \mathbb{N}$.

Since f is effectively uniformly continuous, there exists a recursive function $\psi : \mathbb{N}^2 \to \mathbb{N}$ such that for any $k \in \mathbb{N}$ and $x, y \in K_k$, it holds that $d(x, y) \leq 2^{-\psi(k,p)}$ implies $d'(f(x), f(y)) \leq 2^{-p}$.

We define a recursive function $\varphi : \mathbb{N}^2 \to \mathbb{N}^2$ recursively with auxiliary definitions of $\bar{k}_n, \bar{p}_n \in \mathbb{N}$ and $\theta_n : \mathbb{N} \to \mathbb{N}^2$ as follows.

$$\varphi(0, l) = \varphi'(l), \qquad \varphi(n+1, l) = \varphi''(\theta_n(l), l+1),$$
$$\bar{k}_n = \kappa(\lambda l \in \mathbb{N}.\, \varphi(n, l)), \qquad \bar{p}_n = \rho(\lambda l \in \mathbb{N}.\, \varphi(n, l)),$$
$$\theta_n(l) = \varphi(n, \max\{\psi(\bar{k}_n + 1, l+1), \bar{p}_n).$$

It is straightforward from the definition that φ is recursive. We will show by induction that φ is total and that $(\xi_{\varphi(n,l)})$ converges to $(f^n(a))$ uniformly

in n and exponentially in l as $l \to \infty$. It is trivial that $\varphi(0,l)$ is defined and $d(\xi_{\varphi(0,l)}, a) < 2^{-l}$ for any $l \in \mathbb{N}$. It remains to show that if $\varphi(n,l)$ is defined and $d(\xi_{\varphi(n,l)}, f^n(a)) < 2^{-l}$ for any $l \in \mathbb{N}$, then $\varphi(n+1,l)$ is defined and $d(\xi_{\varphi(n+1,l)}, f^{n+1}(a)) < 2^{-l}$ for any $l \in \mathbb{N}$.

From the induction hypothesis, we obtain $\bar{B}(f^n(a), 2^{-\bar{p}_n}) \subseteq K_{\bar{k}_n}$. Furthermore, we also obtain

$$\begin{aligned} d(\xi_{\theta_n(l)}, f^n(a)) &\le 2^{-\max\{\psi(\bar{k}_n+1,l+1),\bar{p}_n\}} \\ &\le 2^{-\bar{p}_n}. \end{aligned}$$

Therefore, $f^n(a), \xi_{\theta_n(l)} \in K_{\bar{k}_n}$.

On the other hand, we obtain

$$\begin{aligned} d(\xi_{\theta(l)}, f^n(a)) &\le 2^{-\max\{\psi(\bar{k}_n+1,l+1),\bar{p}_n\}} \\ &\le 2^{-\psi(\bar{k}_n+1,l+1)}. \end{aligned}$$

It concludes that

$$\begin{aligned} d(\xi_{\varphi(n+1,l)}, f^{n+1}(a)) &\le d(\xi_{\varphi''(\theta_n(l),l+1)}, f(\xi_{\theta_n(l)})) + d(f(\xi_{\theta_n(l)}), f(f^n(a))) \\ &\le 2^{-(l+1)} + 2^{-(l+1)} = 2^{-l}. \end{aligned}$$

Due to Lemma 2 (*3* $\Rightarrow$ *1*), this implies that $(f^n(a))_{n\in\mathbb{N}}$ forms a computable sequence of points. □

We are now ready to introduce the main theorem.

Theorem 1 (Effective Contraction Theorem). *Let $(M,d,\mathcal{S})$ be an effectively locally compact metric space. If $f : M \to M$ is a computable contraction, then* Fix f *is a computable point.*

Proof. Let L be a real such that $0 < L < 1$ and $d(f(x), f(y)) \le Ld(x,y)$ for any $x, y \in M$. We can assume without loss of generality that L is computable.

We start with a computable point $x_0 \in M$ and construct a sequence (x_n) by $x_{n+1} = f(x_n)$. Note that (x_n) converges to Fix f. Using Lemma 3, we obtain from computability of x_0 and f that (x_n) is a computable sequence. By induction on n, we have $d(x_n, x_{n+1}) \le L^n d(x_0, x_1)$. Hence

$$d(x_m, x_n) \le \frac{L^m d(x_0, x_1)}{1-L} \qquad \text{for } m < n,$$

which implies that (x_n) is an effective Cauchy sequence. Using Lemma 1, we obtain that Fix f is a computable point. □

The following form of the effective contraction theorem is often more convenient to check the computability of the fixed point.

Corollary 1. *Let $(M,d,\mathcal{S})$ be an effectively locally compact metric space with (K_k) and $(\xi_{k,i})$. If $f : M \to M$ is a contraction that maps $(\xi_{k,i})$ to a computable sequence, then* Fix f *is a computable point.*

Proof. It suffices to show that f is a computable function. From f being a contraction, it follows immediately that f is effectively uniformly continuous. What remains to show is that f is sequentially computable.

Let (x_n) be a computable sequence. Using Lemma 2, we obtain there exists a recursive function $\varphi : \mathbb{N}^2 \to \mathbb{N}^2$ such that $(\xi_{\varphi(n,l)})$ converges to (x_n) effectively in n and l as $l \to \infty$. Since $(f(\xi_{\varphi(n,l)}))$ is an effectively selected subsequence of $(f(\xi_{k,i}))$, it holds that $(f(\xi_{\varphi(n,l)}))$ is a computable sequence. Since f is a contraction, $(f(\xi_{\varphi(n,l)}))$ converges to $(f(x_n))$ effectively in n and l as $l \to \infty$. Due to Lemma 2, it follows that $(f(x_n))$ is a computable sequence. □

4 Application

4.1 Effective Implicit Function Theorem

If $F(x, y)$ is continuous and partially differentiable on y in a neighborhood Ω of a point (x_0, y_0), the partial derivative $F_y(x, y)$ is continuous in Ω, $F(x_0, y_0) = 0$, and $F_y(x_0, y_0) \neq 0$, then there exists a function f on a neighborhood I of x_0 such that $F(x, f(x)) = 0$ for any $x \in I$. This theorem is known as the implicit function theorem.

We will introduce the effective implicit function theorem and prove it by using the effective contraction theorem.

Theorem 2 (Effective Implicit Function Theorem). *Let $I \subset \mathbb{R} \times \mathbb{R}$ be a computable closed rectangle and (x_0, y_0) a computable point that is an interior point of I. Let $F : I \to \mathbb{R}$ be a computable function such that $F(x, y)$ is partially differentiable on y and the partial derivative F_y is computable. Suppose $F(x_0, y_0) = 0$ and $F_y(x_0, y_0) \neq 0$. Then there exist a computable closed interval $I' \subset \mathbb{R}$ and a computable function $f : I' \to \mathbb{R}$ such that x_0 is an interior point of I' and $F(x, f(x)) = 0$ for any $x \in I'$.*

Proof. Pick an arbitrary computable real L such that $0 < L < 1$.

Since F_y is a computable function, we can construct a computable rectangle $[x_0 - \delta, x_0 + \delta] \times [y_0 - \varepsilon, y_0 + \varepsilon] \subseteq I$ such that for all $(x, y) \in [x_0 - \delta, x_0 + \delta] \times [y_0 - \varepsilon, y_0 + \varepsilon]$, it holds that

$$|F_y(x, y) - F_y(x_0, y_0)| < L|F_y(x_0, y_0)|.$$

Since F is a computable function, we can construct a computable interval $I' \subseteq [x_0 - \delta, x_0 + \delta]$ such that x_0 is an interior point of I' and for all $x \in I'$, it holds that

$$|F(x, y_0)| < (1 - L)\varepsilon|F_y(x_0, y_0)|.$$

For any polynomial function p such that all of the coefficients are rational numbers and the constant term equals to 0, we construct a computable function $\xi : I' \to \mathbb{R}$ by:

$$\xi(x) = \begin{cases} y_0 - \varepsilon & \text{if } p(x - x_0) < -\varepsilon, \\ y_0 + p(x - x_0) & \text{if } -\varepsilon \leq p(x - x_0) \leq \varepsilon, \\ y_0 + \varepsilon & \text{if } p(x - x_0) > \varepsilon. \end{cases}$$

Pick arbitrarily (ξ_n) among effective enumerations of all functions of this form.

Now we construct an effectively compact metric space. Let $\mathcal{F}$ be the set of all continuous real-valued function f on I' such that $f(x_0) = y_0$ and for any $x \in I'$, it holds that $|f(x) - y_0| \leq \varepsilon$. It is trivial that $\mathcal{F}$ is a subset of $C(I')$. Thus $d(f,g) = \sup_{x \in I'} |f(x) - g(x)|$ is a metric on $\mathcal{F}$. It is easy to verify that any Cauchy sequence over $\mathcal{F}$ converges to a point in $\mathcal{F}$. Hence $(\mathcal{F}, d)$ is a complete metric space. Let $\mathcal{S}$ be the set of all sequences of functions in $\mathcal{F}$ that are computable as a sequence of functions from I' to $\mathbb{R}$. Then some manipulation with the effective Weierstrass approximation [7, Section 7 of Chapter 0] yields that $(\mathcal{F}, d, \mathcal{S})$ is an effectively compact metric space with (ξ_n).

Next we construct a computable contraction on $\mathcal{F}$. We define $\Phi : \mathcal{F} \to \mathcal{F}$ by:

$$\Phi(f)(x) = f(x) - \frac{F(x, f(x))}{F_y(x_0, y_0)}.$$

We should first show that Φ is well-defined. Namely, we should verify $\Phi(f) \in \mathcal{F}$ for any $f \in \mathcal{F}$. Since $F(x_0, y_0) = 0$, we have that $f(x_0) = y_0$ implies $\Phi(f)(x_0) = y_0$. Since F is continuous, we have that continuity of f implies continuity of $\Phi(f)$. By using the mean value theorem, we obtain that for any $f \in \mathcal{F}$ and $x \in I'$, there exists a real θ between $f(x)$ and y_0 such that

$$F(x, f(x)) = F(x, y_0) + F_y(x, \theta)(f(x) - y_0).$$

Therefore,

$$\begin{aligned} |\Phi(f)(x) - y_0| &= \left| f(x) - y_0 - \frac{F(x, y_0) + F_y(x, \theta)(f(x) - y_0)}{F_y(x_0, y_0)} \right| \\ &\leq \frac{|F_y(x, \theta) - F_y(x_0, y_0)|}{|F_y(x_0, y_0)|} |f(x) - y_0| + \frac{|F(x, y_0)|}{|F_y(x_0, y_0)|} \\ &\leq L\varepsilon + (1 - L)\varepsilon = \varepsilon. \end{aligned}$$

Thus we have $\Phi(f) \in \mathcal{F}$.

It is straightforward from the definition that Φ is computable. It remains to show that Φ is a contraction. By using the mean value theorem, we obtain that for any $f, g \in \mathcal{F}$ and $x \in I'$, there exists a real θ' between $f(x)$ and $g(x)$ such that

$$F(x, f(x)) - F(x, g(x)) = F_y(x, \theta')(f(x) - g(x)).$$

Using such a θ', we obtain

$$\begin{aligned} |\Phi(f)(x) - \Phi(g)(x)| &= \left| f(x) - g(x) - \frac{F(x, f(x)) - F(x, g(x))}{F_y(x_0, y_0)} \right| \\ &= \frac{|F_y(x, \theta') - F_y(x_0, y_0)|}{|F_y(x_0, y_0)|} |f(x) - g(x)| \\ &\leq L|f(x) - g(x)|. \end{aligned}$$

This yields $d(\Phi(f), \Phi(g)) \leq Ld(f, g)$.

Now we are ready to apply the effective contraction theorem. Application of the effective contraction theorem to $\mathcal{F}$ and Φ yields that Φ has a unique fixed point and the fixed point is computable. It concludes that there is a computable function $f : I' \to \mathbb{R}$ such that $F(x, f(x)) = 0$ for any $x \in I'$. □

Note that it is not computability of θ and θ' but existence of them that is required in the proof of Theorem 2. It is not necessary to effectivize the usages of the mean value theorem in the proof.

4.2 Computability of Self-Similar Sets

We denote by $\|\text{-}\|$ the Euclidean norm. For a nonempty subset S of $\mathbb{R}^q$, the function $\underline{d}_S : \mathbb{R}^q \to [0, +\infty)$ is defined by $\underline{d}_S(x) = \inf_{a \in S} \|x - a\|$.

Let $\varphi_1, \ldots, \varphi_m : \mathbb{R}^q \to \mathbb{R}^q$ be contractions. Then there exists a unique nonempty compact subset X of $\mathbb{R}^q$ such that

$$X = \varphi_1(X) \cup \cdots \cup \varphi_m(X). \tag{1}$$

Hutchinson [4] first proved this theorem by using the contraction theorem on the space of nonempty compact subsets of a Euclidean space with the Hausdorff metric. Hata [3] generalized the theorem. Kamo and Kawamura [5] added the viewpoint of computability; they have shown that if all of $\varphi_1, \ldots, \varphi_m$ are computable, then the unique solution X of the equation (1) satisfies that $\underline{d}_X$ is a computable function. We will rewrite Kamo and Kawamura's proof by using the effective contraction theorem.

Let $\mathcal{K}(\mathbb{R}^q)$ denote the set of all nonempty compact subsets of $\mathbb{R}^q$. A complete metric on $\mathcal{K}(\mathbb{R}^q)$ known as the *Hausdorff metric* d_{H} is defined by:

$$d_{\mathrm{H}}(K, L) = \max\{\sup_{a \in K} \inf_{b \in L} \|a - b\|, \sup_{b \in L} \inf_{a \in K} \|b - a\|\}.$$

Lemma 4. *For any $K, L \in \mathcal{K}(\mathbb{R}^q)$ and $x \in \mathbb{R}^q$, it holds that $|\underline{d}_K(x) - \underline{d}_L(x)| \leq d_{\mathrm{H}}(K, L)$.*

Proof. Some manipulation of sup's and inf's yields that

$$\begin{aligned} \underline{d}_K(x) - \underline{d}_L(x) &= \sup_{b \in L} \inf_{a \in K} (\|x - a\| - \|x - b\|) \\ &\leq \sup_{b \in L} \inf_{a \in K} \|b - a\| \end{aligned}$$

By exchanging K and L, we obtain also that

$$\underline{d}_L(x) - \underline{d}_K(x) \leq \sup_{a \in K} \inf_{b \in L} \|a - b\|.$$

Therefore, $|\underline{d}_K(x) - \underline{d}_L(x)| \leq d_{\mathrm{H}}(K, L)$. □

We denote by $\mathcal{S}_{\mathrm{H}}$ a set of sequences over $\mathcal{K}(\mathbb{R}^q)$ such that $(K_n) \in \mathcal{S}_{\mathrm{H}}$ iff $(\underline{d}_{K_n})$ is a computable sequence of functions. We define $\mathfrak{K}_k = \{K \in \mathcal{K}(\mathbb{R}^q) \mid K \subseteq \bar{B}(0, k+1)\}$. We call $K \in \mathcal{K}(\mathbb{R}^q)$ a rational finite set if K is a finite set of rational points. Let $(\Xi_{k,i})_{k,i}$ be an effective enumeration of all rational finite sets in $\mathcal{K}(\mathbb{R}^q)$ such that $(\Xi_{k,i})_i$ is an effective enumeration of all rational finite sets in $\mathfrak{K}_k$ for each k.

Lemma 5. *If (K_m) and (L_n) are effectively selected subsequences of $(\Xi_{k,i})$, then $(\sup_{a \in K_m} \underline{d}_{L_n}(a))_{m,n}$ forms a computable double sequence of reals.*

The proof is straightforward from the effectiveness of enumeration of $(\Xi_{k,i})$.

Proposition 1. *$(\mathcal{K}(\mathbb{R}^q), d_{\mathrm{H}}, \mathcal{S}_{\mathrm{H}})$ is an effectively locally compact metric space with $(\mathfrak{K}_k)$ and $(\Xi_{k,i})$.*

Proof. It is a well-known property of the Hausdorff metric that $(\mathcal{K}(\mathbb{R}^q), d_{\mathrm{H}})$ is a complete metric space.

Next, we verify that $\mathcal{S}_{\mathrm{H}}$ is a computability structure on $(\mathcal{K}(\mathbb{R}^q), d_{\mathrm{H}})$. Lemma 5 implies that if $(K_m), (L_n) \in \mathcal{S}$, then $(d_{\mathrm{H}}(K_m, L_n))_{m,n}$ forms a computable sequence of reals, i.e., the metric axiom holds. Checking the subsequence axiom is straightforward from the properties of computable real-valued functions on $\mathbb{R}^q$. Lemma 4 implies that if $(K_{n,k})$ converges to (K_n) effectively in n and k as $k \to \infty$, then $(\underline{d}_{K_{n,k}})$ uniformly converges to $(\underline{d}_{K_n})$ effectively in n and k as $k \to \infty$. Hence the limit axiom holds.

Finally, we verify that $(\mathcal{K}(\mathbb{R}^q), d_{\mathrm{H}}, \mathcal{S}_{\mathrm{H}})$ is an effectively locally compact metric space with $(\mathfrak{K}_k)$ and $(\Xi_{k,i})$. It follows immediately from the definition that $\overline{\{\Xi_{k,i} \mid i \in \mathbb{N}\}} = \mathfrak{K}_k$. It follows from the effectiveness of enumeration of $(\Xi_{k,i})$ and denseness of $(\Xi_{k,i})$ in $\mathfrak{K}_k$ that there exists a recursive function $\psi : \mathbb{N}^2 \to \mathbb{N}$ such that for any k, it holds that $\bigcup_{i<\psi(k,p)} B(\Xi_{k,i}, 2^{-p}) \supseteq \mathfrak{K}_k$. Since $K \subseteq \bar{B}(0, k+1)$ and $d_{\mathrm{H}}(K, K') \leq 1$ imply $K' \subseteq \bar{B}(0, k+2)$, we have for any $\boldsymbol{i} \in \mathbb{N}^{\mathbb{N}}$ and $k \in \mathbb{N}$, if $(\Xi_{k,\boldsymbol{i}[j]})$ converges exponentially in j as $j \to \infty$ and $\lim_{j\to\infty} \xi_{k,\boldsymbol{i}[j]} \in \mathfrak{K}_k$, then $B(\lim_{j\to\infty} \Xi_{k,\boldsymbol{i}[j]}, 1) \subseteq \mathfrak{K}_{k+1}$. □

The following lemma is used in the proof of Hutchinson and Hata's Theorem. Refer to [4] or [3] for detail.

Lemma 6. *For $\varphi_1, \ldots, \varphi_m : \mathbb{R}^q \to \mathbb{R}^q$, define $\Phi : \mathcal{K}(\mathbb{R}^q) \to \mathcal{K}(\mathbb{R}^q)$ by:*

$$\Phi(X) = \varphi_1(X) \cup \cdots \cup \varphi_m(X). \tag{2}$$

If all of $\varphi_1, \ldots, \varphi_m$ are contractions, then Φ is a contraction on $\mathcal{K}(\mathbb{R}^q)$.

In addition, we will use the following lemma.

Lemma 7. *For $\varphi_1, \ldots, \varphi_m : \mathbb{R}^q \to \mathbb{R}^q$, define $\Phi : \mathcal{K}(\mathbb{R}^q) \to \mathcal{K}(\mathbb{R}^q)$ by (2) in Lemma 6. If all of $\varphi_1, \ldots, \varphi_m$ are computable functions, then Φ maps $(\Xi_{k,i})$ to a computable sequence.*

Proof. Since $\Xi_{k,i}$ is a finite set, so is $\Phi(\Xi_{k,i})$. Thus

$$\underline{d}_{\Phi(\Xi_{k,i})}(x) = \min\Big(\bigcup_{a\in\Xi_{k,i}} \{\|\varphi_1(a) - x\|, \ldots, \|\varphi_m(a) - x\|\}\Big). \tag{3}$$

The right-hand side is the minimum of the values of finitely many computable functions and the minimization is uniform on k and i. Therefore $(\underline{d}_{\Phi(\Xi_{k,i})})$ is a computable sequence of functions, i.e., $(\Phi(\Xi_{k,i}))$ is computable. □

Now we prove the result on computability of self-similar sets by using the effective contraction theorem. Let $\varphi_1, \ldots, \varphi_m : \mathbb{R}^q \to \mathbb{R}^q$ be computable contractions. Define $\Phi : \mathcal{K}(\mathbb{R}^q) \to \mathcal{K}(\mathbb{R}^q)$ by (2) in Lemma 6. By using Lemmata 6 and 7, we obtain that $\operatorname{Fix}\Phi$ is a computable point on $\mathcal{K}(\mathbb{R}^q)$, i.e., the unique compact nonempty solution X of the equation (1) satisfies that $\underline{d}_X$ is a computable function.

Theorem 3 (Kamo and Kawamura). *Let $\varphi_1, \ldots, \varphi_m : \mathbb{R}^q \to \mathbb{R}^q$ be computable contractions. Then there exists a unique nonempty compact subset X of $\mathbb{R}^q$ such that*

$$X = \varphi_1(X) \cup \cdots \cup \varphi_m(X).$$

For such an X, it holds that $\underline{d}_X$ is a computable function.

Consider an extension of Hutchinson and Hata's Theorem to systems of fixed-point equations [1]. During this consideration, we implicitly declare the index j ranging over $1, \ldots, p$ and we omit the phrase "for any $j = 1, \ldots, p$".

Let $\varphi_{j11}, \ldots, \varphi_{j1m_{j1}}, \ \ldots\ , \varphi_{jp1}, \ldots, \varphi_{jpm_{jp}} : \mathbb{R}^q \to \mathbb{R}^q$ be contractions. Let each $K_j \subset \mathbb{R}^q$ be a compact subset. Then there exists a unique tuple $(X_1, \ldots, X_p)$ of nonempty compact subsets of $\mathbb{R}^q$ such that

$$X_j = K_j \cup (\varphi_{j11}(X_1) \cup \cdots \cup \varphi_{j1m_{j1}}(X_1)) \cup \cdots \cup (\varphi_{jp1}(X_p) \cup \cdots \cup \varphi_{jpm_{jp}}(X_p)).$$

We effectivize also this extended theorem.

Theorem 4. *Let $\varphi_{j11}, \ldots, \varphi_{j1m_{j1}}, \ \ldots\ , \varphi_{jp1}, \ldots, \varphi_{jpm_{jp}} : \mathbb{R}^q \to \mathbb{R}^q$ be computable contractions. Let $K_j \subset \mathbb{R}^q$ be a compact subset such that K_j is empty or $\underline{d}_{K_j}$ is a computable function. Then there exists a unique tuple $(X_1, \ldots, X_p)$ of nonempty compact subsets of $\mathbb{R}^q$ such that*

$$X_j = K_j \cup (\varphi_{j11}(X_1) \cup \cdots \cup \varphi_{j1m_{j1}}(X_1)) \cup \cdots \cup (\varphi_{jp1}(X_p) \cup \cdots \cup \varphi_{jpm_{jp}}(X_p)).$$

For such a tuple $(X_1, \ldots, X_p)$, each $\underline{d}_{X_j}$ is a computable function.

The proof is analogous to that for Theorem 3. Use $\mathcal{K}(\mathbb{R}^q)^p$ instead of $\mathcal{K}(\mathbb{R}^q)$ as an effectively locally compact space. Use the defining formula

$$\begin{aligned}
&\Phi(X_1, \ldots, X_p) = (\Phi_1(X_1, \ldots, X_p), \ldots, \Phi_p(X_1, \ldots, X_p)),\\
&\Phi_j(X_1, \ldots, X_p)\\
&\quad = K_j \cup (\varphi_{j11}(X_1) \cup \cdots \cup \varphi_{j1m_{j1}}(X_1)) \cup \cdots \cup (\varphi_{jp1}(X_p) \cup \cdots \cup \varphi_{jpm_{jp}}(X_p))
\end{aligned}$$

instead of (2) in Lemmata 6 and 7 to construct a computable contraction Φ on $\mathbb{R}^p$. The important intermediate result corresponding to (3) in the proof of Lemma 7 is that

$$\begin{aligned}
&\underline{d}_{\Phi_j(\Xi_{k_1,i_1},\ldots,\Xi_{k_p,i_p})}(x) \\
&\quad = \min\Big(S_j \cup \bigcup_{a\in\Xi_{k_1,i_1}} \{\|\varphi_{j11}(a)-x\|,\ldots,\|\varphi_{j1m_{j1}}(a)-x\|\} \\
&\qquad\qquad \cup\cdots\cup \bigcup_{a\in\Xi_{k_p,i_p}} \{\|\varphi_{jp1}(a)-x\|,\ldots,\|\varphi_{jpm_{jp}}(a)-x\|\}\Big) \\
&\qquad\qquad\qquad \text{where } S_j = \begin{cases} \emptyset & \text{if } K_j = \emptyset, \\ \{\underline{d}_{K_j}(x)\} & \text{otherwise.} \end{cases}
\end{aligned}$$

Note that Theorem 3 is a special case of Theorem 4 with $p = 1$ and $K_1 = \emptyset$.

4.3 Computability of the Takagi Function

The Takagi function $T : \mathbb{R} \to \mathbb{R}$ is defined by:

$$T(x) = \sum_{i=0}^{\infty} \frac{1}{2^i}\psi(2^i x)$$
$$\text{where } \psi(x) = \begin{cases} x-n & \text{if } n \le x < n+1/2 \text{ for some } n \in \mathbb{Z}, \\ n-x & \text{if } n-1/2 \le x < n \text{ for some } n \in \mathbb{Z}. \end{cases}$$

We will check the computability of the Takagi function by using the effective contraction theorem.

Since T is a periodic function with a period 1, it is sufficient to check the computability of the restriction of T on $[0,1]$. We will in general identify a periodic function with a period 1, a function on $[0,1]$ with $f(0) = f(1)$, and a function on a circle S^1.

The set $C(S^1) = \{f \in C[0,1] \mid f(0) = f(1)\}$ is a complete metric space with the metric $d(f,g) = \sup\{|f(x) - g(x)| \mid x \in [0,1]\}$. Define $\Phi : C(S^1) \to C(S^1)$ by:

$$\Phi(f)(x) = \begin{cases} \frac{1}{2}f(2x) + x & \text{if } 0 \le x \le 1/2, \\ \frac{1}{2}f(2x-1) - x + 1 & \text{if } 1/2 \le x \le 1. \end{cases}$$

Then it follows immediately from the definition that $d(\Phi(f),\Phi(g)) = (1/2)d(f,g)$ for any $f, g \in C(S^1)$, i.e., Φ is a contraction on X. We can easily verify that $T = \mathrm{Fix}\,\Phi$ with some calculation.

Let $\mathcal{S}$ be a set of sequences over $C(S^1)$ such that $(f_n) \in \mathcal{S}$ iff (f_n) is a computable sequence of functions. Let (K_k) be a sequence of subsets of $C(S^1)$ such that $K_k = \{f \in C(S^1) \mid \sup|f| \le k+1\}$. We call $f \in C(S^1)$ a rational polygon function if the graph of f is a polygon connecting rational points. Let

$(p_{k,i})_{k,i}$ be an effective enumeration of all rational polygon functions in $C(S^1)$ such that $(p_{k,i})_i$ is an effective enumeration of all rational polygon functions in K_k for each k. Then it is easy to verify that $(C(S^1), d, \mathcal{S})$ is an effectively locally compact set with (K_k) and $(p_{k,i})$.

It is easy to show that Φ maps $(p_{n,i})$ to an effectively selected subsequence of $(p_{n,i})$. Therefore Φ is a computable contraction. It follows from the effective contraction theorem that the Takagi function T is computable since $T = \operatorname{Fix}\Phi$.

5 Future Work

Effective σ-total boundedness takes a small part in the discussion on the effective contraction theorem; it appears only in the definition of computable functions. It suggests that effective σ-total boundedness is a too strong assumption. We are seeking mathematical objects more general than effective locally compact metric spaces and on which the effective contraction theorem still holds. If we will find ones, the effective contraction theorem shall be of wider application.

Acknowledgments

The author thanks to the participants of CCA 2000 for their helpful suggestions and comments.

References

1. C. Bandt: Self-Similar Sets 3. Constructions with Sofic systems, Monatshefte für Mathematik, **108** (1989) 89–102.
2. A. Grzegorczyk: On the definition of computable real functions, Fund. Math. **44** (1957) 61–71.
3. M. Hata: On the structure of self-similar sets, Japan J. appl. Math., **2** (1985) 381–414.
4. J. E. Hutchinson: Fractals and self-similarity, Indiana Univ. Math. J., **30** (1981) 713–747.
5. H. Kamo, K. Kawamura: Computability of self-similar sets, Math. Log. Quart. **45** (1999) 23–30.
6. T. Mori, Y. Tsujii, M. Yasugi: Computability structures on metric spaces, in: D. S. Bridges et al (Eds.), Combinatorics, Complexity and Logic, Proc. DMTCS '96, Springer, Berlin, (1996) 351–362.
7. M. B. Pour-El, J. I. Richards: *Computability in Analysis and Physics*, Springer-Verlag, Berlin, Heidelberg, 1989.
8. H. G. Rice: Recursive real numbers, Proc. Am. Math. Soc., **5** (1954) 784–791.
9. T. Takagi: A simple example of the continuous function without derivative, Proc. of the Physico-Mathematical Society of Japan, **II-1** (1903) 176–177. Re-recorded in: The Collected Papers of Teiji Takagi, Iwanami Shoten Publ., Tokyo, (1973) 5–6.
10. K. Weihrauch: *Computability*, Springer-Verlag, Berlin, Heidelberg, 1987.
11. M. Yasugi, T. Mori, Y. Tsujii: Effective properties of sets and functions in metric spaces with computability structure, Theoret. Comput. Sci. **219** (1999) 467–486.

Polynomially Time Computable Functions over p-Adic Fields

George Kapoulas

Athens University of Economics and Business
Dept. of Accounting and Finance
gkapou@math.ntua.gr

Abstract. Based on the notion of a computable p-adic number the notion of a polynomially time computable function over the field of p-adic numbers is introduced and studied. Theorems relating analytical properties with computability properties are established. The complexity of roots, and inverse function theorems are established at the level of polynomial time complexity. Relations between differentiability and polynomial time complexity and the maximization problem are discussed. Differences and similarities between the analogous questions for the real numbers are pointed out. [1]
Keywords: Computable, polynomially time computable numbers, computable, polynomially time computable functions, p-adic numbers.

1 Introduction

A. Turing in [23] defined the notion of a computable real number based on the binary expansion of a real number. This definition proved to be inadequate. One reason is that we cannot effectively determine correctly the first digit of the representation of the sum of two computable real numbers from the representation of the two summands (see D. Bridges [2], Y. Moschovakis [17], K. Skandalis [22].)

The definition used eventually in the literature was to effectivize the construction of the real numbers from the set of rationals. Either the notion of Dedekind cut or the notion of Cauchy completion was used subject to some effectivity constraints regarding the constructions. The set of rationals is used, since technically it can be enumerated by a one to one total recursive function and philosophically it is a set of simple objects.

The notion of a computable function is an important relevant notion. Intuitively a computable function is an effective transformation of some input data to some output data and this transformation is represented by a recursive function (or any other formulation of the notion of algorithm). There are two approaches in the literature. The first is to study computable real functions which have as domain computable real numbers only and the other is to admit all possible real numbers. In both approaches the fundamental idea is that a real

[1] The author expresses his acknowledgements to the helpful comments of the two anonymous referees.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 101–118, 2001.

number or a function is an infinite object and the approach to understand it is to use effective approximations (representations, descriptions) to the desired object. However not all numbers admit such an effective representation (description) and we obtain a subset of the classical objects. Regarding the functions, a function is an effective transformation of approximations to the argument x to approximations to the value $f(x)$. This effective transformation is expressed (represented) by a Turing machine.

In the approach where all real numbers are allowed as arguments to the function, the Turing machine calculating the function has one input a natural number which is a measure of the desired accuracy of the result $f(x)$ (the analogue of ϵ of Analysis) and another input in the form of an oracle. The second input (oracle) represents the real number x which is the argument x of the function f.

In the approach where only computable real numbers are allowed there is one only input, a natural number which is the code of the real number which is the argument x of the function f.

In the approach where computable real numbers only are allowed most of the theorems of Classical Analysis fail (see B. Kushner [11] and O. Aberth [1] for details).

The approach where all possible real numbers are admitted as inputs is the approach followed by H. Friedman and Ker – I Ko [6], [8] among others, where the set of questions studied there are studied for the case of the real numbers and their functions (functions regarded as effective transformations). This latter approach is closer to the practice of Analysis.

The present approach falls within the scope of the traditional approach used in Constructive and Recursive Analysis and should be contrasted to the one used in L. Blum, M. Shub and S. Smale [12], where a real number is understood as an object that can be comprehended and manipulated in its totality at once. Such an approach does not admit so far a natural representation of higher order objects in Analysis like the integral, the maximum value problems and the derivatives to mention some problems addressed by the present model. This seems to fail to express the notion of Analysis that the traditional approach of Recursive and Constructive Analysis are able to express.

Using the approach of the topological completion of the rational numbers it is natural to examine all possible completions of the field of rational numbers via Cauchy sequences. These completions are the real numbers and for each prime p the field of p-adic numbers. These are the only possibilities by a theorem of Ostrowski [18]. Basic properties of the computable p-adic numbers are studied in G. Kapoulas [7]. For a detailed introduction to the p-adic fields the interested reader is referred to J. W. Cassels [3], F. Q. Gouvea [5], N. Koblitz [9], K. Mahler [16] and W. H. Shikhoff [21]. The first two have an Algebraic approach, the last two an Analytic one and N. Koblitz [9] has a brief introduction to the subject.

The p-adic fields are equipped with a metric, where the metric is defined over the rational numbers and extended by continuity to the p-adic fields.

The study of computability and complexity questions over p-adic fields falls broadly within the scope of Recursive, Computable or Constructive Analysis depending on the taste of the author and the philosophical background and offers a variety of open questions. Most of the proofs in the case of the p-adic numbers are technically easier than the proofs for the case of real numbers due to the topology of the p-adic fields. The p-adic fields are totally disconnected as topological spaces and the topology can be defined by Algebraic means since the set of p-adic integers (which is the analogue of the unit interval of the real line) are natural examples of local rings.

The theorems established in the present are:

1. A polynomially time computable computable function is continuous and has modulus of continuity bounded by a polynomial. As a partial converse a continuous function which has modulus of continuity bounded by a polynomial is computable in polynomial time relative to an oracle (Theorem 1, Theorem 2).
2. There exist polynomially time computable functions that have roots of arbitrarily high complexity (Theorem 4). A condition that guarantees that roots of polynomially time computable functions are polynomially time computable is given (Theorem 5).
3. An inverse function theorem (Theorem 5), and an implicit function theorem are proved for polynomially time computable functions (Proposition 2).
4. There exists a polynomially time computable function which is nowhere differentiable (Theorem 6).
5. The maximum value problem for computable functions is studied and it is proved that for non identically zero functions this problem is computable in constant time (Theorem 7).

Notation and Conventions. The end of a proof will be denoted by $\square$.

The set of natural, integer and rational numbers will be denoted by $\mathbf{N}$, $\mathbf{Z}$ and $\mathbf{Q}$ respectively. An arbitrary prime number will be denoted by p.

A standard encoding (indexing) of the recursive functions will be assumed. The recursive function with index e will be denoted by ϕ_e.

The set of p-adic numbers, p-adic integers, will be denoted by $\mathbf{Q}_p$, $\mathbf{Z}_p$ respectively.

For a set A the notation $A^{<\omega}$ will denote the set of finite sequences of elements of A.

The open sphere with center x and radius p^{-m} will be denoted by $S\left(x, p^{-m}\right)$.

The standard (unique) base p representation of a p-adic number will be used i.e.

$$x \in \mathbf{Q}_p \setminus \{0\} \leftrightarrow x = p^k \sum_{n=0}^{\infty} a_n p^n$$

with $a_i \in \{0, 1, \ldots, p-1\},\ k \in \mathbf{Z},\ a_0 \neq 0$. It is possible to use the Teichmüller representation but it is expected that these two representations are equivalent since we can translate from the one to the other in linear time.

The model used for the study of functions is the oracle Turing machine model. Intuitively if a function f is computable by an oracle Turing machine $\mathcal{M}$ the oracle can furnish some finite approximation to the value of x written on the oracle tape, and the output of $\mathcal{M}$ is some finite approximation to the value of $f(x)$. The values for x and $f(x)$ are given not as completely known objects, but by finite approximations to the values x and $f(x)$ and the machine $\mathcal{M}$ operates on these approximations of the argument x and produces as output approximations to $f(x)$. The input to the machine $\mathcal{M}$ is a natural number n in unary and represents the desired accuracy of the output. To carry out the calculations the Turing machine $\mathcal{M}$ asks (possibly several times) for approximations to the value of x. This information is given by the oracle and the finite approximation is the content of the oracle tape.

The details for the model for representing computable functions over the p-adic fields are described below: A function $f\colon \mathbf{Z}_p \to \mathbf{Q}_p$ is computable if there is a (function) oracle Turing machine $\mathcal{M}$ such that $\mathcal{M}$ having n written on the input tape, with x as the oracle will (roughly) calculate the first n digits of $f(x)$. The oracle when asked will write some finite approximation to the value x on the oracle tape. The representation of a p-adic number that will be used is the base p representation of a p-adic number. Because of this convention the Turing machines will have as alphabet the set $\{0, 1, \ldots, p-1\}$. At any time, the content of the oracle tape will be an element of the set $\{0, 1, \ldots, p-1\}^{<\omega}$, and the content of the output tape will be an element of the set $\{0, 1, \ldots, p-1\}^{<\omega}$. More specifically the tapes of the Turing machine are an input tape, an output tape, an oracle tape, a query tape and a number of tapes as workspace. The oracle tape contains information about the argument x to the function f in the form of finite approximations to x.

The Turing machine which calculates (represents) a function f in addition to the standard states has a query state. The input tape contains a number n in unary which is the desired accuracy of $f(x)$. During the calculations the Turing machine may write a number m (in unary which is represented by 0^m) on the query tape. When the machine enters the query state the oracle will examine the content of the query tape and will write $r_m = \phi(0^m)$ on the oracle tape which is a rational number and is an approximation s.t. $|r_m - x| < p^{-m}$ (roughly the first m digits of x.) The next state is determined by the Turing machine before it enters the query state. The above model is called a function oracle Turing machine since the oracle reads a string and gives a new string as an answer. The time complexity of this model of computation is defined to be the number of moves that the Turing machine requires for the calculations, where the process of writing 0^n querying and writing $\phi(0^n)$ requires $2n + 1$ steps.

The input to the Turing machine representing a function f is in unary. This contrasts with the standards of complexity theory but is necessary for the study of computable numbers and functions. For example if the input number n is given in the binary representation of n then the input size will be roughly $\log(n)$. This in turn entails that functions as $f(x) = x$ will have exponential time complexity. As an alternative to the convention of using the unary representation of a

number n it is possible to use the size of the number n and not the size of the representation of n.

2 General Properties of Polynomially Time Computable Functions

Definition 1. *Let f be a function $f : \mathbf{Z}_p \to \mathbf{Q}_p$. The function f is* computable *if and only if there is an oracle Turing machine $\mathcal{M}$ over the alphabet $\{0, 1, 2, \ldots, p-1\}$ such that having n as the content of the input tape (representing the desired output accuracy), and x (representing the argument to the function) as the oracle, $\mathcal{M}$ outputs a rational number r_n such that $|r_n - f(x)| < p^{-n}$. In this case the Turing machine $\mathcal{M}$* represents *f.*

Definition 2. *A computable function $f : \mathbf{Z}_p \to \mathbf{Q}_p$* has time complexity *$g(n)$ if and only if there exists an oracle Turing machine $\mathcal{M}$ that represents f, such that having n in unary as the content of the input tape and any x as oracle, the running time of $\mathcal{M}$ is $g(n)$ uniformly in x.*

Definition 3. *A computable function $f : \mathbf{Z}_p \to \mathbf{Q}_p$* is polynomially time computable (or computable in polynomial time) *if and only if there exists a Turing machine $\mathcal{M}$ that represents f and the running time of $\mathcal{M}$ having n in unary as content of the input tape and any x as oracle, is bounded by a polynomial in n uniformly in x.*

A basic theorem of recursive analysis over the real numbers is that a computable function is continuous and that a continuous function is computable relative to some oracle set. A similar statement is also true at the polynomially time computable level for the real numbers [6] and for the polynomially time computable p-adic numbers. For the proof of this theorem which is a computation relative to a set, the notion of a set oracle Turing machine is used. The model is a variant of the function oracle model. In addition to the states of an ordinary Turing machine the set oracle Turing machine has a yes state, a no state and a set query state. During the computations if the Turing machine enters the set query state the set oracle will examine the content of the oracle tape and then the Turing machine will continue the calculations having as the next state the yes state or the no state, depending on the membership of the content of the oracle tape in the set oracle.

Theorem 1. *If $f\colon \mathbf{Z}_p \to \mathbf{Q}_p$ is a computable function then f is continuous.*

Proof. Let f be computed by a Turing machine $\mathcal{M}$ as above. Then $\mathcal{M}$ on input n, and with any oracle y will halt after a finite number of steps $m(n, y)$. Then we have that on any oracle y the machine will examine at most the first $m(n, y)$ cells of the oracle tape, i.e. the Turing machine will examine only the first $m(n, y)$ digits of y. Hence we have:

$$|x - y| < p^{-m(n,y)} \Rightarrow |\mathcal{M}^y(n) - \mathcal{M}^x(n)| < p^{-n}.$$

From this we have:

$$|f(x) - f(y)| = |f(x) - \mathcal{M}^x(n) + \mathcal{M}^x(n) - \mathcal{M}^y(n) + \mathcal{M}^y(n) - f(y)| \leq$$

$$\leq \max\Big\{|f(x) - \mathcal{M}^x(n)|, |\mathcal{M}^x(n) - \mathcal{M}^y(n)|, |\mathcal{M}^y(n) - f(y)|\Big\} \leq$$

$$\leq \max\Big\{p^{-n}, p^{-n}, p^{-n}\Big\} = p^{-n}.$$

Hence we have $|x - y| < p^{-m(n,y)} \Rightarrow |f(x) - f(y)| < p^{-n}$ which shows that f is continuous. □

Theorem 2. *If $f\colon \mathbf{Z}_p \to \mathbf{Q}_p$ is polynomially time computable, then f is continuous and has modulus of continuity bounded by a polynomial. As a partial converse, if $f\colon \mathbf{Z}_p \to \mathbf{Q}_p$ has modulus of continuity bounded by a polynomial, then there exists an oracle set $\mathcal{O}$ such that f is polynomially time computable with respect to $\mathcal{O}$.*

Proof. Let f be polynomially time computable by a Turing machine $\mathcal{M}$, and let $m(n)$ be the time bound of the Turing machine with x as the content of the oracle tape and n as the content of the input tape. Since the computation halts after a finite number of steps, we have that the amount of information used from the oracle set is finite, and bounded by the number of steps that the Turing machine requires to halt. This in turn implies that on input n for every p-adic number z, which admits a representation which agrees with the content of the oracle tape for x up to the $m(n)$th position, the computation will be the same, and hence the output will be the same as well. Therefore we have the following:

$$|x - z| \leq p^{-m(n)} \Rightarrow |f(x) - f(z)| \leq p^{-n},$$

which shows that the modulus of continuity is bounded by a polynomial.

For the converse we have that f is uniformly continuous on $\mathbf{Z}_p$ since $\mathbf{Z}_p$ is compact. Suppose that the modulus of continuity of f is m where m is bounded by a polynomial. For $x \in \mathbf{Z}_p$ and $n \in \mathbf{N}$, we have to calculate an approximation of $f(x)$ within p^{-n} (roughly the first n digits of $f(x)$). To achieve this consider the set:

$$\mathcal{O} = \{\langle n, c, d\rangle \mid n \in \mathbf{N}, c \in \{0, \ldots, p-1\}^{m(n)}, d \in \{0, \ldots, p-1\}^n \quad \text{and},$$

$$|d - f(c)| \leq p^{-n}\}.$$

Then we have that f is computable relative to $\mathcal{O}$ as follows:
First calculate $m(n)$, and obtain $c \in \{0, \ldots, p-1\}^{m(n)}$ such that $|c - x| \leq p^{-m(n)}$.
The value of $f(x)$ can be calculated as follows:
Stage 0: let $d_0^k = k$, for $0 \leq k \leq p-1$.
Using the oracle $\mathcal{O}$, exactly one of the triples $\langle 1, c, k\rangle$ belongs to $\mathcal{O}$, and this determines a (unique) value for k. Let $d_0 = k$.
Stage $s+1$: Assuming that the correct value d_s has been determined (by induction), let $d_{s+1}^k = d_s + kp^{s+1}$ with $0 \leq k \leq p-1$. Again using the oracle exactly

one of the triples $\langle s+1, c, d_{s+1}^k \rangle$ will be an element of the oracle set $\mathcal{O}$, and this one determines $f(c)$ correctly up to $s+1$ digits. Let $d_{s+1} = d_s + k_0 p^{s+1}$, where k_0 is the value of k such that $\langle s+1, c, d_s + k_0 p^{s+1} \rangle \in \mathcal{O}$. Stop at stage n, and let $d = d_n$.

We have $|d - f(x)| = \max\{|d - f(c)|, |f(c) - f(x)|\} \leq p^{-n}$. For the time bounds now we have that $O(m(n))$ steps are required to determine the approximation c. Then at each stage s we had to query the oracle at most $p-1$ times to determine the correct answer, so the query phase requires $(p-1)n$ steps, which gives a time bound of $O(k(n))$ with $k(n) = \max\{m(n), (p-1)n\}$, hence it is bounded by a polynomial. □

3 Roots of Polynomially Time Computable Functions over p-Adic Fields

The existence of roots of equations, their properties and the effective determination of the roots is a fundamental problem in Mathematics. In the following we study the complexity of roots of polynomially time computable functions.

We observe that by taking the identically zero function there exist functions computable in polynomial time that have as roots p-adic numbers of arbitrarily high complexity, even non computable numbers as we have in the case of the real numbers.

Regarding the complexity of roots of polynomially time computable functions over the real numbers a technique due to H. Friedman and Ker – I Ko in [6] is to construct a polynomially time computable function having a root of arbitrarily high complexity as the limit of a suitable sequence of piecewise linear functions.

In the case of functions over the p-adic numbers we have a similar situation and the idea for the proofs is roughly the same. The members of the sequence of approximating functions are locally constant functions, which are continuous functions in the case of p-adic numbers. A function computable in polynomial time over the p-adic numbers then can be defined as the uniform limit of a sequence of locally constant functions. The modulus of convergence of the sequence is also bounded by a polynomial.

In the following definitions we assume an one to one and onto recursive enumeration of the set of rational numbers. This allows the introduction of the notions of effective (recursive) sequences, of effective (recursive) convergence and effective convergence rate of sequences of rational numbers.

Definition 4. *A sequence a_n of rational numbers* effectively (recursively) converges *to $x \in \mathbf{Q}_p$ if and only if there is a recursive function f s.t. $\forall n \in \mathbf{N},\ m \geq f(n) \Rightarrow |a_m - x|_p < 2^{-n}$.*

Definition 5. *A sequence of rational numbers $\{a_n \mid n \in \mathbf{N}\}$ is* recursive (effective) *if and only if there is a recursive function f s.t. $\forall n \in \mathbf{N}\ f(n) = a_n$. An index e s.t. $\phi_e(n) = f(n)$ is called an* index *of the sequence $\{a_n \mid n \in \mathbf{N}\}$.*

Definition 6. *A recursive sequence of rational numbers* $\{a_n \mid n \in \mathbf{N}\}$ *is* recursively or effectively Cauchy (in the p-adic metric) *if and only if there is a recursive function g s.t.* $\forall n, m > g(l)$ *we have* $\mid a_n - a_m \mid < p^{-l}$. *The function g is called* a modulus of the Cauchy criterion (or modulus of convergence) *of the sequence* $\{a_n \mid n \in \mathbf{N}\}$. *An index of the function g is called an* index of the Cauchy criterion.

Definition 7. *A* recursive p-adic number x *is the limit of recursive Cauchy sequence of rational numbers* $\{a_n \mid n \in \mathbf{N}\}$ *which converges recursively (in the p-adic metric) to x.*

The set of recursive p-adic numbers, and recursive p-adic integers, will be denoted by $\mathbf{Q}_p^c$, $\mathbf{Z}_p^c$ respectively.

Definition 8. *A recursive p-adic number x is* polynomially time computable *if and only if it is the limit of recursive Cauchy sequence of rational numbers* $\{a_n \mid n \in \mathbf{N}\}$ *which converges recursively (in the p-adic metric) to x, such that the (recursive) functions representing the modulus of convergence and the sequence* $\{a_n \mid n \in \mathbf{N}\}$ *are polynomially time computable w.r.t. the length of n in unary.*

Definition 9. *A function* $f\colon \mathbf{Z}_p \to \mathbf{Q}_p$ *is a* locally constant polynomially time computable function *if there exists a finite number of clopen balls* $\{A_i \mid i \leq n\}$, *and a fixed finite set of polynomially time computable p-adic numbers* $\{x_i \mid i \leq n\}$ *such that:*

1. *The sets* $A_i,\ i \leq n$ *are mutually disjoint,*
2. *The sets* $A_i,\ i \leq n$ *cover* $\mathbf{Z}_p$,
3. *For all* $i \leq n$ $f(A_i) = \{x_i\}$.

Definition 10. *Let* $\{f_n \mid n \in \mathbf{N}\}$ *be a sequence of locally constant computable functions. Then* $\{f_n \mid n \in \mathbf{N}\}$ uniformly converges to f with modulus of convergence bounded by a polynomial *(denoted by* $f_n \overset{p,u}{\to} f$ *) if and only if there exist two functions ϕ, ψ computable in polynomial time where the input n is given in unary, such that:*

$$\phi\colon \mathbf{N} \to \mathbf{N}, \quad \textit{and} \quad \psi\colon \mathbf{N} \times \{0, 1, 2, \cdots, p-1\}^{<\omega} \to \{0, 1, 2, \cdots, p-1\}^{<\omega},$$

such that:

1. *ϕ is bounded by a polynomial function,*
2. *length(d)* $\leq n \Rightarrow \psi(n, d) = f_n(d)$,
3. $\forall t \in \mathbf{Z}_p,\ \forall d \in \{0, 1, 2, \cdots, p\}^{<\omega},\ \forall n \in \mathbf{N}$ *with length(d)* $\leq \phi(n)$ *and* $|t - d| \leq p^{-\phi(n)} \Rightarrow |f(t) - \psi(\phi(n), d)| \leq p^{-n}$.

Theorem 3. *Let $f\colon \mathbf{Z}_p \to \mathbf{Q}_p$, f is computable in polynomial time if and only if there exists a sequence $\{f_n \mid n \in \mathbf{N}\}$ of locally constant functions such that $f_n \overset{p,u}{\to} f$. Each member of the set of (finite) values of p-adic numbers that each f_n takes on, is computable in polynomial time.*

Proof. ($\Leftarrow$) The conditions 1, 2 and 3 from definition 10 give an algorithm which runs in time bounded by a polynomial.

($\Rightarrow$) Suppose that f is computable in polynomial time, $\mathcal{M}$ is the function oracle Turing machine computing f, $n \in \mathbf{N}$, and P_f is the (polynomial) bound on the computation time of $\mathcal{M}$. By Theorem 2 f has modulus of continuity bounded by a polynomial. We can also assume without loss of generality that the modulus of continuity satisfies $m(n) \geq n$, so that we have:

$$\text{for all} \quad x, y \in \mathbf{Z}_p \quad |x - y| \leq p^{-m(n)} \Rightarrow |f(x) - f(y)| \leq p^{-n}.$$

Let f_n be defined as follows: The open balls for the partition of $\mathbf{Z}_p$ are the balls having "centers" the points $\sum_{i=0}^{m(n)} a_i p^i$ with $a_i \in \{0, 1, \ldots, p-1\}$, and radius $p^{-m(n)}$. These balls are mutually disjoint, and they cover $\mathbf{Z}_p$. The value of the function f_n on each of these balls is $\mathcal{M}^x(n)$ where $x = \sum_{i=0}^{m(n)} a_i p^i$ is the "center" of this ball. For $w \in \mathbf{Z}_p$, there exists a point x among the above mentioned "centers" such that $|x - w| \leq p^{-m(n)}$ so that $\mathcal{M}^w(n) = \mathcal{M}^x(n)$. Let $\psi(n, w) = \mathcal{M}^w(n)$, so that the second condition of the definition is satisfied. For the third and first condition now let $\phi(n) = m(n)$. For $z \in \mathbf{Z}_p$, and x among the centers of the above clopen sets such that

$$|z - x| \leq p^{-\phi(n)} \Rightarrow$$

$$\Rightarrow |f(z) - \psi(\phi(n), x)| \leq \max\{|f(z) - f(x)|, |f(x) - \psi(\phi(n), x)|\} =$$
$$= \max\{p^{-n}, p^{-\phi(n)}\} \leq p^{-n}.$$

We have that ϕ is a bounded by polynomial function, for d with $\text{length}(d) \leq n$ $\psi(n, d) = f_n(d)$, that $\psi(n, d)$ can be computed in $P_f(n)$ steps, and that each of the possible values of each f_n is polynomially time computable since it is a rational number. □

An important question when studying various number systems is, which equations are solvable and what are the properties of the solutions. In the case of p-adic functions (as in the case of functions over the real numbers [6]) we have that a polynomially time computable function can have as root a p-adic number of arbitrarily high complexity in $\mathbf{Q}_p^c$.

Definition 11. *A recursive p-adic number x has time complexity bounded from above by a function $f\colon \mathbf{N} \to \mathbf{N}$ if there exists a Turing machine $\mathcal{M}$ with index e, such that for each n, $\mathcal{M}(n)$ is a code for a rational number, the sequence $\{\mathcal{M}(n) \mid n \in \mathbf{N}\}$ effectively converges to x, the number of steps required for the calculation of $\mathcal{M}(n)$ is bounded from above by $f(n)$, and the modulus of convergence of the sequence $\{\mathcal{M}(n) \mid n \in \mathbf{N}\}$ is bounded from above by $f(n)$.*

Definition 12. *A recursive p-adic number x has time complexity bounded below by a function $f\colon \mathbf{N} \to \mathbf{N}$ if for all Turing machines with index e, such that for each n, $\mathcal{M}(n)$ is a code for a rational number, and the sequence $\{\mathcal{M}(n) \mid n \in \mathbf{N}\}$ effectively converges to x, with modulus of convergence n, and the number of steps required for the calculation of $\mathcal{M}(n)$ is bounded from below by $f(n)$.*

Lemma 1. *For any constant c there exists a computable p-adic number x with complexity $\Theta\left(2^{cn}\right)$.*

Proof. Consider the characteristic function χ_A of a set A which has complexity exactly 2^{cn} and define $x = \sum_{n=0}^{\infty} \chi_A(n)\, p^n$. □

Theorem 4. *There exists a function computable in polynomial time $f\colon \mathbf{Z}_p \to \mathbf{Q}_p$, $f \neq 0$ on any sphere $S \subseteq \mathbf{Z}_p$, such that f is a uniform limit of locally constant functions, $f' \equiv 0$, and f has a root which is not polynomially time computable.*

Proof. Let x be a number in $\mathbf{Z}_p$ such that x has time complexity $g(n)$, with $g \in \Theta(2^{cn})$ for some constant c. Let $\{\phi(i) \mid i \in \mathbf{N}\}$ be a computable sequence of rational numbers such that $|\phi(n) - x| \leq p^{-n}$.

Define a sequence of functions $\{f_n \mid n \in N\}$ as follows:

1. For $p^k < n < p^{k+1}$ let $f_n = f_{p^k}$,
2. For $n = p^k$ define f_n to be the following function:

$$f_n(y) = \begin{cases} 0, & \text{if } |x - y| < p^{-k}, \\ p^{p^i} & \text{if } |x - y| = p^{-i},\ i \leq k. \end{cases}$$

i.e. $f_{p^k}(y) = p^{p^i}$ if and only if $y = \phi(i)$ for some $i \leq k$, otherwise the value of the function is 0. We have that for any value of k, f_{p^k} and $f_{p^{k+1}}$ will differ only at the "annulus" with center x and radii p^k, p^{k+1} and so we will have:

$$\forall y \in \mathbf{Z}_p \quad \sup |f_{p^k}(y) - f_{p^{k+1}}(y)| = |p^{p^{k+1}}| = p^{-p^{k+1}},$$

so f_n converges uniformly with a linear modulus to a function f. We have that $f_n(d) = p^{p^i}$ for $d \in \mathbf{Z}_p$ such that $\text{length}(d) \leq n$ and $|x - d| = p^i$, hence $f(d) = p^{p^i}$ for such d.

We have $f(x) = 0$ and $f'(y) = 0$ for $y \neq x$. For x we have:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = \lim_{h \to 0} \frac{f(x+h)}{h} = 0.$$

To show that f is polynomially time computable it suffices to show $f_n \overset{p,u}{\to} f$. For this let $\phi(n) = n$, $\psi(n, d) = f_n(d)$. From these definitions we have that the first two clauses of the definition 10 are satisfied. For $t \in \mathbf{Z}_p$ such that $|t - d| \leq p^{-n} \Rightarrow |f(d) - \psi(n, d)| = 0$ so that the third clause of the definition 10 is satisfied. The values of the function $f_{p^k}(y)$ can be calculated by generating the first k digits of x and y and comparing, this requires time bounded above by $\Theta\left(2^{c(k+1)}\right)$ and then we have to output the appropriate string which has length bounded by p^k. □

The idea in the above proof can be generalized to the case where x can have arbitrarily high complexity in $\mathbf{Q}_p^c$.

Corollary 1. *For any computable p-adic number $x \in \mathbf{Z}_p$ there exists a polynomially time computable function $f: \mathbf{Z}_p \to \mathbf{Q}_p$ such that $f(x) = 0$, $f \neq 0$, on any sphere $S \subseteq \mathbf{Z}_p$, f is differentiable and $f' = 0$.*

Proof. There exists a recursive function B such that for all n we can calculate a p-adic integer z such that $|z-x| \leq p^{-n}$ in at most $B(n)$ steps, i.e. we can calculate the set $\{\phi(k) \mid k \in \mathbf{N}\}$ referred in the proof of Theorem 4 in time $\sum_{i=0}^{n} B(i)$. Since B represents time complexity we can assume w.l.o.g. that $\forall n \in \mathbf{N}\ B(n)$ is computable in time $O(B(n))$. Define:

$$\overline{B}(l) = \sum_{i=0}^{l} B(i),$$

and let $g_{\overline{B}(l)}$ be defined as follows:

$$g_{\overline{B}(l)}(y) = \begin{cases} 0, & \text{if } |x-y| < p^{-l}, \\ p^{\overline{B}(i)} & \text{if } |x-y| = p^{-i},\ i \leq l. \end{cases}$$

and $g_n(y) = g_{\overline{B}(k)}(y)$ for $\overline{B}(k) \leq n < \overline{B}(k+1)$. These two inequalities are decidable in polynomial time with respect to $\max\{k, n\}$ (from the definition of g_n). As before we have that there exists a function g such that $g_n \overset{p,u}{\to} g$ and $|g_{\overline{B}(l)} - g| \leq p^{-\overline{B}(l)}$ so g is polynomially computable, $g(x) = 0$, and $g'(x) = 0$. □

For a certain class of functions we will show that the solutions to the equation $f(x) = 0$ are polynomially time computable as in the case with the real number system [6]. However the condition for the p-adic case is weaker than the condition for the real numbers since it involves the behavior of the limit of the first quotient of differences while in the real case the corresponding condition is analyticity [6]. The condition for the p-adic case is a local version of the Mean Value Theorem of Real Analysis which gives all the necessary information for the calculations. We note that the p-adic fields are not ordered fields and the Mean Value Theorem fails in general. The local version of the Mean Value Theorem is strong enough to imply a local isometry. Having a local isometry makes it possible to use an argument similar to the binary search argument adapted to the p-adic case. The time estimates in the present case are linear with respect to the input length and logarithmic with respect to the search space.

In the case of polynomially time computable real numbers, it is proved that a root of an analytic polynomially time computable function is polynomially time computable. This statement is essentially about inverting a function at a single point [6].

For the case of polynomially time computable function over the p-adic numbers it is possible to invert a C^1 polynomially time computable function over a neighborhood of any point x when $f'(x) \neq 0$. The definition of C^1 p-adic functions is as follows:

Definition 13. *A function* $f\colon \mathbf{Q}_p \to \mathbf{Q}_p$ *is* C^1 *at* a *if and only if*

$$\lim_{(x,y)\to(a,a)} \frac{f(x)-f(y)}{x-y}$$

exists at a.

Theorem 5. *Let* $f\colon \mathbf{Z}_p \to \mathbf{Q}_p$ *polynomially time computable, and* $f \in C^1$ *at* $a \in \mathbf{Z}_p$. *Suppose that* $f'(a) \neq 0$, *then a neighborhood* $\mathcal{N}$ *of* a *exists such that* f^{-1} *is polynomially time computable in* $\mathcal{N}$.

Proof. Assume first that $|f'(a)| = 1$ and that $f(a) = 0$. In particular, the value of $|f'(a)|$ is a rational number. Let m be the modulus of convergence of the quotient $\frac{f(x)-f(y)}{x-y}$ to $f'(a)$. Then there exists $k \in \mathbf{N}$ such that for $x, y \in S\left(a, p^{-m(k)}\right) = \mathcal{N}$ we have:

$$\left|\frac{f(x)-f(y)}{x-y} - f'(a)\right| < |f'(a)|.$$

Since we have that for any triangle at least two of the sides of the triangle have equal length this inequality reduces to:

$$\left|\frac{f(x)-f(y)}{x-y}\right| = |f'(a)| \Rightarrow$$

$$\Rightarrow |f(x)-f(y)| = |f'(a)||x-y| \Rightarrow$$

$$\Rightarrow |f(x)-f(y)| = |x-y|. \tag{1}$$

Let f be computed by a Turing machine $\mathcal{M}$ which runs in time bounded by a polynomial q. Given $n \geq m(k)$, the following procedure will determine correctly the first n digits of a: Let $z \in \mathbf{Z}_p$ such that $|z-a| = p^{-m(k)}$ and z has minimum length (omitting trailing zeroes), let $c_0 = z$.

Stage 1: Let $c_1^i = z + ip^{m(k)+1}$, $0 \leq i \leq p-1$. Compute $\mathcal{M}^{c_1^i}(m(k)+1)$, for each i and choose the c_1^i that gives 0 as the first digit. Exactly one of the c_1^i will satisfy the above condition since $|f(x)-f(y)| = |x-y|$ so the first digit is computed correctly.

Suppose now that c_l is the initial approximation to the $f^{-1}(x)$ i.e. l stages of the above algorithm have been carried out.

Stage $l+1$: Let $c_{l+1}^i = c_l + ip^{l+1}$, $0 \leq i \leq p-1$. For all i compute $\mathcal{M}^{c_{l+1}^i}(m(k)+l+1)$ and choose the one that has 0 as output (by the condition $|f(x)-f(y)| = |x-y|$ there is going to be exactly one number satisfying this condition). Stop at stage $n - m(k)$.

Since we have an isometry there exists only one root in this neighborhood and the approximation to the root will be determined correctly by the above procedure.

The running time of the algorithm is calculated as follows: At stage l, $p-1$ computations are required (if the result of the first $p-1$ is not the correct one

then the last choice is the correct one). Each one of these will take $q(l)$ steps by $\mathcal{M}$, hence in total will have an upper bound for the calculations given by: $(p-1)[q(1)+q(2)+q(3)+\ldots+q(n)]$. Without loss of generality we may assume that $q(x)$ is non decreasing so $(p-1)(\sum_{i=1}^{n} q(i)) \leq n(p-1)q(n)$ is an upper bound for the time required.

In case that $|f'(a)| = p^m \neq 1$ and $|f(a)| = 0$ where $m \in \mathbf{Z}$, equation 1 becomes in this case $|f(x)-f(y)| = |f'(a)(x-y)| = p^m|x-y|$. In this case at the previous proof at stage l we will have to carry the i computations $\mathcal{M}^{c_l^i}(n+m)$ and the running time will be bounded above by $n(p-1)q(n+m)$ with m fixed and $m+n \geq 0$.

For the case that $f(a) = d \neq 0$, given $n \in \mathbf{N}$ there exists a k such that for all $x, y \in S\left(a, p^{-m(k)}\right) = \mathcal{N}$ we have:

$$\left|\frac{f(x)-f(y)}{x-y} - f'(a)\right| < |f'(a)|.$$

which as in the previous case reduces to :

$$|f(x)-f(y)| = |f'(a)(x-y)| = p^m|x-y|.$$

Hence we have a local isometry again and we can determine $z \in \mathbf{Z}$ such that $|z-a| < p^{-m(k)}$ and z has minimal length. Letting $c_0 = z$ we we can repeat the above argument and at the case that we determine the digits c_l^i, instead of checking against 0 we have to check with the appropriate digit of d to determine the value of each c_l^i, and the above estimates for the bounds are the same in this case. □

Essentially the theorem states that the modulus of continuity of the inverse function of f has a bound similar with the bound of the modulus of continuity of the function f under certain hypotheses.

It is interesting to compare the above theorem with the case of polynomially time computable functions over the real numbers. In this case Ker – I Ko [8] proved that there exists a C^∞ polynomially time computable function over the real numbers which has a root which is not polynomially time computable (excluding the case of being identically zero).

Proposition 1. *Let $f\colon \mathbf{Z}_p \to \mathbf{Q}_p$ be given by a power series expansion over $\mathbf{Z}_p$, with domain of convergence of the power series that contains $\mathbf{Z}_p$, f not identically 0 and computable in polynomial time, then all roots of f are computable in polynomial time.*

Proof. Since f is analytic and $\mathbf{Z}_p$ is compact f has finitely many roots. From this fact we have that the roots of f are isolated. Let x be a root, then it will have a multiplicity $m < \infty$. Apply Proposition 5 to $f^{(m-1)}$ where m is the multiplicity of the root. □

After establishing a version of the inverse function theorem it is natural to ask whether there is a version of the implicit function theorem.

Proposition 2. *Let $f\colon \mathbf{Z}_p \times \mathbf{Z}_p \to \mathbf{Q}_p$ be a polynomially time computable function. For $(x_0, w_0) \in \mathbf{Z}_p \times \mathbf{Z}_p$ suppose that*

$$\lim_{(y,z)\to(w_0,w_0)} \frac{f(x_0,y) - f(x_0,z)}{y-z}$$

exists and is not 0*. Then the equation $f(x, y(x)) = 0$ with $y(x_0) = w_0$ defines y as a polynomially time computable function of x, in a neighborhood of (x_0, w_0).*

Proof. Let D_2 denote the partial derivative with respect to the second variable. Fix x_0 and choose a neighborhood $\mathcal{N}$ of (x_0, w_0) such that for all $y, z \in \mathcal{N}$:

$$\Big|\frac{f(x_0,y) - f(x_0,z)}{y-z} - D_2 f(x_0,w_0)\Big| < \big|D_2 f(x_0,w_0)\big|,$$

so that for all $y, z \in \mathcal{N}$ we have:

$$\big|f(x_0,y) - f(x_0,z)\big| = \big|D_2 f(x_0,w_0)\big|\big|y-z\big|.$$

As in the proof of Theorem 5 given m we can search (essentially one digit at each stage) for m stages. At each stage i, $p-1$ calculations are required to calculate a p-adic number z such that $|f(x,z)| \le p^{-i}$. If the first $p-1$ calculations are wrong, then the last one has to be the correct value since we have an isometry.

The total running time of the algorithm now is :

$$(p-1)\sum_{i=0}^{m} q\,(i) \le (p-1)\,(m+1)\,q\,(m)$$

(assuming w.l.o.g. that $q(m)$ is nondecreasing). □

4 Derivatives of Polynomially Time Computable Functions over *p*-Adic Fields

Since most of analysis is concerned with approximate calculations, and in computability and complexity theory the effort is to delineate the effectiveness of these approximate calculations or obtain as sharp quantitative results as possible, the notion of the derivative is an important topic as a way of doing simpler approximate calculations.

The first question is whether there exists any relationship between the notion of differentiation and computability. In the case of the p-adic numbers the situation is similar to the real number case [6] as we have that there exists a function f which is polynomially time computable and f is nowhere differentiable. The proof essentially is a verification that the constructed function has linear modulus of continuity and hence it is computable in polynomial time.

Theorem 6. *There exists a function $f\colon \mathbf{Z}_p \to \mathbf{Q}_p$, such that f is polynomially time computable on $\mathbf{Z}_p$, and $f'(x)$ does not exist for any $x \in \mathbf{Z}_p$.*

Proof. Case 1. ($p \neq 2$, Dieudonné) Define the following function f:

$$\text{for} \quad x = \sum_{i=0}^{\infty} a_i p^i, \qquad \text{let} \qquad f(x) = \sum_{i=0}^{\infty} a_i^2 p^i.$$

with $a_i \in \{0, 1, \ldots, p-1\}$. We have that f is well defined, since $|a_i| \geq |a_i^2|$ so that $\lim_{i\to\infty} |a_i^2 p^i| = 0$. We have:

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}.$$

For $h = bp^n$, with $|b| = 1$ we have that the limits in the summation defining $f(x)$ exist and the convergence is absolute hence we can rearrange the summation and we have:

$$f'(x) = \lim_{n\to\infty} \frac{\sum_{i=0}^{n-1} a_i^2 p^i + (a_n + b)^2 p^n + \sum_{n+1}^{\infty} a_i^2 p^i - \sum_{i=0}^{\infty} a_i^2 p^i}{bp^n} =$$

$$= \lim_{n\to\infty} \frac{(2a_n b + b^2) p^n}{bp^n} = \lim_{n\to\infty} 2a_n + b.$$

We have that this limit depends on a_n. Since $p \neq 2$ the possible values for b are more than one, and the limit depends on b as well, hence the limit does not exist. From the definition of f we have that f is polynomially time computable on $\mathbf{Z}_p$ since it has linear modulus of continuity.

Case 2. ($p = 2$) Consider the representation of x in base 4 instead of base 2 so that we have the following definition for f:

$$\text{for} \quad x = \sum_{i=0}^{\infty} a_i 4^i, \qquad \text{let} \qquad f(x) = \sum_{i=0}^{\infty} a_i^2 4^i.$$

The derivative is:

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}.$$

For $h = b4^n$ with $b \in \{1, 2, 3\}$ we have:

$$f'(x) = \lim_{n\to\infty} \frac{\sum_{i=0}^{n-1} a_i^2 4^i + (a_n + b)^2 4^i + \sum_{i=n+1}^{\infty} a_i 4^i - \sum_{i=0}^{\infty} a_i^2 4^i}{b4^n} =$$

$$= \lim_{n\to\infty} \frac{(2a_n b + b^2) 4^n}{b4^n} = \lim_{n\to\infty} 2a_n + b,$$

and the limit of this expression does not exist as in the previous case. The function f is polynomially time computable since it has linear modulus of continuity. □

The proof for the case $p \neq 2$ is given in Dieudonné [4] without any computability considerations.

5 Maximum Values of Polynomially Time Computable Functions

In discrete complexity theory optimization problems play an important role in delineating complexity classes. Usually the setting is maximizing a function with the possible presence of constraints. A reasonable candidate for a class of optimizing problems in analysis (real and p-adic numbers) is the study of the extrema of functions. In the real case the problems for maximum and minimum are essentially the same, and the study of either kind of extremum is equivalent with the other.

In the p-adic case these two notions seem to give rise to two different cases. The problem of maximizing a function (for a non identically zero function), reduces to the determination of rational number, and once a value has been determined this value will not change. The problem of determining the minimum value includes the problem of finding roots of functions. This is one more difference compared with the real number case.

The problem of minimizing a function in the case of p-adic numbers contains the problem of finding roots. In the case of of polynomially time computable functions from Corollary 1 we have that such numbers can have arbitrarily high complexity.

Theorem 7. *Let $f : \mathbf{Z}_p \to \mathbf{Q}_p$ computable. Suppose $f \not\equiv 0$. The maximum value M of $|f|$ and a point x_0 such that $|f(x_0)| = M$ can be determined in constant time.*

Proof. Let m be the modulus of continuity of f. Order $\mathbf{Z}_p$ lexicographically. Searching (using the above order) we can find x_0 such that $f(x_0) \neq 0$. Let k such that $p^{-k} = |f(x_0)|$.

For $x \in \mathbf{Z}_p$ such that $|x - x_0| < p^{-m(k)}$, we have that $|f(x_0) - f(x)| < p^{-k}$ so suffices to check (in the above order) the values of f at one point of the finitely many disjoint spheres of radius $p^{-m(k)+1}$ that partition $\mathbf{Z}_p$. Either we have that the maximum value of $|f(x)|$ occurs at x_0 i.e. $\max\{|f(x)| : x \in \mathbf{Z}_p\} = p^{-k}$ or the maximum value of $|f(x)|$ occurs at a different point z where $|f(z)| = p^{-l}$ with $l < k$. In the latter case such a point z will be member of the finitely many disjoint spheres of radius $p^{-(m(k)+1)}$. We can take such a z to be of minimum length w.r.t. the lexicographic ordering (omitting trailing zeroes).

The time requirements of the above procedure are independent of the input n hence the maximum value is calculated in constant time. □

Proposition 3. *Let $f(x, y)\colon \mathbf{Z}_p \times \mathbf{Z}_p \to \mathbf{Q}_p$ be computable. Suppose that for all x there exists y such that $f(x, y) \neq 0$, then the mapping $x \mapsto y$ such that $|f(x, y)|$ is maximum, and y is the least in the lexicographic ordering of $\mathbf{Z}_p$ is computable in constant time.*

Proof. Claim: Let $M_x = \max\{|f(x, y)| \mid y \in \mathbf{Z}_p\}$. Then $\min\{M_x \mid x \in \mathbf{Z}_p\} \neq 0$. Proof of claim. Suppose $\min\{M_x \mid x \in \mathbf{Z}_p\} = 0$. Then for any $n \in \mathbf{N}$ there

exists $x_n \in \mathbf{Z}_p$ such that $M_{x_n} < p^{-n}$. By compactness of $\mathbf{Z}_p$ the sequence $\{x_n \mid n \in \mathbf{N}\}$ has a convergent subsequence $\{\hat{x}_n \mid n \in \mathbf{N}\}$. Let $\hat{x} = \lim_{n\to\infty} \hat{x}_n$. Then by Theorem 1 f is continuous so we have $\max\{|f(\hat{x}, y)| \mid y \in \mathbf{Z}_p\} = 0$ which contradicts that for any $x \in \mathbf{Z}_p$ there exists a $y \in \mathbf{Z}_p$ such that $f(x, y) \neq 0$.
End of claim.
For fixed x_0 the function $|f(x_0, y)|$ attains a nonzero maximum because $range(f)$ is compact. Let $m(n)$ be the modulus of continuity of $f(x, y)$ (in both variables). Without loss of generality we may assume that the modulus of continuity is an increasing function.
For each i, starting with $i = 1$, calculate $f(x_0, y)$ for all $y \in \mathbf{Z}_p$ with length$(y) \leq m(i)$. If for some y,

$$|f(x_0, y)| > p^{-i},$$

stop and output the least such y. the result. This will happen for some y since for some y $f(x_0, y) \neq 0$. Suppose y_0 is the result of the calculations. Let $a = f(x_0, y_0)$. Then $|a|$ is the maximum value of f. If $p^{-j} = \max_{y\in\mathbf{Z}_p} |f(x_0, y)| > |a| = p^{-i}$ then the value of y such that $f(x_0, y)$ attains maximum value would be a value such that length$(y) \leq m(j) < m(i)$, which contradicts the fact that the y found was the least one in the lexicographic order of $\mathbf{Z}_p$.

We have that y is computable in constant time because y is a rational integer and if there is a z such that $|f(x_0, z)| = p^{-j}$ then there is a z' such that $f(x_0, z') = p^{-j}$ with length$(z') \leq m(j)$, as m is the modulus of continuity of f. Hence some y_0 exists with length$(y) \leq m(n)$ with $p^{-n} = \min\{M_x \mid x \in \mathbf{Z}_p\}$ and this value y_0 will be found in the above calculations. Once such value for y has been determined this value will not change. □

References

1. O. Aberth. *Computable Analysis.* Mc Graw Hill, 1980.
2. D. S. Bridges. *Computability.* Springer–Verlag, 1994.
3. J. W. S. Cassels. *Local Fields.* Cambridge University Press, Cambridge, 1986.
4. J. Dieudonné. Sur les fonctions continues p-adiques. *Bulletin des Sciences Mathematiques*, 68:79–95, 1944-45.
5. F. Q. Gouvea. *p–adic Numbers, an Introduction.* Springer–Verlag, 1997.
6. H. Friedman, and Ker – I Ko. Computational Complexity of Real Functions. *Theoretical Computer Science*, 20:323–352, 1982.
7. G. Kapoulas. Computable p-adic numbers. *Submitted in J. of Automata, Languages and Complexity.* Also Technical Report No. 115 CDMTCS.
8. Ker – I Ko. *Complexity Theory of Real Functions.* Birkäuser, Boston, 1991.
9. N. Koblitz. *p–adic Numbers, p–adic Analysis and zeta Functions.* Springer–Verlag, New York, 1977.
10. G. Kreisel, D. Lacombe, and J. R. Shoenfield. Partial recursive functionals and effective operators. In A. Heyting, editor, *Constructivity in mathematics (Proceedings of the colloquium held at Amsterdam), 1957*, pages 195 – 207. North – Holland, 1959.
11. B. A. Kushner. *Lectures on Constructive Mathematical Analysis.* Nauka, 1973.

12. L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers : NP-completeness, recursive functions and universal machines. *Bulletin of American Mathematical Society (New Series)*, 21:1–46, 1989.
13. D. Lacombe. Extension de la notion de fonction recursive aux fonctions d' une ou plusiers variables reels. *C. R. Acad. Sci. Paris*, 240:2478–2480, 1955.
14. D. Lacombe. Extension de la notion de fonction recursive aux fonctions d' une ou plusiers variables reels. *C. R. Acad. Sci. Paris*, 241:13–14, 151–153, 1955.
15. D. Lacombe. Remarques sur les operateurs recursifs et sur les fonctions recursive d' une variable reel. *C. R. Acad. Sci. Paris*, 241:1250–1252, 1955.
16. K. Mahler. *p–adic Numbers and their Functions*. Cambridge University Press, Cambridge, second edition, 1981.
17. Y. Moschovakis. Notation systems and recursive ordered fields. *Comp. Math.*, 17:40–71, 1965.
18. A. Ostrowski. Über einige Lösungen der Funktionalgleichunng $\phi(x)\phi(y) = \phi(x\,y)$. *Acta Math.*, 41:271 – 284, 1918.
19. M. B. Pour–El and J. I. Richards. *Computability in Analysis and Physics*. Springer–Verlag, 1988.
20. H. G. Rice. Recursive real numbers. *Proc. Amer. Math. Soc.*, 5, 1954.
21. W. H. Shikhoff. *Ultrametric Calculus*. Cambridge University Press, Cambridge, 1984.
22. K. Skandalis. Non recursiveness of the operations on real numbers. *Theoretical Computer Science*, 71:425 – 429, 1990.
23. A. M. Turing. On computable numbers, with an application to the Entscheidungs problem. *Proc. London Math. Society*, 42:230–265, 1937.

On the Computational Content of the Krasnoselski and Ishikawa Fixed Point Theorems

Ulrich Kohlenbach

BRICS[⋆]
Department of Computer Science, University of Aarhus
Ny Munkegade, DK-8000 Aarhus C, Denmark
kohlenb@brics.dk

Abstract. This paper is part of a case study in proof mining applied to non-effective proofs in nonlinear functional analysis. More specifically, we are concerned with the fixed point theory of nonexpansive selfmappings f of convex sets C in normed spaces. We study Krasnoselski and more general so-called Krasnoselski–Mann iterations which converge to fixed points of f under certain compactness conditions. But, as we show, already for uniformly convex spaces in general no bound on the rate of convergence can be computed uniformly in f. However, the iterations yield even without any compactness assumption and for arbitrary normed spaces approximate fixed points of arbitrary quality for bounded C (asymptotic regularity, Ishikawa 1976). We apply proof theoretic techniques (developed in previous papers) to non-effective proofs of this regularity and extract effective uniform bounds (with elementary proofs) on the rate of the asymptotic regularity. We first consider the classical case of uniformly convex spaces which goes back to Krasnoselski (1955) and show how a logically motivated modification allows to obtain an improved bound. Moreover, we get a completely elementary proof for a result which was obtained in 1990 by Kirk and Martinez-Yanez only with the use of the deep Browder–Göhde-Kirk fixed point theorem.
In section 4 we report on new results ([29]) we established for the general case of arbitrary normed spaces including new quantitative versions of Ishikawa's theorem (for bounded C) and its extension due to Borwein, Reich and Shafrir (1992) to unbounded sets C. Our explicit bounds also imply new qualitative results concerning the independence of the rate of asymptotic regularity from various data.

⋆ Basic Research in Computer Science, Centre of the Danish National Research Foundation.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 119–145, 2001.

1 General Introduction

This paper (and its companion [29]) is another case study in the project of 'proof mining' [1] in analysis by which we mean the logical analysis of mathematical proofs (typically using non-effective analytical tools) with the aim of extracting new numerically relevant information (e.g. effective uniform bounds or algorithms etc.) hidden in the proofs.[2]

Many problems in numerical (functional) analysis can be seen as instances of the following general task: construct a solution x of an equation

$$A(x) :\equiv (F(x) = 0),$$

where x is an element of some Polish (i.e. complete separable metric) space (typically with additional structure) and $F : X \to \mathbb{R}$ (usually F will depend on certain parameters a which again belong to Polish spaces). Quite often the construction of such a solution is obtained in two steps:

1) One shows how to construct (uniformly in the parameters of A) approximate solutions (sometimes called 'ε-solutions') $x_\varepsilon \in X$ for an ε-version of the original equation
$$A_\varepsilon(x) :\equiv \big(|F(x)| < \varepsilon\big).$$
2) Exploiting compactness conditions on X one concludes that either $(x_{\frac{1}{n}})_{n\in\mathbb{N}}$ itself or some subsequence of it converges to a solution of $A(x)$.

The first step usually is constructive. However, the non-effectivity of the second step in many cases prevents one from being able to compute a solution $\widehat{x}$ of A effectively within a prescribed error $\frac{1}{k}$, i.e. to compute a function $n(k)$ such that $d_X(x_{n(k)}, \widehat{x}) < \frac{1}{k}$. In many cases $X := K$ is compact and $\widehat{x}$ is uniquely determined. Then (x_n) itself converges to $\widehat{x}$ so that no subsequence needs to be selected. However, the problem of how to get a-priori bounds (in particular not depending on $\widehat{x}$ itself) on the rate of convergence of that sequence remains. In numerical analysis, often such rates are not provided (due to the ineffectivity of the proof of the uniqueness of $\widehat{x}$).[3]
In a series of papers we have demonstrated the applicability of proof theoretic techniques to extract so-called uniform moduli of uniqueness (which generalize

[1] The term 'proof mining' (instead of G. Kreisel's 'unwinding of proofs') for the activity of extracting additional information hidden in given proofs using proof theoretic tools was suggested to the author by Professor Dana Scott.
[2] For a different case study in analysis in the context of best approximation theory see [21],[22]. For other kinds of logical analyzes of specific proofs see [33] and [36].
[3] In interesting critical discussion of this and related issues can be found in [32].

the concept of strong unicity as used e.g. in Chebycheff approximation theory) from non-constructive uniqueness proofs and to use them to get effective rates of convergence (e.g. [25], [21],[22],[23],[30] for concrete applications to approximation theory).

In this paper we carry out a logical analysis of examples for the first of the two steps mentioned above in situations where an effective solution of 2) is not possible (mainly due to the lack of uniqueness) and already the fact that the sequence $y_n := x_{\frac{1}{n}}$ yields better and better approximate solutions is proved non-constructively (using sequential compactness).

These applications to 1) fall under (an extension of) the same general logical scheme as our previous applications to 2). In a series of papers ([23],[24],[25],[27] among others) we have developed general meta-theorems which guarantee the extractability of uniform bounds from proofs which are allowed to make use of substantial parts of analysis. In particular, we specified situations where (due to the fact that only weak forms of induction are used) exponential and even polynomial bounds are guaranteed. Furthermore, these results show that many lemmas used in such proofs do not need to be analyzed (since they do not contribute to the bound) because of their logical form. The proofs of these meta-theorems actually provide an extraction algorithm (based on certain proof-theoretic transformations of the specific proof to be analyzed). So applied to a given proof p in analysis we get another proof p^* which provides more numerical information. When this transformation is carried out explicitly we obtain a new ordinary mathematical proof of a stronger statement which no longer relies on any logical tools at all. Of course, the general proof-theoretic algorithm will usually be used only as a guideline but not followed step by step in the actual construction of p^* (unless this is necessary).

The special case of our general meta-theorems which is relevant for the present paper has the following form:
Let X be a Polish space, K a compact metric space and A_1 a purely existential property. If a theorem of the form

$$(*)\ \forall n \in \mathbb{N} \forall x \in X \forall y \in K \exists m \in \mathbb{N} A_1(n, x, y, m)$$

has been proved in certain formal systems $\mathcal{T}$ for (fragments of) analysis, then one can extract a computable uniform bound $\Phi(n, x)$[4] for $\exists m$, i.e.

$$(**)\ \forall n \in \mathbb{N} \forall x \in X \forall y \in K \exists m \leq \Phi(n, x) A_1(n, x, y, m).$$

An important feature of the bound $\Phi(n, x)$ is that it does not depend on $y \in K$. Typically, $\exists m A_1$ is monotone in m so that the bound $\Phi(n, x)$ actually realizes

[4] This bound (as well as the logical form of A_1) will in general depend on the specific representation of $x \in X$ used in $\mathcal{T}$.

the quantifier. In [25] we have specified a system **PBA** of polynomially bounded analysis which guarantees that $\Phi(n,x)$ will be a polynomial. If we add the exponential function to **PBA** we obtain a system **EBA** which guarantees that Φ uses at most a finite iteration of exp (so if exp is not iterated at all the bound will be exponential in n relative to x). Whereas for our first application in the present paper (theorem 4 below) this result for **PBA** is already sufficient for providing the general logical framework, our analysis of a proof from [4] carried out in theorem 7 needs an extended version due to the use of a principle used in that proof which is not available in **PBA** or **EBA**. Whereas these systems contain quite some parts of non-constructive analysis, principles based on sequential compactness are not included. The significant and highly non-trivial impact of such principles for the extraction of bounds has been determined completely in [27] and [28]. We only discuss the results for the particular simple case of the principle

$$\mathrm{PCM}(a_k) :\equiv [\forall n(0 \le a_{n+1} \le a_n) \to \exists a \in \mathbb{R}_+(\lim_{n\to\infty} a_n = a)]$$

of convergence for bounded monotone sequences $(a_n)_{n\in\mathbb{N}}$ of reals, as it is this principle which is used in the proof from [4] we are going to analyze. In systems like **PBA**, real numbers are represented as Cauchy sequences of rational numbers with fixed rate of convergence. Because of this representation $\mathrm{PCM}(a_n)$ is a fairly strong principle equivalent to

$$(+)\ \forall n\big(0 \le a_{n+1} \le a_n\big) \to \exists f : \mathbb{N} \to \mathbb{N} \forall k, m\big(m \ge f(k) \to a_{f(k)} - a_m \le \frac{1}{k+1}\big).$$

Because of the existence of the 'Cauchy modulus f', '$\forall (a)_n\ \mathrm{PCM}(a_n)$' is equivalent to the principle of so-called arithmetical comprehension which potentially creates bounds of huge complexity when added to systems like **PBA**, **EBA** (see [28]). What we showed in [27] is that things are quite different when $\mathrm{PCM}(a_n)$ is only applied to sequences (a_n) in a given proof of a theorem $(*)$ which can be explicitly defined in terms of the parameters n, x, y of $(*)$. Then, relative to **PBA** and **EBA**, the use of $\mathrm{PCM}(a_n)$ can be reduced to its arithmetical version[5]

$$\mathrm{PCM}_{ar}(a_n) :\equiv \left[\forall n\big(0 \le a_{n+1} \le a_n\big) \to \forall k \exists n \forall m\big(m \ge n \to a_n - a_m \le \frac{1}{k+1}\big)\right].$$

By further proof theoretic considerations, the use of PCM_{ar} can even be reduced to that of its so-called 'no-counterexample interpretation' (or 'Herbrand normal

[5] Because of $(+)$, $\mathrm{PCM}(a_n)$ essentially is the so-called Skolem normal form of $\mathrm{PCM}_{ar}(a_n)$. In general it is NOT possible (in a context like **PBA** or **EBA**) to reduce the use of the Skolem normal form of an arithmetical principle A to A itself. The fact that this IS possible for PCM_{ar} makes profound use of the fact that this principle satisfies a strong monotonicity property, see [26].

form') $\mathrm{PCM}^H_{ar}(a_n) :\equiv$

$$\left[\forall n\big(0 \le a_{n+1} \le a_n\big) \to \forall k, g \exists n\big(g(n) \ge n \to a_n - a_{g(n)} \le \frac{1}{k+1}\big)\right].$$

The computational significance of this reduction is, that in contrast to the quantifier dependency '$\forall k \exists n$' in PCM_{ar}, which in general has no computable bound, the quantifier '$\exists n$' in PCM^H_{ar} can be bounded (uniformly in k, g and an upper bound $N \in \mathbb{N}$ of a_0) by $\widehat{\Psi}(k, g, N) := \max\limits_{i \le (k+1)N} \Psi(i, g)$, where $\Psi(k, g)$ is the k-times iteration of g applied to 0, i.e.

$$\Psi(0, g) := 0,\ \Psi(k+1, g) := g(\Psi(k, g))$$

(see [27] for details on all this). We like to stress, that this quantitative bound for $\mathrm{PCM}^H_{ar}(a_n)$ only depends on (a_n) via an upper bound $N \ge a_0$ whereas a bound for $\exists n$ in $\mathrm{PCM}_{ar}(a_n)$ of course has to depend on (a_n). So the reduction from $\mathrm{PCM}_{ar}(a_n)$ to $\mathrm{PCM}^H_{ar}(a_n)$ also provides an important independence from (a_n) which will play a crucial role in our proof of corollaries 3 and 4 below.
We have seen that the use of $\mathrm{PCM}(a_n)$ for definable (a_n) contributes to the bound Φ in $(**)$ by $\widehat{\Psi}$. By finite iteration, Ψ (and hence $\widehat{\Psi}$) is able to produce arbitrary primitive recursive growth rates. However, in practice it will usually only be applied to some fixed functions g which can explicitly be defined in terms of the parameters of the problem. It is the construction of these functions which plays a crucial role in the process of proof mining. This fact is clearly reflected in the bounds we obtain in theorem 7 and corollary 3 below (see the definition of $\widehat{\alpha}$ in these results).[6]

We have discussed the logical background of our results in order to convince the reader, that these results are instances of a general scheme for proof mining. Once one is familiar with this scheme, one can produce improvements of existence theorems of the kind we illustrate here using examples from fixed point theory also in other areas of analysis.

[6] In the concrete application in this paper it is mainly the reduction from PCM_{ar} to PCM^H_{ar} which plays a significant role in the proof mining. The (in general much more complicated) reduction from PCM to PCM_{ar} is almost straightforward. However, we expect that this will be different for other examples. In any case, we believe that the fact that our applications to concrete proofs reflect crucial steps of the general proof-theoretic reduction and are instances of a general meta-theorem (which at least under an additional compactness assumption predicts the type of results we obtain), makes it justified to call them genuine applications of logic in the sense discussed in [10].

2 Applications to the Fixed Point Theory of Nonexpansive Mappings

In this paper we will analyse proofs from the fixed point theory of nonexpansive mappings $f : C \to C$ for certain sets C in normed spaces X.

Definition 1. *Let $(X, \|\cdot\|)$ be a normed linear space and $S \subseteq X$ be a subset of X. A function $f : S \to S$ is called nonexpansive if*

$$(*)\ \forall x, y \in S\big(\|f(x) - f(y)\| \leq \|x - y\|\big).$$

Whereas the fixed point theory for mappings with Lipschitz constant < 1 (i.e. contractions) is essentially trivial (even from a computational point of view) because of the well-known Banach fixed point theorem, the fixed point theory for nonexpansive mappings has been one of the most active research areas in nonlinear functional analysis from the 50's until today. Let us indicate how the picture known for contractions breaks down for nonexpansive mappings:

1) Whereas in Banach's fixed point theorem no boundedness conditions are necessary, fixed points of a nonexpansive mapping will not exist unless the set C is at least bounded: take $X := C := \mathbb{R}$ and $f(x) := x + 1$.
2) Even when C is compact (and therefore fixed points exist by the fixed point theorems of Brouwer and Schauder), they are not uniquely determined: take $X := \mathbb{R}, C := [0, 1]$ and $f(x) = x$.
3) Even when the fixed point is uniquely determined, it will in general not be approximated by the Banach iteration $x_{n+1} := f(x_n)$: take $X := \mathbb{R}, C := [0, 1], f(x) := 1 - x$ and $x_0 := 0$. Then x_n alternates between 0 and 1.

The early history of the fixed point theory for nonexpansive mappings rests on two main theorems which both use a geometric assumption on the normed space X, namely that it is uniformly convex:

Definition 2 ([6]). *A normed linear space $(X, \|\cdot\|)$ is uniformly convex if*

$$\forall \varepsilon > 0 \exists \delta > 0 \forall x, y \in X\big(\|x\|, \|y\| \leq 1 \wedge \|x - y\| \geq \varepsilon \to \|\frac{1}{2}(x + y)\| \leq 1 - \delta\big).$$

A function $\eta : (0, 2] \to (0, 1]$ providing such a $\delta := \eta(\varepsilon) > 0$ for given $\varepsilon > 0$ is a modulus of uniform convexity.

The following fundamental existence theorem for fixed points of nonexpansive mappings and uniformly convex Banach spaces was proved independently by Browder, Göhde and Kirk (note that no compactness assumption is made in this result):

Theorem 1 ([5],[13],[17]). *Let $(X, \|\cdot\|)$ be a uniformly convex Banach space, $C \subseteq X$ a non-empty convex closed and bounded subset of X and $f : C \to C$ a nonexpansive mapping. Then f has a fixed point.*

Remark 1. In 1975, a counterexample showing that the assumption of X being uniformly convex in theorem 1 cannot be omitted was found (see [7], p. 37).

Another fundamental theorem in the fixed point theory for nonexpansive mappings is the following result due to Krasnoselski, which shows that (under an additional compactness condition, which by the Schauder fixed point theorem guarantees the existence of a fixed point) a fixed point of f can be approximated by a special iteration sequence:

Theorem 2 (Krasnoselski [31]). *Let K be a non-empty convex closed and bounded set in a uniformly convex Banach space $(X, \|\cdot\|)$ and f a nonexpansive mapping of K into a compact subset of K. Then for every $x_0 \in K$, the sequence*

$$x_{k+1} := \frac{x_k + f(x_k)}{2}$$

converges to a fixed point $z \in K$ of f.

Remark 2. Due to a much more general result from [16], which we will discuss below, the assumption of X being uniformly convex is actually superfluous in theorem 2.

We will show below that there cannot be an effective procedure to compute a rate of convergence of the iteration in the Krasnoselski fixed point theorem uniformly in f and the starting point $x \in K$ of the iteration. This already holds for the special case of $X := \mathbb{R}$ and $K := [0, 1]$ and the fixed starting point $x_0 := 0$ as there exists no computable function F from the set of all nonexpansive functions $f : [0, 1] \to [0, 1]$ into $[0, 1]$ which computes uniformly in f a fixed point of f (this is closely related to the fact that such a function F cannot be continuous with respect to the maximum norm $\|f\|_\infty$). Logically, this ineffectivity in Krasnoselski's theorem corresponds to the fact the statement that $(x_k)_{k\in\mathbb{N}}$ converges is Π^0_3.
On the other hand if we consider the weaker question of how far we have to go in the iteration to obtain an ε-fixed point, then we notice that the logical form of the statement

$$(*)\ \forall k \in \mathbb{N} \exists n \in \mathbb{N}\big(\|x_n - f(x_n)\| < \frac{1}{k+1}\big)$$

is Π^0_2 (assuming that real numbers are represented as Cauchy sequences with fixed rate of convergence so that $<_{\mathbb{R}} \in \Sigma^0_1$). That is why we are able to extract

an algorithm for n in $(*)$ uniformly in x_0 and f (if X, C have a computable representation).
The following crucial monotonicity property holds (see lemma 2 below):

$$\|x_{m+1} - f(x_{m+1})\| \leq \|x_m - f(x_m)\|$$

Hence the formula

$$\|x_n - f(x_n)\| < \frac{1}{k+1}$$

is equivalent to

$$\forall m \geq n\big(\|x_m - f(x_m)\| < \frac{1}{k+1}\big).$$

Thus any bound on $(*)$ provides a rate of convergence for

$$(**)\ \|x_n - f(x_n)\| \overset{n\to\infty}{\to} 0,$$

where $(**)$ is called the asymptotic regularity of (x_n) (we will see below that this asymptotic regularity holds without any compactness assumption).
Let us assume for the moment that $C := K$ itself is compact. Then the standard proof (as given in e.g. [3]) of the Krasnoselski fixed point theorem (more precisely of its consequence $(**)$ above) directly fits into the general extraction scheme discussed above. Besides basic arithmetical reasoning only the existence of a fixed point $y \in K$ of f (which follows from the Schauder fixed point theorem) is used to show $(*)$. Since the statement

$$(a)\ \exists y \in K\big(\|f(y) - y\| = 0\big)$$

has the logical form of those assumptions which do not contribute to the growth of extractable bounds and which furthermore can be reduced to their ε-weakening

$$(b)\ \forall \varepsilon > 0 \exists y \in K\big(\|f(y) - y\| < \varepsilon\big)$$

and since furthermore the starting point $x_0 \in K$ belongs to a compact set and the set of all nonexpansive mappings $f : K \to K$ is also compact, we know a-priori that the extractability of a uniform bound (of low complexity) for n in $(*)$ which does not depend on x_0 or f (but only on ε and a modulus of uniform convexity) is guaranteed and its verification only uses (b). The actual extraction shows that instead of the compactness only the boundedness of K is needed. This is even true for the reduction of (a) to (b) which allows furthermore to remove the assumption on K being closed (and X being complete), since the existence of approximate fixed points (but not of fixed points) can be shown without these assumptions. This, of course, is a-posteriori information which was

not guaranteed by a general logical result. Nevertheless, the a-priori information provided for the special case with K being compact prompted the search for such uniform bounds. As a result we get for an arbitrary convex bounded subset $C \subset X$ a uniform bound for $(*)$ depending only on ε, a modulus of uniform convexity η of X and an upper bound for the diameter of C. The bound itself is not new: for the special compact case it is essentially already contained in Krasnoselski's original paper ([31]) and was proved for the case of closed bounded convex sets in [18] using the deep Browder–Göhde–Kirk fixed point theorem. We nevertheless carry out the analysis because it shows two phenomena:
1) the possibility of replacing the existence of fixed points by the existence of ε-fixed points which permits a completely elementary verification of the bound (without assuming X complete or C closed), which does not even rely on the Schauder fixed point theorem used by Krasnosleski in the compact case. Even the qualitative asymptotic regularity had been obtained before only either with the use of the Browder–Göhde–Kirk fixed point theorem or as a corollary of a much more general result due to Ishikawa which we discuss below.
2) There is a logical modification of the proof from [3] which makes use of the above mentioned no-counterexample interpretation PCM_{ar}^{H} of PCM_{ar} and allows the use of a certain multiplicative property typically satisfied by moduli η, by which we obtain (for such moduli) a numerically better result. As a special instance of this we get a bound which is polynomial in ε of degree p for the spaces L_p with $p \geq 2$ (a result which (for this special case only) was first obtained in 1990 in [18] by an ad hoc calculation).[7] For $X := \mathbb{R}, C := [0,1]$ we even get a linear bound (see also [18], p.192).

We now move on to vast extensions of Krasnoselski's fixed point theorem. In [16] it is shown that Krasnoselski's fixed point theorem even holds without the assumption of X being uniformly convex and for much more general so-called Krasnoselski–Mann iterations

$$x_{k+1} := (1 - \lambda_k)x_k + \lambda_k f(x_k),$$

where λ_k is a sequence in $[0,1]$ which is divergent in sum and satisfies $\limsup_{k\to\infty} \lambda_k < 1$.
Furthermore, [16] establishes that for such iterations

$$\text{(I)} \quad \lim_{k\to\infty} \|x_k - f(x_k)\| = 0,$$

where X is an arbitrary normed linear space, C any bounded convex subset of X and $f : C \to C$ is nonexpansive.

[7] For $p = 2$, no better bound is known. For $p \geq 3$ a better bound was only obtained by the extremely complicated work of [1].

In [4], a generalization of this result to the case of unbounded convex sets C is proved:

$$\text{(II)} \quad \lim_{k\to\infty} \|x_k - f(x_k)\| = r_C(f),$$

where

$$r_C(f) := \inf_{x\in C} \|x - f(x)\|$$

will in general be strictly positive. Note that (by lemma 1 below) $r_C(f) = 0$ for bounded convex C. Hence (II) entails (I) as a special case.

In section 4, we will report on results from [29] which for the first time provide a quantitative analysis of (II) (see theorem 7). These results were obtained as an instance of our general result on the extractability of bounds from proofs using $\mathrm{PCM}(a_k)$ for a sequence $(a_k)_{k\in\mathbb{N}}$ which is definable in the parameters of the problem. In the case at hand $(a_k)_{k\in\mathbb{N}}$ is just $(\|x_k - f(x_k)\|)_{k\in\mathbb{N}}$. By specializing the resulting bound to the case where C is bounded we get a uniform bound for (I) which only depends on ε, an upper bound d_C for the diameter $d(C)$ of C and some rather general information on (λ_k) (see corollary 3). In particular the bound is independent of f, the starting point x_0 and the space X. Such uniformity results are of great interest in the area of nonlinear functional analysis. In the final section of this paper we discuss the long history of partial results towards our new full uniformity result from [29].

These applications clearly show the usefulness of logical proof mining even if one is not primarily interested in quantitative results like the numerical quality of the bounds (or the bounds extractable happen to be too large to be useful in practice) but is interested in new qualitative results on the independence of the quantity in question from certain input data.[8]

3 Effective Uniform Bounds on the Krasnoselski Iteration in Uniformly Convex Spaces

We start by showing that the rate of convergence of the Krasnosleski iteration in Krasnolselski's fixed point theorem is in general not computable (uniformly in the input data). We then show in the main part of this section that, in contrast to this negative result, one can obtain computable rates of convergence (of low complexity) for the asymptotic regularity of x_n, i.e. for $\|x_n - f(x_n)\| \to 0$. This

[8] For another instance of this see our explicit uniform constants of strong unicity for Chebycheff approximation which were extracted in [21],[22] from classical uniqueness proofs for the best Chebycheff approximation (known already since about 1905-1917). These constants in particular imply the existence of a common constant of unicity for compact sets K of functions $f \in C[a,b]$, if $\inf_{f\in K} dist(f,H) > 0$ (H a Haar space), a fact that was proved in approximation theory only in 1976 and non-effectively (see [15]).

is even true without any compactness, closedness or completeness assumptions on the convex set C or the space X.

Let $(X, \|\cdot\|)$ be a uniformly convex normed space, $K \subset X$ a compact and convex set and $f : K \to K$. By the Krasnoselski fixed point theorem we know that the Krasnoselski iteration (x_n) converges to a fixed point of f. We now show that already for $X := \mathbb{R}, K := [0,1]$ and a very simple class of nonexpansive mappings f, no rate of convergence for (x_n) (starting from $x_0 := 0$) can be computed uniformly in f.

Theorem 3. *There is no Turing machine M^α which uniformly in $\alpha : \mathbb{N} \to \{0,1\}$ as an oracle computes a number m such that*

$$\forall j \geq m\big(|x_j - x_m| < \frac{1}{2}\big),$$

where

$$x_0 := 0,\ x_{n+1} := \tfrac{x_n + f_{\lambda_\alpha}(x_n)}{2} \text{ and}$$
$$f_{\lambda_\alpha}(x) := \lambda_\alpha x + 1 - \lambda_\alpha,\ \text{where } \lambda_\alpha := \sum_{i=0}^{\infty} \alpha(i) 2^{-i-1}.$$

Proof: Assume that there is a Turing machine M^α which computes an m satisfying

$$\forall j \geq m\big(|x_j - x_m| < \frac{1}{2}\big).$$

One easily verifies that

$$(1)\ \lambda_\alpha < 1 \Rightarrow x_n \to 1 \Rightarrow x_m \in [\tfrac{1}{2}, 1] \text{ and}$$
$$(2)\ \lambda_\alpha = 1 \Rightarrow \forall n(x_n = 0) \Rightarrow x_m = 0.$$

Since with m also x_m is computable uniformly in α and one can decide whether $x_m \in [\frac{1}{2}, 1]$ or $x_m = 0$, one can decide uniformly in α whether $\lambda_\alpha < 1$ or $\lambda_\alpha = 1$, i.e. whether $\exists n(\alpha(n) \neq 1)$ or $\forall n(\alpha(n) = 1)$, which is impossible. ⊣

Remark 3. The representation of $\lambda \in [0,1]$ via α in the proof above is very strong in that it provides a rather special Cauchy sequence of rationals with fixed rate of convergence which in general cannot be uniformly computed in an arbitrary Cauchy sequence of rationals with fixed rate of convergence. However, this makes the non-computability result even stronger in that not even this strong representation of the input allows one to compute a fixed point.

Definition 3. *Let $(X, \|\cdot\|)$ be a normed linear space, S a subset of X, $f : S \to S$ and $\varepsilon > 0$. A point $x \in S$ is called an ε-fixed point of f if $\|x - f(x)\| \leq \varepsilon$.*

Lemma 1. *Let $(X, \|\cdot\|)$ be a normed linear space, let $\emptyset \neq C \subseteq X$ be convex with bounded diameter $d(C) < \infty$ and let $f : C \to C$ be nonexpansive. Then f has ε-fixed points in C for every $\varepsilon > 0$.*

Proof: Since the lemma is trivial for $\varepsilon > d(C)$, we may assume that $\varepsilon \leq d(C)$. To reduce the situation to the Banach fixed point theorem we use the following well-known construction (see e.g. [4] but also [13]): $f_t(x) := (1-t)f(x) + tc$ for some $c \in C$ and $t \in (0,1]$. $f_t : C \to C$ is a contraction and therefore Banach's fixed point theorem applies. Note furthermore that the completeness assumption in Banach's theorem is needed only to guarantee the existence of a limit of the Cauchy sequence $(f_t^n(c))_{n\in\mathbb{N}}$, where f_t^n denotes the n-times iteration of f_t, which is not necessary to ensure that $f_t^n(c)$ is an ε-fixed point of f_t for sufficiently large n and hence (for $t := \varepsilon/d(C)$) a 2ε-fixed point of f. That is why we do not have to assume that X is complete or that C is closed. ⊣

The following lemma belongs to the 'folklore' of the subject. We include its simple proof for the sake of completeness.

Lemma 2. *Let $(X, \|\cdot\|)$ be a normed linear space, let $C \subseteq X$ be a convex subset of X and let $f : C \to C$ be a nonexpansive function. Let $x_0 \in C$ be arbitrary and define $x_{k+1} := \frac{x_k + f(x_k)}{2}$. Then*

$$\forall k\big(\|x_{k+1} - f(x_{k+1})\| \leq \|x_k - f(x_k)\|\big).$$

Proof:

$$\begin{array}{l}\|x_{k+1} - f(x_{k+1})\| = \|\frac{1}{2}x_k + \frac{1}{2}f(x_k) - f(\frac{1}{2}x_k + \frac{1}{2}f(x_k))\| = \\ \|(\frac{1}{2}x_k - \frac{1}{2}f(x_k)) + (f(x_k) - f(\frac{1}{2}x_k + \frac{1}{2}f(x_k)))\| \leq \\ \|\frac{1}{2}x_k - \frac{1}{2}f(x_k)\| + \|f(x_k) - f(\frac{1}{2}x_k + \frac{1}{2}f(x_k))\| \leq \\ \|\frac{1}{2}x_k - \frac{1}{2}f(x_k)\| + \|x_k - (\frac{1}{2}x_k + \frac{1}{2}f(x_k))\| = \\ \frac{1}{2}\|x_k - f(x_k)\| + \frac{1}{2}\|x_k - f(x_k)\| = \|x_k - f(x_k)\|.\end{array}$$

⊣

Quantitative Analysis of the Proof of Theorem 2 in [3]:

We now give two quantitative versions of the consequence

$$(*)\ \forall k \in \mathbb{N}\exists n \in \mathbb{N}\big(\|x_n - f(x_n)\| < \frac{1}{k+1}\big)$$

of theorem 2 discussed in the previous section. The first one follows directly the proof of the theorem as given in [3]. The second one uses a logical modification of that proof which is motivated by our general elimination procedure for PCM_{ar}. This second analysis allows to take into account in a very easy way a property which is satisfied by many moduli of uniform convexity, e.g. for all spaces L_p with $p \geq 2$, which makes it possible to improve the results obtained from the first, direct analysis for such spaces.

General Logical Preliminaries:

Let us for the moment assume that K itself is compact. In Bonsall's [3] proof of theorem 2, the following is established (where $x_0 := x,\ x_{k+1} := (x_k + f(x_k))/2$ is the Krasnoselski iteration starting from x):

$$\forall x, y \in K \forall \varepsilon > 0 \big(f(y) = y \wedge \forall k (\|f(x_k) - x_k\| \geq \varepsilon) \to \lim_{n \to \infty} \|x_n - y\| = 0 \big)$$

and hence

$$\forall x, y \in K \forall \varepsilon > 0 \big(f(y) = y \wedge \forall k (\|f(x_k) - x_k\| \geq \varepsilon) \to \exists n (\|x_n - y\| < \varepsilon) \big),$$

where the existence of a fixed point $y \in K$ is derived from the Schauder fixed point theorem. This can be rephrased in the following form

$$\forall x, y \in K \forall \varepsilon > 0 \exists k, n, l \in \mathbb{N} \underbrace{\left(\|f(y) - y\| \leq \frac{1}{l+1} \wedge \|f(x_k) - x_k\| \geq \varepsilon \to \exists \tilde{n} \leq n (\|x_{\tilde{n}} - y\| < \varepsilon) \right)}_{\in \Sigma^0_1}.$$

By our general results on the extractability of uniform bounds we know a priori (using the compactness of K as well as of the space of all nonexpansive mappings $f : K \to K$) that we can extract bounds $K(\varepsilon), N(\varepsilon), L(\varepsilon)$ (and hence because of the monotonicity in k, n, l of the formula above, which follows from lemma 2, also realizations) for k, n, l which are independent of $x, y \in K$ and f and only depend on $\varepsilon > 0$ (and a modulus of uniform convexity η of X). Since we may assume that $L(\varepsilon) > \frac{1}{\varepsilon}$ and since by the nonexpansivity of f

$$\|f(y) - y\| \leq \varepsilon \wedge \|x_{\tilde{n}} - y\| \leq \varepsilon \to \|f(x_{\tilde{n}}) - x_{\tilde{n}}\| \leq 3\varepsilon,$$

this yields

$$\exists n \leq \max \big(K(\varepsilon), N(\varepsilon) \big) \big(\|f(x_n) - x_n\| \leq 3\varepsilon \big),$$

and so again by lemma 2

$$\forall n \geq \max \big(K(\varepsilon), N(\varepsilon) \big) \big(\|f(x_n) - x_n\| \leq 3\varepsilon \big).$$

Thus we have obtained a uniform bound and at the same time reduced the assumption '$\exists y \in K (f(y) = y)$' to '$\forall \varepsilon > 0 \exists y \in K (\|f(y) - y\| < \varepsilon)$'. In particular, as the bound does not depend on y, the computation of such an approximate fixed point and hence an analysis of the proof of its existence is not needed.
The actual extraction of the bound carried out below reveals that such uniform bounds K, N, L even exist when the compactness assumption on K is replaced by the boundedness of K. Since by lemma 1 the existence of approximate fixed

points (but not of fixed points) in this much more general setting is even guaranteed for spaces X which are not complete, we can remove this assumption as well and the result is proved without appeal to any fixed point theorem other than Banach's (actually only its ε-version):

Theorem 4 (Direct Analysis of Bonsall's [3] Proof of Theorem 2). *Let $(X, \|\cdot\|)$ be a uniformly convex normed space with modulus of convexity $\eta : (0,2] \to (0,1]$ and $C \subseteq X$ be a non-empty convex set with*

$$d(C) := \sup_{x_1, x_2 \in C} \|x_1 - x_2\| \leq d_C \in \mathbb{Q}^*_+.$$

Let $f : C \to C$ be a nonexpansive function.
Define for arbitrary $x \in C$

$$x_0 := x, \quad x_{k+1} := \frac{x_k + f(x_k)}{2}.$$

Then

$$\forall x \in C \forall \varepsilon > 0 \forall k \geq h(\varepsilon, d_C)\big(\|x_k - f(x_k)\| \leq \varepsilon\big),$$

where $h(\varepsilon, d_C) := \left\lceil \frac{\ln(4d_C) - \ln(\varepsilon)}{\eta(\varepsilon/(d_C+1))} \right\rceil$ for $\varepsilon < d_C$ and $h(\varepsilon, d_C) := 0$ otherwise.

Proof: The theorem is trivial for $\varepsilon \geq d_C$. So we can assume that $\varepsilon < d_C$. By lemma 1, f has ε-fixed points $x_\varepsilon \in C$, $\|f(x_\varepsilon) - x_\varepsilon\| < \varepsilon$ for every $\varepsilon > 0$. Let $\delta > 0$ be such that $\delta < \min(1, \frac{\varepsilon}{12h(\varepsilon, d_C)})$ and let $y \in C$ be a δ-fixed point of f, i.e.

$$(1)\ \|y - f(y)\| < \delta.$$

Assume that

$$(2)\ \|x_k - f(x_k)\| = \|(x_k - y) - (f(x_k) - y)\| > \varepsilon.$$

Then

$$(3)\ \left\| \frac{x_k - y}{\|x_k - y\| + \delta} - \frac{f(x_k) - y}{\|x_k - y\| + \delta} \right\| > \frac{\varepsilon}{\|x_k - y\| + \delta} \geq \frac{\varepsilon}{d_C + 1}.$$

Because of

$$(4)\ \|f(x_k) - y\| \overset{(1)}{\leq} \|f(x_k) - f(y)\| + \delta \leq \|x_k - y\| + \delta,$$

we have

$$(5)\ \left\| \frac{x_k - y}{\|x_k - y\| + \delta} \right\|, \left\| \frac{f(x_k) - y}{\|x_k - y\| + \delta} \right\| \leq 1$$

and therefore

$$(6)\ \left\|\frac{1}{2}\left(\frac{x_k - y}{\|x_k - y\| + \delta} + \frac{f(x_k) - y}{\|x_k - y\| + \delta}\right)\right\| \leq 1 - \eta(\varepsilon/(d_C + 1)).$$

Hence

$$(7)\ \begin{cases} \|x_{k+1} - y\| = \|\frac{1}{2}(x_k + f(x_k)) - y\| = \|\frac{1}{2}(x_k - y + f(x_k) - y)\| \leq \\ \big(1 - \eta(\varepsilon/(d_C + 1))\big)(\|x_k - y\| + \delta). \end{cases}$$

Therefore, if (2) holds for all $k \leq k_0 := h(\varepsilon, d_C) - 1$ then

$$(8)\ \begin{cases} \|x_{k_0+1} - y\| \\ \leq \big(1 - \eta(\varepsilon/(d_C + 1))\big)^{k_0+1}\|x_0 - y\| + \sum\limits_{i=1}^{k_0+1}(1 - \eta(\varepsilon/(d_C + 1)))^i \cdot \delta \\ \leq \big(1 - \eta(\varepsilon/(d_C + 1))\big)^{k_0+1} \cdot d_C + (k_0 + 1)\delta \\ \leq \big(1 - \eta(\varepsilon/(d_C + 1))\big)^{k_0+1} \cdot d_C + \frac{\varepsilon}{12}. \end{cases}$$

We now show that

$$(9)\ \big(1 - \eta(\varepsilon/(d_C + 1))\big)^{k_0+1} \cdot d_C \leq \frac{\varepsilon}{4}.$$

Proof of (9): If $\eta(\varepsilon/(d_C + 1)) = 1$, then the claim holds trivially. Otherwise, (9) is equivalent to

$$k_0 + 1 \geq \frac{\ln(\varepsilon/4d_C)}{\ln(1 - \eta(\varepsilon/(d_C + 1)))}.$$

Since $\ln(1) = 0$ and $\frac{d}{dx}ln(x) = \frac{1}{x} \geq 1$ for all $x \in (0, 1]$, we get

$$-\ln(1 - \eta(\varepsilon/(d_C + 1))) \geq \eta(\varepsilon/(d_C + 1)).$$

Together with $-\ln(\varepsilon/4d_C) = \log(4d_C) - \ln(\varepsilon)$ this yields (9).

(8) and (9) together imply

$$(10)\ \forall k \leq h(\varepsilon, d_C) - 1\big(\|x_k - f(x_k)\| > \varepsilon\big) \to \|x_{h(\varepsilon,d_C)} - y\| \leq \frac{\varepsilon}{3}.$$

Since f is nonexpansive and y is an $\frac{\varepsilon}{3}$-fixed point of f the right-hand side of the implication yields $\|x_{h(\varepsilon,d_C)} - f(x_{h(\varepsilon,d_C)})\| \leq \varepsilon$. So

$$(11)\ \exists k \leq h(\varepsilon, d_C)\big(\|x_k - f(x_k)\| \leq \varepsilon\big)$$

and hence by lemma 2 above

$$(12)\ \forall k \geq h(\varepsilon, d_C)\big(\|x_k - f(x_k)\| \leq \varepsilon\big),$$

which concludes the proof of the theorem. ⊣

Theorem 5 (Analysis of a Modification of Bonsall's [3] Proof of Theorem 2).
Under the same hypotheses as in theorem 4 we obtain

$$\forall x \in C \forall \varepsilon > 0 \forall k \geq h(\varepsilon, d_C)\big(\|x_k - f(x_k)\| \leq \varepsilon\big),$$

where $h(\varepsilon, d_C) := \left\lceil \frac{4 \cdot d_C}{\varepsilon \cdot \eta(\frac{\varepsilon}{d_C+1})} \right\rceil$ *for* $\varepsilon < d_C$ *and* $h(\varepsilon, d_C) := 0$ *otherwise.*
Moreover, if $\eta(\varepsilon)$ *can be written as* $\eta(\varepsilon) = \varepsilon \cdot \tilde{\eta}(\varepsilon)$ *with*

$$(*)\ \forall \varepsilon_1, \varepsilon_2 \in (0,2]\big(\varepsilon_1 \geq \varepsilon_2 \to \tilde{\eta}(\varepsilon_1) \geq \tilde{\eta}(\varepsilon_2)\big),$$

then the bound $h(\varepsilon, d_C)$ *can be replaced (for* $\varepsilon < d_C$*) by*

$$\tilde{h}(\varepsilon, d_C) := \left\lceil \frac{2 \cdot d_C}{\varepsilon \cdot \tilde{\eta}(\frac{\varepsilon}{d_C+1})} \right\rceil.$$

Proof: By lemma 1, f has ε-fixed points $x_\varepsilon \in C$, $\|f(x_\varepsilon) - x_\varepsilon\| < \varepsilon$ for every $\varepsilon > 0$.
Let $\delta > 0$ be such that $\delta < \min(1, \frac{\varepsilon}{3}, \frac{\varepsilon}{12} \cdot \eta(\varepsilon/(d_C+1)))$ and let $y \in C$ be a δ-fixed point of f, i.e.

$$(1)\ \|y - f(y)\| < \delta.$$

Assume that

$$(2)\ \|x_k - y\| \geq \frac{\varepsilon}{3} \text{ and}$$

$$(3)\ \|x_k - f(x_k)\| = \|(x_k - y) - (f(x_k) - y)\| > \varepsilon.$$

As in the proof of theorem 4 one shows that

$$(4)\ \left\| \frac{1}{2} \left(\frac{x_k - y}{\|x_k - y\| + \delta} + \frac{f(x_k) - y}{\|x_k - y\| + \delta} \right) \right\| \leq 1 - \eta(\varepsilon/(d_C+1)).$$

Hence

$$(5)\ \begin{cases} \|x_{k+1} - y\| = \|\frac{1}{2}(x_k + f(x_k)) - y\| = \|\frac{1}{2}(x_k - y + f(x_k) - y)\| \leq \\ \|x_k - y\| + \delta - (\|x_k - y\| + \delta) \cdot \eta(\varepsilon/(d_C+1)) \overset{(2)}{\leq} \\ \|x_k - y\| + \delta - \frac{\varepsilon}{3} \cdot \eta(\varepsilon/(d_C+1)) \leq \|x_k - y\| - \frac{\varepsilon}{4} \cdot \eta(\varepsilon/(d_C+1)). \end{cases}$$

Define

$$n_\varepsilon := \left\lceil \frac{d_C}{\frac{\varepsilon}{4} \cdot \eta(\varepsilon/(d_C+1))} \right\rceil = \left\lceil \frac{4 \cdot d_C}{\varepsilon \cdot \eta(\varepsilon/(d_C+1))} \right\rceil.$$

If (2), (3) both hold for all $k \leq n_\varepsilon$, then (5) yields

$$(6)\ \|x_{n_\varepsilon+1} - y\| < \|x_0 - y\| - d_C,$$

which contradicts the choice of d_C by which $\|x_k - y\| \in [0, d_C]$ for all $k \in \mathbb{N}$. Hence

$$(7)\ \exists k \leq n_\varepsilon\big(\|x_k - y\| \leq \frac{\varepsilon}{3} \vee \|x_k - f(x_k)\| \leq \varepsilon\big).$$

By the choice of δ,(1) and the nonexpansivity of f, the first disjunct also implies that $\|f(x_k) - x_k\| \leq \varepsilon$ and so by the preceding lemma

$$(8)\ \forall k \geq n_\varepsilon\big(\|x_k - f(x_k)\| \leq \varepsilon\big).$$

The last claim in the theorem follows by choosing $y \in C$ as a δ-fixed point of f with $\delta < \min(1, \frac{\varepsilon}{3}, \frac{\varepsilon}{2} \cdot \tilde{\eta}(\varepsilon/(d_C + 1)))$ and the following modifications of (4), (5) to

$$(4)^*\ \left\|\frac{1}{2}\left(\frac{x_k - y}{\|x_k - y\| + \delta} + \frac{f(x_k) - y}{\|x_k - y\| + \delta}\right)\right\| \leq 1 - \eta(\varepsilon/(\|x_k - y\| + \delta)).$$

$$(5)^*\begin{cases} \|x_{k+1} - y\| = \|\frac{1}{2}(x_k + f(x_k)) - y\| = \|\frac{1}{2}(x_k - y + f(x_k) - y)\| \leq \\ \|x_k - y\| + \delta - (\|x_k - y\| + \delta) \cdot \eta(\varepsilon/(\|x_k - y\| + \delta)) = \\ \|x_k - y\| + \delta - \varepsilon \cdot \tilde{\eta}(\varepsilon/(\|x_k - y\| + \delta)) \overset{(*)}{\leq} \|x_k - y\| + \delta - \varepsilon \cdot \tilde{\eta}(\varepsilon/(d_C + 1)) \\ \leq \|x_k - y\| - \frac{\varepsilon}{2} \cdot \tilde{\eta}(\varepsilon/(d_C + 1)) \end{cases}$$

(note that we can apply η to $\varepsilon/(\|x_k - y\| + \delta)$ since (3) and

$$\|f(x_k) - y\| \overset{(1)}{\leq} \|f(x_k) - f(y)\| + \delta \leq \|x_k - y\| + \delta$$

imply

$$\varepsilon \leq \|x_k - y\| + \|f(x_k) - y\| \leq 2(\|x_k - y\| + \delta)$$

and therefore

$$\varepsilon/(\|x_k - y\| + \delta) \in (0, 2]).$$

⊣

If we disregard for a moment the diameter estimate d_C in the bounds in theorems 4 and 5 and put $\varepsilon := 2^{-n}$, then we see that the bound from theorem 4 essentially is $n/\eta(2^{-n})$, whereas the first bound in theorem 5 is only about $2^n/\eta(2^{-n})$. If, however, $\eta(\varepsilon)$ can be written as $\varepsilon \cdot \tilde{\eta}(\varepsilon)$ with $\tilde{\eta}$ satisfying $(*)$, then theorem 5 roughly gives $1/\eta(2^{-n})$ which is better than the bound from theorem 4. It is this fact that we will use in the example below to obtain a polynomial bound for L_p $(p \geq 2)$ which is of degree p.

Examples: It is well-known that the Banach spaces L_p with $1 < p < \infty$ are uniformly convex (this was first proved in [6], see also [20]). For $p \geq 2$, the following explicit modulus η_p of uniform convexity was obtained in [14]

$$\eta_p(\varepsilon) := 1 - (1 - (\varepsilon/2)^p)^{1/p}.$$

One easily shows (using the derivative of $x^{1/p}$) that (for $\varepsilon \in (0,2]$)

$$\eta_p(\varepsilon) \geq \frac{\varepsilon^p}{p2^p}.$$

Hence $\frac{\varepsilon^p}{p2^p}$ is a modulus of convexity as well. Since

$$\frac{\varepsilon^p}{p2^p} = \varepsilon \cdot \tilde{\eta_p}(\varepsilon)$$

with

$$\tilde{\eta_p}(\varepsilon) = \frac{\varepsilon^{p-1}}{p2^p}$$

satisfying $(*)$ in the theorem above, we obtain the following

Corollary 1. *Let $p \geq 2$, $C \subseteq L_p$ a non-empty convex subset with $d(C) \leq d_C \in Q_+^*$, $f : C \to C$ nonexpansive and $(x_k)_{k\in\mathbb{N}}$ defined as in the theorem. Then*

$$\forall x \in C \forall \varepsilon > 0 \forall k \geq \left\lceil \frac{d_C p (d_C+1)^{p-1} 2^{p+1}}{\varepsilon^p} \right\rceil \big(\|x_k - f(x_k)\| \leq \varepsilon\big).$$

Note that the bound in corollary 1 only depends on p, ε and an upper bound d_C of $d(C)$ but not on $x \in C$ or f.

For the case $X := \mathbb{R}, C := [0,1]$, theorem 5 even gives a linear bound, since $\varepsilon/2$ is a modulus of uniform convexity in this case and $\tilde{\eta}(\varepsilon) := \frac{1}{2}$ satisfies $(*)$.

Remark 4. Our result in corollary 1 can easily be improved by replacing (d_C+1) by $(d_C+\delta)$ for any $\delta > 0$ and so in the limit by d_C. In [18], using a direct calculation based on the modulus of uniform convexity for L_p, essentially the same result is obtained (only with a better constant as the factor '$p2^{p+1}$' is missing). For a linear bound in the case $[0,1]$, [18] refers to an unpublished result of J. Alexander. Note, however, that our bounds in these examples were derived just as special instances of the general bound in theorem 5.

4 Effective Uniform Bounds on the Krasnoselski–Mann Iteration in Arbitrary Normed Spaces

In this section we discuss some of the results from [29]. Throughout this section, $(X, \|\cdot\|)$ will be an arbitrary normed linear space, $C \subseteq X$ a non-empty convex subset of X and $f : C \to C$ a nonexpansive mapping.

We consider the so-called Krasnoselski–Mann iteration (which is more general than the Krasnoselski iteration and due to Mann [34]) generated starting from an arbitrary $x \in C$ by

$$x_0 := x, \quad x_{k+1} := (1 - \lambda_k)x_k + \lambda_k f(x_k),$$

where $(\lambda_k)_{k\in\mathbb{N}}$ is a sequence of real numbers in $[0, 1]$. For the background information on this iteration and related references see [4].

Lemma 3 ([4]). *For all $k \in \mathbb{N}$ and $x \in C$:*
$\|x_{k+1} - f(x_{k+1})\| \leq \|x_k - f(x_k)\|$.

For the results in this section we assume (following [4]) that $(\lambda_k)_{k\in\mathbb{N}}$ is divergent in sum, which can be expressed (since $\lambda_k \geq 0$) as

$$(A)\ \forall n, i \in \mathbb{N}\exists k \in \mathbb{N} \left(\sum_{j=i}^{i+k} \lambda_j \geq n \right).$$

We also assume (again as in [4]) that

$$(B)\ \limsup_{k\to\infty} \lambda_k < 1.$$

Define

$$r_C(f) := \inf_{x\in C} \|x - f(x)\|.$$

Theorem 6 ([4]). [9] *Suppose that $(\lambda_k)_{k\in\mathbb{N}}$ satisfies the conditions (A) and (B). Then for any starting point $x \in C$ and the Krasnoselski–Mann iteration (x_n) starting from x we have*

$$\|x_n - f(x_n)\| \overset{n\to\infty}{\to} r_C(f).$$

By lemma 1, the theorem implies

Corollary 2 ([16],[11],[4]). *Under the assumptions of theorem 6 plus the additional assumption that C has bounded diameter $d(C) < \infty$ the following holds:*

$$\|x_n - f(x_n)\| \overset{n\to\infty}{\to} 0.$$

Remark 5. In [11] it is actually shown that one can choose n in the corollary independently of $x \in C$ and f. Whereas in [11] a complicated functional theoretic embedding into the space of all nonexpansive mappings is used to derive

[9] With the additional assumption that λ_k is bounded away from zero, this result is also proved in [35].

this uniformity statement, it trivially follows from our quantitative analysis in corollary 3 below which even provides an explicit effective description of such a uniform n. For a more restricted iteration the existence of a bound n independent of x was also obtained by [8] using, however, also a universal embedding theorem (due to Banach and Mazur). The use of non-trivial functional theoretic arguments in [11] and [8] to obtain the (ineffective) existence of a uniform n clearly indicates that the authors were not aware of explicit effective uniform bounds hidden in the proof of $\lim_{k\to\infty} \|f(x_k) - x_k\| = 0$ as given e.g. in [11] and its generalization in [4].[10]

We will now show how the proof of theorem 6 as given in [4] fits under the general schema of logical proof mining discussed in the introduction. The actual extraction of the bound itself will be carried out in [29].

General Logical Form of the Quantitative Analysis of the Proof of Theorem 6 in [4]:

As we have discussed above, we only can expect to be able to extract a bound $\forall x \exists y \leq \Phi(x) A(x,y)$ from a non-constructive proof if A is a purely existential formula. Since the statement in theorem 6 involves two implicative assumptions on $(\lambda_k)_{k\in\mathbb{N}}$ as well as the existence of $r_C(f)$, it prima facie does not have the required form. However, it can be reformulated so as to have the right logical form by enriching the input $(\lambda_k)_{k\in\mathbb{N}}, f, x, \varepsilon$ by additional data $K \in \mathbb{N}, \alpha : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ and $x^* \in C$.
Let us first examine conditions (A) and (B) on $(\lambda_k)_{k\in\mathbb{N}}$:
An explicit version of (A) asks for a function $\alpha : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ realizing the existential quantifier, i.e.

$$(A_\alpha)\ \forall n, i \in \mathbb{N} \left(\sum_{j=i}^{i+\alpha(i,n)} \lambda_j \geq n \right).$$

(B) states the existence of a $K \in \mathbb{N}$ such that

$$\lambda_k \leq 1 - \frac{1}{K}$$

from some index k_0 on. Since k_0 only contributes an additive constant to our bound we may assume for simplicity that $k_0 = 0$. So let

$$(B_K)\ \forall k \in \mathbb{N}\big(\lambda_k \leq 1 - \frac{1}{K}\big).$$

[10] For a more detailed discussion, see the final section of this paper.

We now formulate the theorem more explicitly as follows:

$$(*)\begin{cases}\forall(\lambda_k)\in[0,1]^{\mathbb{N}}\forall f:C\to C\forall x,x^*\in C\forall K,\alpha\forall\varepsilon>0\exists n\in\mathbb{N}\\ \big(f \text{ nonexpans.}\wedge(A_\alpha)\wedge(B_K)\to\|x_n-f(x_n)\|<\|x^*-f(x^*)\|+\varepsilon\big).\end{cases}$$

Note that by lemma 3, $(*)$ immediately implies theorem 6.
By our representation of real numbers by which $\leq_{\mathbb{R}}\in\Pi^0_1$ and $<_{\mathbb{R}}\in\Sigma^0_1$, the implication

$$\big(f \text{ nonexpansive}\wedge(A_\alpha)\wedge(B_K)\to\|x_n-f(x_n)\|<\|x^*-f(x^*)\|+\varepsilon\big)$$

is equivalent to a purely existential formula. The proof of $(*)$ only uses tools formalizable in **EBA** plus the principle $\mathrm{PCM}(\|x_k-f(x_k)\|)$ (discussed in the introduction) applied to $(\|x_k-f(x_k)\|)_{k\in\mathbb{N}}$ and a complicated inequality due to [11]. This inequality can be treated just as another purely universal implicative premise and does therefore not increase the logical complexity of the theorem (nor does its proof need to be analyzed). Since, furthermore, the Hilbert cube $[0,1]^{\mathbb{N}}$ is a compact space, our general results discussed in the introduction guarantee (at least for complete separable X and definable C)[11] the existence of an effective bound for n which does not depend on (λ_k) directly but which may possibly depend on $K,\alpha,x,x^*,f,\varepsilon$ and γ. This information on what type of result we should look for is a significant application of our logical approach to the specific proof of theorem 6 which would not have been visible without the reformulation of the theorem focusing on its logical form.
We also know a-priori from our general logical meta-theorem, that a uniform bound on n which does not depend on $x,x^*\in C,\gamma>0$ or f is extractable **if** C is compact (and hence has bounded diameter). For the bound actually extracted, the dependence on x,x^*,f,γ can already be eliminated as soon as we have an upper bound on the diameter $d(C)$ of C. This stronger uniformity result is a-postiori information we get for free just by examining the extracted bound.

The extraction itself will be published in another paper [29]. We present here only the results:

Theorem 7 ([29]). *Let $(X,\|\cdot\|)$ be a normed linear space, $C\subseteq X$ a nonempty convex subset and $f:C\to C$ a nonexpansive mapping. Let $(\lambda_k)_{k\in\mathbb{N}}$ be a sequence in $[0,1]$ which is divergent in sum and satisfies*

$$\forall k\in\mathbb{N}\big(\lambda_k\leq 1-\frac{1}{K}\big)$$

[11] The actually extracted bound will in fact turn out to be valid for arbitrary normed linear spaces X and convex subsets $C\subset X$. Note that the convexity assumption on C is purely universal.

for some $K \in \mathbb{N}$.
Let $\alpha : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *be such that*

$$\forall i, n \in \mathbb{N}\big(\alpha(i,n) \le \alpha(i+1,n)\big) \text{ and}$$

$$\forall i, n \in \mathbb{N}\big(n \le \sum_{s=i}^{i+\alpha(i,n)-1} \lambda_s\big).$$

Let $(x_n)_{n\in\mathbb{N}}$ *be the Krasnoselski–Mann iteration*

$$x_{n+1} := (1-\lambda_n)x_n + \lambda_n f(x_n),\ x_0 := x$$

starting from $x \in C$. *Then the following holds*

$$\forall x, x^* \in C \forall \varepsilon > 0 \forall n \ge h(\varepsilon, x, x^*, f, K, \alpha)(\|x_n - f(x_n)\| < \|x^* - f(x^*)\| + \varepsilon),$$

where

$$\begin{array}{l} h(\varepsilon, x, x^*, f, K, \alpha) := \widehat{\alpha}(\lceil 2\|x - f(x)\| \cdot \exp(K(M+1))\rceil \dot{-} 1, M), \\ \text{with } M := \left\lceil \frac{1+2\|x-x^*\|}{\varepsilon} \right\rceil \text{ and} \\ \widehat{\alpha}(0,M) := \tilde{\alpha}(0,M),\ \widehat{\alpha}(m+1,M) := \tilde{\alpha}(\widehat{\alpha}(m,M),M) \text{ with} \\ \tilde{\alpha}(m,M) := m + \alpha(m,M) \quad (m \in \mathbb{N}) \end{array}$$

(*Instead of* M *we may use any upper bound* $\mathbb{N} \ni \tilde{M} \ge \frac{1+2\|x-x^*\|}{\varepsilon}$). *Likewise, we may replace* $\|x - f(x)\|$ *by any upper bound).*

Corollary 3 ([29]).
Under the same assumptions as in theorem 7 plus the assumption that C *has a positive*[12] *bounded diameter* $d(C) < \infty$ *the following holds:*

$$\forall x \in C \forall \varepsilon > 0 \forall n \ge h(\varepsilon, d(C), K, \alpha)\big(\|x_n - f(x_n)\| \le \varepsilon\big),$$

where

$$h(\varepsilon, d(C), K, \alpha) := \widehat{\alpha}(\lceil 2d(C) \cdot \exp(K(M+1))\rceil - 1, M), \text{ with } M := \left\lceil \frac{1+2d(C)}{\varepsilon} \right\rceil$$

and $\widehat{\alpha}$ *as in the previous theorem.*

Remark 6. The behaviour of the bound in corollary 3 w.r.t. $d(C)$ can be improved as follows: if $d(C)$ is different from 1 we renorm the space by the multiplicative factor $\frac{1}{d(C)}$. Then $h(\varepsilon, 1, K, \alpha)$ gives the rate of the asymptotic regularity w.r.t. this new norm and hence $h(\frac{\varepsilon}{d(C)}, 1, K, \alpha)$ for the original norm.

[12] For $d(C) = 0$ things are trivial.

Corollary 4 ([29]). *Let $d, \varepsilon > 0$, $K \in \mathbb{N}$ and $\beta : \mathbb{N} \to \mathbb{N}$ an arbitrary function. Then there exists an $n \in \mathbb{N}$ such that for any normed space X, any non-empty convex set $C \subseteq X$ such that $d(C) \leq d$, any nonexpansive function $f : C \to C$, any sequence $\lambda_k \in [0, 1 - \frac{1}{K}]$ satisfying $n \leq \sum_{s=0}^{\beta(n)} \lambda_s$ (for all $n \in \mathbb{N}$) and any starting point $x_0 \in C$ of the corresponding Krasnoselski–Mann iteration the following holds*

$$\forall m \geq n\big(\|x_m - f(x_m)\| < \varepsilon\big).$$

Corollary 5 ([29]). *Let $(X, \|\cdot\|)$, $C, d(C), f$ be as in corollary 3, $k \in \mathbb{N}, k \geq 2$ and $\lambda_n \in [\frac{1}{k}, 1 - \frac{1}{k}]$ for all $n \in \mathbb{N}$. Consider the Krasnoselski–Mann iteration $x_{n+1} := (1 - \lambda_n)x_n + \lambda_n f(x_n)$ starting from $x_0 := x \in C$. Then the following holds:*

$$\forall x \in C \forall \varepsilon > 0 \forall n \geq g(\varepsilon, d(C))\big(\|x_n - f(x_n)\| \leq \varepsilon\big),$$

where

$$g(\varepsilon, d(C)) := kM \cdot \lceil 2d(C) \exp(k(M+1)) \rceil \text{ with } M := \left\lceil \frac{1 + 2d(C)}{\varepsilon} \right\rceil.$$

5 Evaluation of the Results of the Case Study

We have seen that there are interesting proofs in non-linear functional analysis (and specifically in fixed point theory) which fall under general proof theoretic results on the extractability of uniform bounds we had obtained in previous papers.
We applied these results to essentially two proofs

1) A standard proof from [3] (from the year 1962)[13] of the well-known Krasnoselski fixed point theorem.
2) A proof from [4] (which contains as a special case a proof from [11] from 1982) for a general result on the asymptotic behaviour of the Krasnoselski–Mann iteration in arbitrary normed spaces (generalizing a result from Ishikawa [16]).

Results on 1): Logical analysis of a proof from 1955/62 yielded uniform bounds together with an elementary verification for arbitrary bounded convex sets C. Under slightly less general conditions and with the use of the deep Browder–Göhde–Kirk fixed point theorem our result in theorem 4 was obtained only in 1990 ([18]) (The compact case is due already to Krasnoselski). Moreover, a logical

[13] Krasnoselski's original proof from 1955 is very similar to that as far as we can judge from the Russian text.

modification of the proof using PCM^H_{ar} (with $g(n) = n+1$ as the Herbrand index function in '$\forall g$' of $\mathrm{PCM}^H_{ar}(a_n)$) allowed to improve this bound under a further condition usually satisfied by moduli of uniform convexity (theorem 5). Applying this general bound to L_p ($p \geq 2$) resulted in a polynomial bound of degree p (a result which for this special case was obtained in [18] by an ad hoc calculation). For $X := \mathbb{R}$ and $C := [0,1]$ we even get a linear bound out of our general result (see also [18], p.192).

Results on 2): The logical analysis of the proof in [4] (resp. [11]) carried out in [29] provides the first effective bound for Ishikawa's theorem on the asymptotic behaviour of **general** Krasnoselski–Mann iterations in arbitrary normed spaces X and for bounded sets C (corollary 3). Moreover, our bound is uniform in the sense that it only depends on the error ε and an upper bound d_C of the diameter of C (and some quite general data from the sequence of scalars λ_k used in defining the iteration). I.e. it is independent of the normed space $(X, \|\cdot\|)$. the starting point $x_0 \in C$ of the iteration, the nonexpansive function f and C-data other than d_C. Moreover, it is to a certain extent independent of λ_k. Our result has a long history of partial results: In [8] the ineffective existence of a bound which is independent of x_0 was shown in the special case of constant $\lambda_k = \lambda$. In [11] the non-effective existence of a bound independent of x_0 and f was shown for the case of general λ_k (both [8] and [11] use non-trivial functional theoretic embeddings to obtain these uniformities. Recently, W.A. Kirk ([9]) found an interesting application of this uniformity). In [18] the non-effectivity of all these results is explicitly mentioned and it is stated that 'it seems unlikely that such estimates would be easy to obtain in general setting' (p.191) and therefore in [18] only the special 'tractable' (p.191) classical case of uniformly convex spaces is studied (see the discussion in remark 4 above). Not even the ineffective existence of bounds which, moreover, only depend on C via d_C (corollary 4), was known before and actually in [12] (p.101) conjectured as 'unlikely' to be true (by the same authors whose proof of $\|x_k - f(x_k)\| \to 0$ in [11] does yield such a bound by logical analysis!). Only in the special case of $\lambda_k := \lambda \in (0,1)$ being constant, a uniform (and in fact optimal quadratic) bound was recently obtained using extremely complicated computer aided proofs involving hypergeometric functions (see [1], where once more the non-effectivity of all known proofs of the full Ishikawa result is emphasized). Subsequently, only for the even more special case of $\lambda_k := \frac{1}{2}$ a classically proved result of that type has been obtained (see [2]). This result, of course, is – for that highly special case – numerically better than the exponential bound we obtain in [29] for the much more general case of $\lambda_k \in [\frac{1}{n}, 1 - \frac{1}{n}]$ ($n \geq 2$). The authors stress, however, that their method as

it stands does not apply to non-constant sequences (λ_k). [14] Our bound for the general case of unbounded C treated in [4] (theorem 7) is apparently all new.

Acknowledgment

I am grateful to Professor Jeff Zucker and my Ph.D. student Paulo Oliva for spotting a number of misprints and minor inaccuracies in an earlier version of this paper.

References

1. Baillon, J, Bruck, R.E., The rate of asymptotic regularity is $0(\frac{1}{\sqrt{n}})$. Theory and applications of nonlinear operators of accretive and monotone type, Lecture Notes in Pure and Appl. Math. 178, pp. 51-81, Dekker, New York, 1996.
2. Bruck, R.E., A simple proof that the rate of asymptotic regularity of $(I+T)/2$ is $O(1/\sqrt{n})$. Recent advances on metric fixed point theory (Seville, 1995), pp. 11–18, Ciencias, 48, Univ. Sevilla, Seville, 1996.
3. Bonsall, F.F., Lectures on some fixed point theorems of functional analysis. Tata Institute of Fundamental Research. Bombay 1962.
4. Borwein, J., Reich, S., Shafrir, I., Krasnoselski–Mann iterations in normed spaces. Canad. Math. Bull. **35**, pp. 21-28 (1992).
5. Browder, F.E., Nonexpansive nonlinear operators in a Banach space. Proc. Nat. Acad. Sci. U.S.A. **54**, pp. 1041-1044 (1965).
6. Clarkson, J.A., Uniformly convex spaces. Trans. Amer. Math. Soc. **40**, pp. 396-414 (1936).
7. Deimling, K., Nonlinear Functional Analysis. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, xiv+450 pp., 1985.
8. Edelstein, M., O'Brian, R.C., Nonexpansive mappings, asymptotic regularity and successive approximations. J. London Math. Soc. **17**, pp. 547-554 (1978).
9. Espinola, R., Kirk, W.A., Fixed points and approximated fixed points in product spaces. Preprint.
10. Feferman, S., Kreisel's 'Unwinding Program'. In: P. Odifreddi (ed.), Kreiseliana: about and around Georg Kreisel, A.K. Peters, Wellesley Massachusetts, pp. 247-273 (1996).
11. Goebel, K., Kirk, W.A., Iteration processes for nonexpansive mappings. In: Singh, S.P., Thomeier, S., Watson, B., eds., Topological Methods in Nonlinear Functional Analysis. Contemporary Mathematics **21**, AMS, pp. 115-123 (1983).
12. Goebel, K., Kirk, W.A., Topics in metric fixed point theory. Cambridge studies in advanced mathematics **28**, Cambridge University Press 1990.

[14] Professor Kirk ([19]) communicated to us a new proof of the uniform (w.r.t. x_0 and f) Ishikawa result for the special case $\lambda_k = \lambda$ (again using a functional theoretic embedding). He works in the even more general setting of so-called directionally nonexpansive mappings. It seems interesting to apply a logical analysis to that proof as well. We are grateful to Professor Kirk for bringing the work of Baillon and Bruck to our attention and for communicating to us his recent papers [9],[19].

13. Göhde, D., Zum Prinzip der kontraktiven Abbildung. Math. Nachrichten **30**, pp. 251-258 (1965).
14. Hanner, O., On the uniform convexity of L_p and l_p. Ark. Mat. **3**, pp. 239-244 (1956).
15. Henry, M.S., Schmidt, D., Continuity theorems for the product approximation operator. In: Law, A.G., Sahney, B.N. (eds.), Theory of Approximation with Applications, pp. 24-42, Academic Press, New York (1976).
16. Ishikawa, S., Fixed points and iterations of a nonexpansive mapping in a Banach space. Proc. Amer. Math. Soc. **59**, pp. 65-71 (1976).
17. Kirk, W.A., A fixed point theorem for mappings which do not increase distances. Amer. Math. Monthly **72**, pp. 1004-1006 (1965).
18. Kirk, W.A., Martinez-Yanez, C., Approximate fixed points for nonexpansive mappings in uniformly convex spaces. Annales Polonici Mathematici **51**, pp. 189-193 (1990).
19. Kirk, W.A., Nonexpansive mappings and asymptotic regularity. To appear in: Nonlinear Analysis.
20. Koethe, G., Topoplogische Lineare Räume. Springer-Verlag Berlin-Göttingen-Heidelberg, VI+456 pp., 1960.
21. Kohlenbach, U., Effective moduli from ineffective uniqueness proofs. An unwinding of de La Vallée Poussin's proof for Chebycheff approximation. Ann. Pure Appl. Logic **64**, pp. 27–94 (1993).
22. Kohlenbach, U., New effective moduli of uniqueness and uniform a–priori estimates for constants of strong unicity by logical analysis of known proofs in best approximation theory. Numer. Funct. Anal. and Optimiz. **14**, pp. 581–606 (1993).
23. Kohlenbach, U., Analysing proofs in analysis. In: W. Hodges, M. Hyland, C. Steinhorn, J. Truss, editors, *Logic: from Foundations to Applications. European Logic Colloquium* (Keele, 1993), pp. 225–260, Oxford University Press (1996).
24. Kohlenbach, U., Mathematically strong subsystems of analysis with low rate of growth of provably recursive functionals. Arch. Math. Logic **36**, pp. 31–71 (1996).
25. Kohlenbach, U., Proof theory and computational analysis. Electronic Notes in Theoretical Computer Science **13**, Elsevier (`http://www.elsevier.nl/locate/entcs/volume13.html`), 34 pages (1998).
26. Kohlenbach, U., Elimination of Skolem functions for monotone formulas in analysis. Arch. Math. Logic **37**, pp. 363-390 (1998).
27. Kohlenbach, U., Arithmetizing proofs in analysis. In: Larrazabal, J.M. et al. (eds.), Proceedings Logic Colloquium 96 (San Sebastian), Springer Lecture Notes in Logic **12**, pp. 115–158 (1998).
28. Kohlenbach, U., Things that can and things that can't be done in PRA. Ann. Pure Appl. Logic **102**, pp. 223-245 (2000).
29. Kohlenbach, U., A quantitative version of a theorem due to Borwein-Reich-Shafrir. Submitted.
30. Kohlenbach, U., Oliva, P., Effective bounds on strong unicity in L_1-approximation. Preprint 30pp. (2001).
31. Krasnoselski, M. A., Two remarks on the method of successive approximation. Usp. Math. Nauk (N.S.) **10**, pp. 123-127 (1955) (Russian).

32. Linz, Peter, A critique of numerical analysis. Bull. Amer. Math. Soc. **19**, pp. 407-416 (1988).
33. Luckhardt, H., Herbrand-Analysen zweier Beweise des Satzes von Roth: Polynomiale Anzahlschranken. J. Symbolic Logic **54**, pp. 234-263 (1989).
34. Mann, W.R., Mean value methods in iteration. Proc. Amer. Math. Soc. **4**, pp. 506-510 (1953).
35. Reich, S., Shafrir, I., Nonexpansive iterations in hyperbolic spaces. Nonlinear Analysis, Theory, Methods and Applications **15**, pp. 537-558 (1990).
36. Schwichtenberg, H., Refined Program Extraction from Classical Proofs. To appear in: Proceedings of 1999 Marktobersdorf Summer School.

Formalisation of Computability of Operators and Real-Valued Functionals via Domain Theory*

Margarita V. Korovina[1] and Oleg V. Kudinov[2]

[1] Institute of Informatics Systems
Lavrent'ev pr., 6, Novosibirsk, Russia
rita@ssc.nsu.ru
[2] Institute of Mathematics, Koptug pr., 4, Novosibirsk, Russia
kud@math.nsc.ru

Abstract. Based on an effective theory of continuous domains, notions of computability for operators and real-valued functionals defined on the class of continuous functions are introduced. Definability and semantic characterisation of computable functionals are given. Also we propose a recursion scheme which is a suitable tool for formalisation of complex systems, such as hybrid systems. In this framework the trajectories of continuous parts of hybrid systems can be represented by computable functionals.

1 Introduction

In recent times, new applications of Domain Theory to computation on various spaces have been developed. Domain Theory was independently introduced by Dana Scott [31] as a mathematical theory of computation in the semantics of programming languages and by Yu.L.Ershov [8] as a theory of partial computable functionals of finite types.

A domain is a partially ordered set equipped with the notions of limit and finite approximation; the partial order corresponds to information on the elements.

Given a computation based on an algorithm, each of the sets of input and output forms a domain. The program which carries out the computation is represented as a function between these domains. Every new step in the computation results in an element of the domain of output which provides more information and better approximation to the ultimate result.

A continuous function is one which preserves the information order (so that more input information gives more output information) and the limits of infinite computations in the domain (so that the total information obtainable as output from an infinite sequence of input elements with refining information is the sum of all information obtained from each input element).

* This research was supported in part by the RFBR (grants N 99-01-00485, N 00-01-00810) and by the Siberian Division of RAS (a grant for young researchers, 2000).

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 146–168, 2001.

There are a number of categories of domains in according to various additional properties that they satisfy e.g. [1,18]. In this work, to construct computational models for real-valued functions and functionals we will use continuous domains. Continuous domains, e.g. [31,32,14,6,7,10,38,39,40], are generalisation of algebraic domains, e.g. [2,29,34,35]. The continuous domain (more precisely, the interval domain) for the reals was first proposed by Dana Scott [31] and later was applied to mathematics, physics and real number computation in [6,7,39,40,29] and others.

In this article we propose continuous domains, named as function domains to construct a computational model of operators and real-valued functionals defined on the set of continuous real-valued functions. In Section 2, we recall basic definitions and tools from [7] and introduce some new ones to construct our computational model. We introduce function domains which are effective ω-continuous domains. In general case topological properties of such continuous domain were investigated in [14,11,12]. Based on the notion of computability of mapping between two domains, we propose computability of operators and functionals defined on continuous real-valued functions. The main feature of this approach is related to the fact that continuous operators and functionals defined on continuous real-valued functions can be extended to continuous operators and functionals defined on a function domain.

Then, in Section 3, we give characterisations of computable functions and functionals in logical terms via the definability theory. Thus, computable operators and functionals can be described by finite Σ-formulas.

We also propose a recursion scheme which is a suitable tool for formalisation of complex systems such as hybrid systems. Modelling, design, and investigation of the behaviour of hybrid systems have recently become active areas of research in computer science (for example, see [16,17,22,25,28]).

In the framework proposed in this paper the trajectories of continuous parts of hybrid systems (performance specifications) can be represented by computable functionals. This is a subject of a further paper.

2 Basic Notions

For background material on continuous domain we refer to [1,31,32,14,7]. To propose notions of computability of operators and real-valued functionals, we, following the papers [31,7], recall the definitions of continuous domain for the reals (the interval domain) and computable functions and introduce functional domains.

2.1 Terminology

Throughout the article, we consider the standard model of the real numbers $\langle \mathbb{R}, 0, 1, +, \cdot, -x, \frac{x}{2}, < \rangle$, denoted also by $\mathbb{R}$, where $+$, $\cdot$, $-x$ and $\frac{x}{2}$ are regarded as the usual arithmetic operations on the reals. We use the language of strictly ordered rings, so the predicate $<$ occurs positively in formulas.

Let $\mathbb{R}^-$ denote $\mathbb{R}\cup\{-\infty\}$, $\mathbb{R}^+$ denote $\mathbb{R}\cup\{+\infty\}$, $\mathbb{N}$ denote the set of natural numbers and $\mathbf{D}_2$ denote the set $\{z\cdot 2^{-n} | z\in\mathbb{Z},\ n\in\mathbb{N}\}$.

Let us use $\bar{r}$ to denote $r_1,\ldots,r_m$. By $\mathrm{dom}(f)$ we denote the domain of a function f. We use the standard notations for real intervals (a,b), $[a,b]$, $(a,b]$, $[a,b)$. In addition, $\langle a,b\rangle$ denotes any of those intervals. We use the standard denotation $C([a,b])$ for the set of continuous real-valued functions defined on a compact interval $[a,b]$.

2.2 The Effective Interval Domain for the Reals

By *the interval domain for the reals* we mean the set of compact intervals of $\mathbb{R}$, partially ordered with reversed subset inclusion and endowed with the least element.

We recall the definition of the *interval domain* $\mathcal{I}$ proposed in [7]:
$\mathcal{I} = \{[a,b]\subseteq\mathbb{R} \mid a,b\in\mathbb{R},\ a\le b\}\cup\{\bot\}$.
The order is reversed subset inclusion, i.e. $\bot\sqsubseteq I$ for all $I\in\mathcal{I}$ and $[a,b]\sqsubseteq[c,d]$ iff $a\le c$ and $d\le b$ in the usual ordering of the reals. One can consider the least element $\bot$ as the set $\mathbb{R}$. Directed suprema are filtered intersections of intervals. The way-below relation is given by $I\ll J$ iff $J\subseteq\mathrm{int}(I)$, where $\mathrm{int}(I)$ denotes the interior of I. For the relation $\ll$ we have the following properties: $\bot\ll J$ for all $J\in\mathcal{I}$ and $[a,b]\ll[c,d]$ if and only if $a<c$ and $b>d$. Note that $\mathcal{I}$ is an effective ω-continuous domain. A countable basis $\mathcal{I}_0$ is given by the collection of all intervals with rational endpoints together with the least element $\bot$. Similarly, we can define the interval domain $\mathcal{I}_{[a,b]}$ for an interval $[a,b]$.

The maximal elements are the intervals $[a,a]$ denoted as $\{a\}$. We denote the set of maximal elements as $\max(\mathcal{I})$. It is easy to see that the maximal elements with the subspace topology of Scott topology on $\mathcal{I}$ is homeomorphic to the real line with the standard topology. That is why we can identify a real number r with $\{r\}$.

Definition 1. *Let $\mathcal{I}_0=\{b_0,\ldots,b_n,\ldots\}$ be an effective enumerated set of all intervals with rational endpoints endowed with the least element $\bot$.*
A continuous function $f:\mathcal{I}\to\mathcal{I}$ is computable, if the relation $b_m\ll f(b_n)$ is computably enumerable in n, m, where $b_m,\ b_n\in\mathcal{I}_0$.

Definition 2. *A function $f:\mathbb{R}\to\mathbb{R}$ is computable if and only if there is an continuous enlargement $g:\mathcal{I}\to\mathcal{I}$ (i.e., $\mathrm{dom}(f)=\{x | g(\{x\})\in\max(\mathcal{I})\}$ and $g(\{x\})=\{f(x)\}$ for all $x\in\mathrm{dom}(f)$) which is computable in the sense of Definition 1.*

2.3 The Effective Function Domain

In this section we introduce effective function domains which are effective ω-continuous domains. Based on the notion of computability of mapping between two domains we propose computability of operators and functionals defined on $C([a,b])$. The main feature of this approach is related to the fact that continuous operators and functionals defined on continuous real-valued functions

can be extended to continuous operators and functionals defined on the corresponding function domain. Moreover, we propose a semantic characterisation of computable operators and functionals via validity of finite Σ-formulas.

Let $\mathcal{I}$ be equipped with Scott topology (for the definition we refer to [31,32,14]). We consider the set of continuous functions $f : [a, b] \to \mathcal{I}$ defined on a compact interval $[a, b]$ with computable endpoints.

Remark 1. A function $f : [a, b] \to \mathcal{I}$ is continuous in x_0 iff $f(x_0) = \bot$ or $f(x_0) = [c, d]$ and $\forall \epsilon_1 \epsilon_2 \exists \delta \left(\forall x \in [a, b]\right) (|x - x_0| < \delta \to [c - \epsilon_1, d + \epsilon_2] \ll f(x))$.

A function is continuous on $[a, b]$ if it is continuous in every point of $[a, b]$.

Definition 3. *A function $f : [a, b] \to \mathbb{R}^-$ is said to be lower semicontinuous if the set $Y_f^- = \{x | f(x) \neq -\infty\}$ is open w.r.t the standard topology and*

$$\left(\forall x_0 \in Y_f^-\right) (\forall y < f(x_0)) \, \exists \delta \left(|x_0 - x| < \delta \to y < f(x)\right).$$

A function $f : [a, b] \to \mathbb{R}^+$ is said to be upper semicontinuous if the set $Y_f^+ = \{x | f(x) \neq +\infty\}$ is open w.r.t the standard topology and

$$\left(\forall x_0 \in Y_f^+\right) (\forall y > f(x_0)) \, \exists \delta \left(|x_0 - x| < \delta \to y > f(x)\right).$$

For the classical theory of semicontinuous functions the reader should is referred to a standard textbook (e.g. [4]). The reader can also find some properties of computability on continuous and semicontinuous real functions in [41]. It is easy to see that a continuous function $f : [a, b] \to \mathcal{I}$ is closely related to the pair of functions $\left\langle f^1 : [a, b] \to \mathbb{R}^-, f^2 : [a, b] \to \mathbb{R}^+ \right\rangle$, where $f^1(x) = \inf \ f(x)$ is lower semicontinuous and $f^2(x) = \sup \ f(x)$ is upper semicontinuous (see [10]). The function f^1 is called a *lower bound* of f and f^2 is called a *upper bound* of f. Below we denote $Y_f = \{x | f(x) \neq \bot\}$ for $f : [a, b] \to \mathcal{I}$.

To introduce our notions of computable operators and real-valued functionals, we introduce functional domains which are effective ω-continuous domains.

Definition 4. *Let a, b be computable real numbers. A function domain $\mathcal{I}_f([a, b])$ is the collection of all continuous functions $f : [a, b] \to \mathcal{I}$ with the least element $\bot_{[a,b]}$ partially ordered by the following relation: $f \sqsubseteq g$ iff $(\forall x \in [a, b]) \, (f(x) \sqsubseteq g(x))$ and $\bot_{[a,b]} \sqsubseteq I$ for all $I \in \mathcal{I}_f([a, b])$.*

The way-below relation $\ll$ is induced by $\sqsubseteq$ in the standard way.

Proposition 1. *For each compact interval $[a, b]$ with computable endpoints the function domain $\mathcal{I}_f([a, b])$ is an effective ω-continuous domain.*

Proof. The existence of $\bigvee^{\uparrow} A$ for each directed subset $A \subseteq \mathcal{I}_f([a, b])$ follows from the properties of semicontinuous functions. Indeed, $\bigvee^{\uparrow} A =< \sup_{f \in A} f^1, \inf_{f \in A} f^2 >$, where f^1 is the lower bound and f^2 is the upper bound of f.

Let us prove that $f = \vee^{\uparrow}\left(\downarrow f\right)$ for $f \in \mathcal{I}_f([a,b])$, where $\downarrow f$ denotes the set $\{g \in \mathcal{I}_f([a,b]) | g \ll f\}$. Let U be open and $\mathrm{cl}U = \bar{U} \subset Y_f$. The set $\downarrow f$ contains all functions of the type $g^n_U = \langle a^n_U, c^n_U \rangle$, where

$$a^n_U(x) = \begin{cases} -\infty & \text{if } x \not\in U, \\ \inf_{z \in \bar{U}} f^1(z) - \frac{1}{n} & \text{if } x \in U, \end{cases}$$

$$c^n_U(x) = \begin{cases} +\infty & \text{if } x \not\in U, \\ \sup_{z \in \bar{U}} f^2(z) + \frac{1}{n} & \text{if } x \in U, \end{cases}$$

By the properties of semicontinuous functions, $\vee^{\uparrow}\{g^n_U | \bar{U} \subset Y_f,\ n \in \omega\} = f$, so $\vee^{\uparrow}\downarrow f = f$.

It is obvious that the function domain $\mathcal{I}_f([a,b])$ is ω-continuous. An example for a countable basis is the set $\mathcal{I}_{f,0}([a,b]) = \{b_n\}_{n \in \omega} \cup \{\bot_{[a,b]}\}$, where the lower bound b^1_n and the upper bound b^2_n of b_n satisfy the following conditions: there exist $a = a_0 \ldots \leq a_i \leq \ldots \leq a_n = b$ such that

1. for all $x \in (a_i, a_{i+1})$ $b^1_n(x) = -\infty$ and $b^2_n(x) = +\infty$ or $b^1_n(x) = \alpha_i x + \beta_i$ and $b^2_n(x) = \gamma_i x + \zeta_i$;
2. if for $x \in (a_i, a_{i+1}) \cup (a_{i+1}, a_{i+2})$ b^1_n and b^2_n are finite then $b^1_n(a_{i+1}) = \alpha_i a_{i+1} + \beta_i = \alpha_{i+1} a_{i+1} + \beta_{i+1}$ and $b^2_n(a_{i+1}) = \gamma_i a_{i+1} + \zeta_i = \gamma_{i+1} a_{i+1} + \zeta_{i+1}$;
3. if b^1_n and b^2_n are infinite on (a_i, a_{i+1}) then $b^1_n(a_i) = b^1_n(a_{i+1}) = -\infty$ and $b^2_n(a_i) = b^2_n(a_{i+1}) = +\infty$, where $a_i,\ \alpha_i,\ \beta_i,\ \gamma_i,\ \zeta_i \in \mathrm{D}_2$.

Using the standard numbering of the set of piecewise linear functions with coefficients from D_2, it is easy to prove that $\mathcal{I}_{f,0}([a,b])$ is countable and effective. □

In the same way we can construct an functional domain $\mathcal{I}_f([a,b]^n)$ for $n \in \omega$.

Corollary 1. *For each compact n-cube $[a,b]^n$, with computable a and b, the functional domain $\mathcal{I}_f([a,b]^n)$ is an effective ω-continuous domain.*

Proof. It is similar to the proof of Proposition 1. □

Now we consider a useful property of the way-below relation '$\ll$'. Thus, $f \ll g$ if and only if these functions are separated. Also we refer to [32,14,11] for similar characterisations of the way-below relation on spaces of continuous functions from topological spaces into continuous posets.

Definition 5. *Let f and g be lower semicontinuous functions, $\mathrm{cl}Y^-_f \subset Y^-_g$ and $f \leq g$. The functions f, g are said to be separated if there exists a continuous on Y^-_g function h such that $f(x) \leq h(x) < g(x)$ for all $x \in Y^-_g$.*

Let f and g be upper semicontinuous functions, $\mathrm{cl}Y^+_g \subset Y^+_f$ and $f \leq g$. The functions f and g are said to be separated if there exists a continuous on Y^+_f function h such that $f(x) < h(x) \leq g(x)$ for all $x \in Y^+_f$.

Let $f : \mathbb{R} \to \mathcal{I}$ and $g : \mathbb{R} \to \mathcal{I}$ be continuous. The functions f and g are said to be separated if $\mathrm{cl}Y^-_{f^1} \subset Y^-_{g^1}$, $\mathrm{cl}Y^+_{f^2} \subset Y^+_{g^2}$ and their lower bounds $f^1,\ g^1$ and their upper bounds $f^2,\ g^2$ are separated.

Proposition 2. *Let f and g be lower semicontinuous and $\mathrm{cl}Y_f^- \subset Y_g^-$, $f \leq g$. The following assertions are equivalent.*

1. *f and g are separated;*
2. *there exists an upper semicontinuous step function h such that*

   $f(x) \leq h(x) < g(x)$ for $x \in Y_g^-$;
3. *there exists an upper semicontinuous function h such that*
   $f(x) \leq h(x) < g(x)$ for $x \in Y_g^-$;
4. *$(\forall x \in Y_g^-)\exists U_x(\exists t_x > 0)(\forall z, w \in U_x)(g(z) > f(w) + t_x)$, where U_x denotes some neighbourhood of x.*

Proof. We prove nontrivial passages.
$1 \to 2$. It follows from the fact that each continuous function is approximated by an upper semicontinuous step function (see [4]).
$2 \to 1$. See [36].
$2 \to 3$. Obviously.
$3 \to 4$. Let $h : [a, b] \to \mathbb{R}^+$ be upper semicontinuous and $f(x) \leq h(x) < g(x)$ for $x \in Y_g^-$. For $x \in Y_g^-$ put $\tau = g(x) - h(x)$ and $\epsilon = \frac{t}{3}$. According to upper semicontinuity of h and lower semicontinuity of g, there exists a neighbourhood U_x of x such that for all $z,\ w \in U_x : f(z) \leq h(z) < h(x) + \epsilon$ and $g(w) > g(x) - \epsilon$. We have $g(w) > g(x) - \epsilon = h(x) + \frac{2}{3}\tau > h(z) + \frac{\tau}{3} \geq f(z)$. For $t_x = \frac{\tau}{3}$ assertion 4 holds.
$4 \to 2$. Let $\{U_x\}_{x \in \mathrm{cl}Y_f^-}$ have the following property: for all $z, w \in U_x$ $g(z) > f(w) + t_x$. Since $\mathrm{cl}Y_f^-$ is compact, we can construct a finite set $\{\bar{U}_{x_i}\}_{i \leq n}$ such that:

1. $\bar{U}_{x_i}$ is closed;
2. $\bar{U}_{x_i} \cap \bar{U}_{x_j}$ is one-element or empty;
3. $Y_f^- \subseteq \bigcup_{i \leq n} \bar{U}_{x_i}$.

Put $h(x) = \sup\{y | y \geq f(x) \wedge (\exists i(x \in \bar{U}_{x_i}) \wedge (y \leq \inf_{z \in \bar{U}_{x_i} \cap \mathrm{cl}Y_f^-} g(z) - t_{x_i})\}$.

By the properties of lower semicontinuity of g, the function h is a required one. □

Proposition 3. *Let f and g be upper semicontinuous and $\mathrm{cl}Y_g^+ \subset Y_f^+$, $f \leq g$. The following assertions are equivalent.*

1. *f and g are separated;*
2. *there exists a lower semicontinuous step function h such that*

   $f(x) < h(x) \leq g(x)$ for $x \in Y_f^+$;
3. *there exists a lower semicontinuous function h such that*
   $f(x) < h(x) \leq g(x)$ for $x \in Y_f^+$;
4. *$(\forall x \in Y_f^+)\exists U_x(\exists t_x > 0)(\forall z, w \in U_x)(g(z) > f(w) + t_x)$, where U_x denotes some neighbourhood of x.*

Proof. It is similar to the proof of Proposition 2. □

Lemma 1. *Let A be a directed set of lower semicontinuous functions and $\lim_{a\in A} a(x) = g(x)$. For a compact V and every $c \in \mathbb{R}$ the following assertion holds. If $g(x) > c$ for all $x \in V$, then there exists $a \in A$ such that $a(x) > c$ for all $x \in V$.*

Proof. Clearly, for all $x \in V$ there exists $a_x \in A$ such that $a_x(x) > c$. By the definition of lower semicontinuity, there exists a neighbourhood U_x of x with $(\forall z \in U_x)\,(a_x(z) > c)$. The set $\{U_x\}_{x\in V}$ covers the compact V, so we can extract a finite subcovering $\{U_{x_i}\}_{i\leq n}$. For all $z \in U_{x_i}$ we have $a_{x_i}(z) > c$. By the definition of directed set, there exists a function $a \in A$ such that $a(x) > a_{x_i}(x)$ for all $x \in V$. This is a required function. □

Theorem 1. *Continuous functions $f : [a, b] \to \mathcal{I}$ and $g : [a, b] \to \mathcal{I}$ are separated if and only if $f \ll g$.*

Proof. Let f and g be separated and f^1, g^1 be their lower bounds. We show that for a directed set $A \subseteq \mathcal{I}_f[a, b]$ with $g \sqsubseteq \vee^{\uparrow} A$ there exists $a \in A$ such that $f \sqsubseteq a$. It is sufficient to prove that there exists a with the lower bound a^1 such that $a^1(x) \geq f^1(x)$ for $x \in [a, b]$. By Proposition 2 we have $\mathrm{cl}Y^-_{f^1} \subseteq Y^-_{g^1} \wedge (\forall x \in Y^-_{g^1})\exists U_x(\exists t_x > 0)(\forall z, w \in U_x)(g^1(z) > f^1(w) + t_x)$, where U_x denotes some neighbourhood of x.

From the set $\{U_x\}_{x\in Y^-_{g^1}}$ which covers $\mathrm{cl}\,Y^-_{f^1}$ we can extract a finite set $\{U_{x_i}\}_{i\leq n}$ such that $\mathrm{cl}\,Y^-_{f^1} \subseteq \{U_{x_i}\}_{i\leq n}$. Moreover, it is easy to construct $\{\bar{U}_{x_i}\}_{i\leq m}$ which covers $\mathrm{cl}Y^-_{f^1}$, where $\bar{U}_{x_i}$ is compact. For $i \leq m$ we define $c_i = \sup\{y | y \leq (\inf_{x\in\bar{U}_{x_i}} g^1(x) - t_{x_i})\}$.

Clearly, $g^1(x) > c_i \geq f^1(x)$ for all $x \in \bar{U}_{x_i}$. From Lemma 1 we have that there exists a_i with the lower bound a^1_i such that $a^1_i(x) > c_i \geq f^1(x)$ for all $x \in \bar{U}_{x_i}$. Since A is directed, there exists $a \sqsupseteq a_i$ for all $i \leq m$. This function is a required one.

Let $f \ll g$. We show that

1. $\mathrm{cl}Y^-_{f^1} \subseteq Y^-_{g^1}$ and
2. $(\forall x \in Y^-_{g^1})\exists U_x(\exists t_x > 0)(\forall z, w \in U_x)(g^1(z) > f^1(w) + t_x)$,

where U_x denotes a neighbourhood of x and f^1, g^1 denote the lower bounds of f and g. For the upper bounds the corresponding assertion is proved by analogy. Obviously, $Y_f \subseteq Y_g$ and $f \sqsubseteq g$. Let us prove that $\mathrm{cl}\,Y^-_f \subseteq Y^-_g$. Suppose the contrary: there exists a sequence $\{x_n\}_{n\in\omega}$ such that for all n $x_n \in Y_f \subseteq Y_g$, but $\lim\limits_{x\to\infty} x_n = x \notin Y_g$, i.e. $g(x) = \bot$. We can extract a subsequence $\{x_{m_n}\}_{n\in\omega}$ such that $|x_{m_n} - x| < \frac{1}{n}$. We define a sequence of lower semicontinuous functions in the following way:

$$a^1(x) = \begin{cases} g^1(y) & \text{if } |y - x| > \frac{1}{n}, \\ -\infty & \text{if } |y - x| \leq \frac{1}{n}. \end{cases}$$

On the one hand $\lim\limits_{n\to\infty} a^1_n(y) = g^1(y)$ for all $y \in [a, b]$ and on the other hand $-\infty = a^1_n(x_{m_n}) < f^1(x_{m_n}) \neq -\infty$. This is a contradiction with the assumption $f \ll g$.

Let us prove $(\forall x \in Y^-_{g^1})\exists U_x(\exists t_x > 0)(\forall z, w \in U_x)(g^1(z) > f^1(w) + t_x)$. Suppose contrary: there exists $x \in Y_g$ such that $\forall U_x \forall t_x\, (\exists z, w \in U_x)\, g^1(z) < f^1(w) + t_x$. Let us define $\{U_n\}_{n\in\omega}$ by the following rule:

$$U_n = (x - \frac{1}{n}, x + \frac{1}{n}) \text{ if } x \in (a, b),$$
$$U_n = [a, a + \frac{1}{n}) \text{ if } x = a,$$
$$U_n = (b - \frac{1}{n}, b] \text{ if } x = b.$$

There exists n_0 such that for all $n \geq n_0$ we have $U_n \subseteq Y^-_{g^1}$ and there exists $w_n \in U_n$ with $\inf_{z\in\bar{U}_n} g^1(z) < f^1(w_n) + \frac{1}{n}$. We construct an increasing sequence of lower semicontinuous functions such that the limit of this sequence is g^1, but there is no n such that $a_n(y) \geq f^1(y)$ for all $y \in [a, b]$.

Put

$$a^1_n(y) = \begin{cases} g^1(y) & \text{if } y \notin U_n, \\ \inf_{z\in\bar{U}_n} g^1(z) - \frac{1}{n} & \text{if } y \in U_n, \end{cases}$$

where $\bar{U}_n$ is closure of U_n. It's easy to see that $\lim\limits_{n\to\infty} a^1_n(y) = g^1(y)$ for all $y \in [a, b]$. For $y \neq x$ it is obvious. We consider the nontrivial case when $y = x$. Suppose the contrary: there exists c such that $a^1_n(x) < c < g^1(x)$. By lower semicontinuity of g, there exists N with $\inf_{z\in\bar{U}_N} g^1(z) > c + \frac{1}{N}$. So $\inf_{z\in\bar{U}_N} g^1(z) = a^1_N(x) > c$. This is a contradiction.

On the one hand, $\lim\limits_{n\to\infty} a^1_n(y) = g^1(y)$ for all $y \in [a, b]$ and, on the other hand, there is no n such that $a_n(y) \geq f^1(x)$ for all $y \in [a, b]$ because for all n $a^1_n(w_n) < f^1(w_n)$. This is a contradiction with the assumption $f \ll g$.

□

Now we introduce notions of computable operators and computable functionals defined on total continuous real-valued functions. Below we use the standard notion of continuity of a total operator $F : \mathcal{I}_f([a, b]) \to \mathcal{I}_f([c, d])$ w.r.t. the Scott topologies on $\mathcal{I}_f([a, b])$ and $\mathcal{I}_f([c, d])$.

Definition 6. *Let $\mathcal{I}_f([a, b])$, $\mathcal{I}_f([c, d])$ be some function domains and $\mathcal{I}_{f,0}([a, b]) = \{b_i\}_{i\in\omega}$, $\mathcal{I}_{f,0}([c, d]) = \{c_i\}_{i\in\omega}$ be their effective bases constructed as in Proposition 1. A continuous total operator $F : \mathcal{I}_f([a, b]) \to \mathcal{I}_f([c, d])$ is computable, if the relation $c_m \ll F(b_n)$ is computably enumerable in n and m, where $b_n \in \mathcal{I}_{f,0}([a, b])$ and $c_m \in \mathcal{I}_{f,0}([c, d])$.*

Definition 7. *A partial operator $F : C[a, b] \to C[c, d]$ is computable, if* dom (F) *is open and there exists a computable operator $F^* : \mathcal{I}_f([a, b]) \to \mathcal{I}_f([c, d])$ such that*

$$F(f) = g \leftrightarrow F^*(\hat{f}) = \hat{g}, \text{ where } \hat{f}(x) = \{f(x)\}, \ \hat{g}(x) = \{g(x)\}.$$

Definition 8. *A partial functional $F : C[a, b] \times [c, d] \to \mathbb{R}$ is computable, if there exists a computable operator $F^* : C[a, b] \to C[c, d])$ such that*

$$F(f, x) = y \leftrightarrow F^*(f)(x) = y.$$

Proposition 4. *Computable operators and functionals defined on continuous real-valued functions are continuous w.r.t. the standard topology induced by the uniform norm.*

Proof. It follows from the definition and continuity of corresponding operator $F^* : \mathcal{I}_f([a,b]) \to \mathcal{I}_f([c,d])$ and the fact that subspace Scott topologies on $C([a,b])$ and $C([c,d])$ coincide with the standard topologies (see [32,14]). □

To introduce computability of a functional of the type $F : C[a,b] \times \mathbb{IR} \to \mathbb{IR}$, we use an effective sequence of domains $\{\mathcal{I}_f([-n,n])\}_{n\in\omega}$ with coordinated bases in the following sense. We consider a sequence of bases $\{\mathcal{I}_{f,0}([-n,n])\}_{n\in\omega} = \{\{b_i^n\}_{i\in\omega}\}_{n\in\omega}$ with the homomorphisms $\mathrm{res}_{m,n} : \mathcal{I}_f([-m,m]) \to \mathcal{I}_f([-n,n])$ of restrictions for $m > n$ defined by the natural rules $\mathrm{res}_{m,n}(b_i^m) = b_i^m|_{[-n,n]} = b_i^n$ and $\mathrm{res}_{m,n}(\bot_{[-m,m]}) = \bot_{[-n,n]}$.

Definition 9. *A sequence $\{F_k\}_{k\in\omega}$ of computable operators $F_k : \mathcal{I}_f[a,b] \to \mathcal{I}_f[-k,k]$ is uniformly computable, if $b_m^k \ll F_k(b_n^k)$ is computably enumerable in k, n and m.*

Definition 10. *A sequence $\{F_k\}_{k\in\omega}$ of computable operators $F_k : C[a,b] \to C[-k,k]$ is uniformly computable, if there exists a uniformly computable sequence $\{F_k^*\}_{k\in\omega}$ of computable operators $F_k : \mathcal{I}_f[a,b] \to \mathcal{I}_f[-k,k]$ such that*

$$F_k(f) = g \leftrightarrow F_k^*(\hat{f}) = \hat{g}, \text{ where } \hat{f}(x) = \{f(x)\},\ \hat{g}(x) = \{g(x)\},\ k \in \omega,\ x \in [a,b].$$

Definition 11. *A functional $F : C[a,b] \times \mathbb{IR} \to \mathbb{IR}$ is computable, if there exists a uniformly computable sequence $\{F_k^*\}_{k\in\omega}$ of computable operators $F_k^* : C[a,b] \to C[-k,k]$ such that*

$$F(f,x) = y \leftrightarrow \forall k\,(x \in [-k,k] \to F_k^*(f)(x) = y)\,.$$

Note that for $m > n$ the condition $\mathrm{res}_{m,n}(F_m^*(f)) = F_n^*(f)$ holds by construction.

Proposition 5. *A computable functional $F : C[a,b] \times \mathbb{IR} \to \mathbb{IR}$ is continuous w.r.t. the standard topology induced by the uniform norm.*

Proof. It follows from the definition and continuity of the corresponding operators $F_k^* : \mathcal{I}_f([a,b]) \to \mathcal{I}_f([-k,k])$ for $k \in \omega$. □

In the same way we can define computability of functionals $F : C[a,b] \times \mathbb{IR}^n \to \mathbb{IR}$.

Corollary 2. *A computable functional $F : C[a,b] \times \mathbb{IR}^n \to \mathbb{IR}$ is continuous.*

Proof. It follows from the definition and continuity of the corresponding operators $F_k^* : \mathcal{I}_f([a,b]) \to \mathcal{I}_f([-k,k]^n)$ for $k \in \omega$. □

Also we point attention to the following property. Every continuous total operator $F : C[a,b] \to C[a,b]$ has a continuous extension to the functional domain. This means there is a continuous operator $F^* : \mathcal{I}_f([a,b]) \to \mathcal{I}_f([a,b])$ such that

$$F(f) = g \leftrightarrow F^*(\hat{f}) = \hat{g}, \text{ where } \hat{f}(x) = \{f(x)\},\ \hat{g}(x) = \{g(x)\}.$$

To prove this fact, we will use the following notion.

Definition 12. *Let f be a lower semicontinuous function defined on $[a,b]$ and g be a upper continuous function defined on $[a,b]$. A sequence $\{h_s\}_{s\in\omega}$ of continuous functions defined on $[a,b]$ is said to be closely approximating to $\langle f,g\rangle \in I_f([a,b])$ if*

$$\forall\varepsilon > 0 \exists N \forall n \geq N \left(h_n \in \langle f-\varepsilon, g+\varepsilon\rangle\right).$$

Theorem 2. *Every continuous total operator $F : C[a,b] \to C[c,d]$ has a continuous extension to the functional domains.*

Proof. It is enough to define operator $F^* : \mathcal{I}^0_f([a,b]) \to \mathcal{I}_f([a,b])$, where $\mathcal{I}^0_f([a,b])$ denotes the set $\{h \in \mathcal{I}_f([a,b]) | h : [a,b] \to \mathcal{I} \setminus \bot\}$ which is an ω-continuous domain. Indeed, the operator F^* can be extended to $F^{**} : I_f([a,b]) \to I_f([a,b])$ by the rule:

$$F^{**}(h) = \begin{cases} F^*(h) & \text{if } h \in I^0_f([a,b]), \\ \bot_{[c,d]} & otherwise. \end{cases}$$

Note that the set $\mathcal{I}^0_{f,0}([a,b]) = \{\langle f,g\rangle \,|f,g \in C[a,b]\}$ can be considered as a basis for $\mathcal{I}^0_f([a,b])$.

Let us denote $U^-_{F(f)} = \{(x,t)|F(f)(x) > t\}$ and $U^+_{F(f)} = \{(x,t)|F(f)(x) < t\}$ for a continuous function f.

First we define auxiliary operators $\mathcal{F}$ defined on the set $\mathcal{I}^0_{f,0}([a,b])$ of stripes with continuous bounds, and then extend it to an operator defined on $\mathcal{I}^0_f([a,b])$.

For $\left\langle f^1, f^2\right\rangle \in \mathcal{I}^0_{f,0}([a,b])$, where f^1 and f^2 are continuous, we define two open sets $U^- < U^+$ by the following rules.
We define $(x,t) \in U^-$ if and only if there exists $\varepsilon > 0$ such that for each sequence $\{h_n\}_{n\in\omega}$ which is closely approximating to $\left\langle f^1, f^2\right\rangle$ we have:

$$\exists N \left(\forall n \geq N\right) B((x,t),\varepsilon) \subset U^-_{F(h_n)},$$

where $B((x,t),\varepsilon)$ is the ball of the radius ε centred at (x,t).
By analogy, $(x,t) \in U^+$ if and only if there exists $\varepsilon > 0$ such that for each sequence $\{h_n\}_{n\in\omega}$ which is closely approximating to $\left\langle f^1, f^2\right\rangle$ we have:

$$\exists N \left(\forall n \geq N\right) B((x,t),\varepsilon) \subset U^+_{F(h_n)},$$

where $B((x,t),\varepsilon)$ is the ball of the radius ε centred at (x,t).

Let us define $g^1(x) = \sup U^-(x)$ and $g^2(x) = \inf U^+(x)$.

Put $\mathcal{F}(\left\langle f^1, f^2\right\rangle) = \left\langle g^1, g^2\right\rangle$. Since g^1 is lower semicontinuous and g^2 is upper semicontinuous, the operator $\mathcal{F}$ is well-defined. For $\mathcal{F}$ we denote U^- as $U^-_{\mathcal{F}(\langle f^1,f^2\rangle)}$ and U^+ as $U^+_{\mathcal{F}(\langle f^1,f^2\rangle)}$.

We show that $\mathcal{F}(\langle f, f\rangle) = \langle F(f), F(f)\rangle$. Indeed, a sequence $\{h_n\}_{n\in\omega}$, which is closely approximate to $\langle f, f\rangle$, uniformly converges to f. By continuity of the operator F, the sequence $\{F(h_n)\}_{n\in\omega}$ uniformly converges to $F(f)$. So $U^-_{\mathcal{F}(\langle f,f\rangle)} = U^-_{F(f)}$ and $U^+_{\mathcal{F}(\langle f,f\rangle)} = U^+_{F(f)}$.

Monotonicity of the operator $\mathcal{F}$ follows from the definitions of $U^-_{\mathcal{F}(\langle f^1,f^2\rangle)}$ and $U^+_{\mathcal{F}(\langle f^1,f^2\rangle)}$. Let $A = \{\langle u^1_n, u^2_n\rangle\}_{n\in\omega}$ be a monotonic directed set and $\bigvee^{\uparrow} A =< u^1, u^2 >$. We check that if u^1_n, u^2_n, u^1 and u^2 are continuous then $\bigvee^{\uparrow} \mathcal{F}(\langle u^1_n, u^2_n\rangle) = \mathcal{F}(\langle u^1, u^2\rangle)$. By monotonicity of $\mathcal{F}$, $\mathcal{F}(\langle u^1, u^2\rangle) \sqsupseteq \mathcal{F}(\langle u^1_n, u^2_n\rangle)$ for all $n \in \omega$. Hence $\mathcal{F}(\langle u^1, u^2\rangle) \sqsupseteq \bigvee^{\uparrow} \mathcal{F}(\langle u^1_n, u^2_n\rangle)$. To prove the inclusion $\bigvee^{\uparrow} \mathcal{F}(\langle u^1_n, u^2_n\rangle) \sqsupseteq \mathcal{F}(\langle u^1, u^2\rangle)$, it is enough to check that for (x, t) such that $(x, t) \in U^-_{\mathcal{F}(\langle u^1,u^2\rangle)}$ there exists $n \in \omega$ with $(x, t) \in U^-_{\mathcal{F}(\langle u^1_n,u^2_n\rangle)}$. Suppose the contrary. For some $(x, t) \in U^-_{\mathcal{F}(\langle u^1,u^2\rangle)}$, and for all $n \in \omega$ we have $(x, t) \notin U^-_{\mathcal{F}(\langle u^1_n,u^2_n\rangle)}$. Let us find $\epsilon > 0$ provided the condition $(x, t) \in U^-_{\mathcal{F}(\langle u^1,u^2\rangle)}$. For all n we have a sequence $\{h^n_m\}_{m\in\omega}$ which is closely approximating to $\langle u^1_n, u^2_n\rangle$ and $B((x, t), \epsilon) \not\subseteq U^-_{F(h^n_m)}$ for infinitely many m. From the set $\{h^n_m\}_{n\in\omega,m\in\omega}$ we can extract a sequence $\{\tau_n\}_{n\in\omega}$ which is closely approximating to $\langle u^1, u^2\rangle$ and $B((x, t), \epsilon) \not\subseteq U^-_{F(\tau_n)}$ for $n \in \omega$. This is a contradiction with the choice of ϵ.

Now, we define F^* for $\langle f^1, f^2\rangle \in I^0_f([a, b])$ by the following rule: $F^*(\langle f^1, f^2\rangle) = \bigvee^{\uparrow} \mathcal{F}(\langle f^1_n, f^2_n\rangle)$, where $\bigvee^{\uparrow} \langle f^1_n, f^2_n\rangle = \langle f^1, f^2\rangle$ and f^1_n, f^2_n are continuous, $n \in \omega$. Let us prove correctness of this definition. Suppose $\bigvee^{\uparrow} \langle f^1_n, f^2_n\rangle = \bigvee^{\uparrow} \langle u^1_n, u^2_n\rangle = \langle f^1, f^2\rangle$. For a fix n we have $\langle u^1, u^2\rangle \sqsubseteq \bigvee^{\uparrow} \langle f^1_n, f^2_n\rangle$ and $\langle u^1_n, u^2_n\rangle = \bigvee^{\uparrow} g.l.b.(\langle f^1_k, f^2_k\rangle, \langle u^1_n, u^2_n\rangle)$. By the property of $\mathcal{F}$, $\mathcal{F}(\langle u^1_n, u^2_n\rangle) = \bigvee^{\uparrow} \mathcal{F}(g.l.b.(< f^1_k, f^2_k >, \langle u^1_n, u^2_n\rangle))$. By monotonicity of $\mathcal{F}$, $\mathcal{F}(g.l.b.(\langle f^1_k, f^2_k\rangle, \langle u^1_n, u^2_n\rangle)) \sqsubseteq \mathcal{F}(\langle f^1_k, f^2_k\rangle)$. So $\mathcal{F}(\langle u^1_n, u^2_n\rangle) \sqsubseteq \bigvee^{\uparrow} \mathcal{F}(\langle f^1_k, f^2_k\rangle)$. As a consequence, $\bigvee^{\uparrow} \mathcal{F}(\langle u^1_n, u^2_n\rangle) \sqsubseteq \bigvee^{\uparrow} \mathcal{F}(\langle f^1_k, f^2_k\rangle)$.

Similarly we can check inclusion $\bigvee^{\uparrow} \mathcal{F}(\langle u^1_n, u^2_n\rangle) \sqsupseteq \bigvee^{\uparrow} \mathcal{F}(\langle f^1_k, f^2_k\rangle)$. Monotonicity of F^* follows from monotonicity of $\mathcal{F}$. Now we prove continuity of F^*. Let the sequence $\{\langle f^1_n, f^2_n\rangle\}_{n\in\omega}$ be monotonic, and $\bigvee^{\uparrow} \langle f^1_n, f^2_n\rangle = \langle f^1, f^2\rangle$ for $\langle f^1, f^2\rangle \in I^0_f([a, b])$. By the property of bases, $\langle f^1_n, f^2_n\rangle = \bigvee^{\uparrow} \kappa^m_n$ where $\kappa^m_n \in I^0_{f,0}([a, b])$. Put $\underset{i\le i\le n}{l.u.b.}\{\kappa^m_n\} = \lambda^m_n$. Then $\langle f^1_n, f^2_n\rangle = \bigvee^{\uparrow} \lambda^m_n$ and $\lambda^{m+1}_n \ge \lambda^m_n$. We have $\bigvee^{\uparrow} \langle f^1_n, f^2_n\rangle = \bigvee^{\uparrow} \bigvee^{\uparrow} \lambda^m_n$.

Let us check that $\bigvee^{\uparrow} F^*(\langle f^1_n, f^2_n\rangle) = F^*(\bigvee^{\uparrow} \langle f^1_n, f^2_n\rangle)$. We have $F^*(\bigvee^{\uparrow} \langle f^1_n, f^2_n\rangle) = F^*(\bigvee^{\uparrow} \bigvee^{\uparrow} \lambda^m_n) = \bigvee^{\uparrow} \bigvee^{\uparrow} F^*(\lambda^m_n) \ge \bigvee^{\uparrow} F^*(\lambda^m_n) = F^*(\bigvee^{\uparrow} \lambda^m_n) = F^*(\langle f^1_n, f^2_n\rangle)$. So $\bigvee^{\uparrow} F^*(\langle f^1_n, f^2_n\rangle) \sqsubseteq F^*(\bigvee^{\uparrow} \langle f^1_n, f^2_n\rangle)$.

Moreover, $F^*(\bigvee^{\uparrow} \langle f^1_n, f^2_n\rangle) = \bigvee^{\uparrow} \bigvee^{\uparrow} F^*(\lambda^m_n)$ and $F^*(\lambda^m_n) \sqsubseteq F^*(\langle f^1_n, f^2_n\rangle)$. So $\bigvee^{\uparrow} F^*(\langle f^1_n, f^2_n\rangle) \sqsupseteq F^*(\bigvee^{\uparrow} \langle f^1_n, f^2_n\rangle)$. Continuity of F^* is proved and so F^* is a required one.

3 Definability of Computable Functions and Functionals

To semantically characterise computable real-valued functions, operators and functionals via validity of finite formulas, we use comparative analyses with real-valued majorant-computable functions proposed in [19,20] and generalised computable operators and functionals introduced below.

The computation of real-valued function is an infinite process that produces approximations closer and closer to the result. The class of majorant-computable real-valued functions has clear and exact classifications in logical and topological terms.

3.1 Majorant-Computable Functions and Generalised Computable Operators and Functionals

To recall the notion of majorant-computability and to introduce generalised computability, let us construct the set of hereditarily finite sets $\mathrm{HF}(M)$ over a model $\mathbf{M}$. This structure is rather well studied in the theory of admissible sets [3] and permits us to define the natural numbers, to code and store information via formulas.

Let $\mathbf{M}$ be a model of a language σ_0 whose carrier set is M. We construct the set of hereditarily finite sets, $\mathrm{HF}(M)$, as follows:

1. $\mathrm{S}_0(M) \rightleftharpoons M$, $\mathrm{S}_{n+1}(M) \rightleftharpoons \mathcal{P}_\omega(\mathrm{S}_n(M)) \cup \mathrm{S}_n(M)$, where $n \in \omega$ and for every set B, $\mathcal{P}_\omega(B)$ is the set of all finite subsets of B.
2. $\mathrm{HF}(M) = \bigcup_{n\in\omega} \mathrm{S}_n(M)$.

We define $\mathbf{HF}(\mathbf{M})$ as the following model:

$$\mathbf{HF}(\mathbf{M}) \rightleftharpoons \left\langle \mathrm{HF}(M), M, \sigma_0, \emptyset_{\mathbf{HF}(\mathbf{M})}, \in_{\mathbf{HF}(\mathbf{M})} \right\rangle,$$

where the unary predicate $\emptyset$ singles out the empty set and the binary predicate symbol $\in_{\mathbf{HF}(\mathbf{M})}$ have the set-theoretic interpretation. Denote $\sigma = \sigma_0 \cup \{\in, \emptyset\}$. To introduce the notions of terms and atomic formulas we use variables of two sorts. Variables of the first sort range over M and variables of the second sort range over $\mathrm{HF}(M)$. Below we will consider $M \rightleftharpoons \mathbb{R}$ and $\sigma_0 = \{0, 1, +, \cdot, -x, \frac{x}{2}, <\}$. The notions of a term and an atomic formula are given in the standard manner.

The terms are defined inductively by:

1. The constant symbols 0 and 1 are terms;
2. The variables of the first sort are terms;
3. If t_1, t_2 are terms then $t_1 + t_2$, $t_1 \cdot t_2$, $-t_1$, $\frac{t_2}{2}$ are terms.

The following formulas are atomic: $t_1 < t_2$, $t \in s$ and $s_1 \in s_2$ where t_1, t_2, t are terms and s_1, s_2 are variables of the second sort.

The set of *Δ_0-formulas* is the closure of the set of atomic formulas under $\wedge, \vee, \neg, (\exists x \in s)$ and $(\forall x \in s)$, where $(\exists x \in s)\ \varphi$ denotes $\exists x(x \in s \wedge \varphi)$ and $(\forall x \in s)\ \varphi$ denotes $\forall x(x \in s \to \varphi)$, s is any variable of second type.

The set of *Σ-formulas* is the closure of the set of Δ_0 formulas under $\wedge, \vee$, $(\exists x \in s), (\forall x \in s)$, and $\exists$. We define *Π-formulas* as negations of Σ-formulas. The natural numbers $0, 1, \ldots$ are identified with $\emptyset, \{\emptyset, \{\emptyset\}\}, \ldots$ so that, in particular, $n + 1 = n \cup \{n\}$ and the set ω is a subset of $\mathrm{HF}(\mathbb{R})$.

Definition 13. *1. A set $B \subseteq \mathrm{HF}(\mathbb{R})$ is Σ-definable , if there exists a Σ-formula $\Phi(x)$ such that $x \in B \leftrightarrow \mathbf{HF}(\mathbb{R}) \models \Phi(x)$.*
2. A function $f : \mathrm{HF}(\mathbb{R}) \to \mathrm{HF}(\mathbb{R})$ is Σ-definable, if there exists a Σ-formula $\Phi(x, y)$ such that $f(x) = y \leftrightarrow \mathbf{HF}(\mathbb{R}) \models \Phi(x, y)$.

In a similar way, we define the notions of *Π-definable functions* and *sets*. The class of *Δ-definable functions* (*sets*) is the intersection of the class of Σ-definable functions (sets) and the class of Π-definable functions (sets).

Note that the sets $\mathbb{R}$ and $\mathbb{R}^n$ are Δ_0–definable. This fact makes $\mathbf{HF}(\mathbb{R})$ a suitable domain for studying functions from $\mathbb{R}^k$ to $\mathbb{R}$.

To introduce the definition of majorant-computability, we use a class of Σ-, Π-definable sets as the basic classes. So, we recall some useful properties of Σ-,Π-definable subsets of $\mathbb{R}^n$.

Proposition 6. *1. The sets $\mathbf{HF}(\emptyset)$, ω and the predicate of equality on $\mathbf{HF}(\emptyset)$ are Σ-definable.*
2. The set $\{\langle n, r\rangle \mid n$ is a Gödel number of a Σ-formula Φ, $r \in \mathbb{R}$, and $\mathbf{HF}(\mathbb{R}) \models \Phi(r)\}$ is Σ-definable.
3. A set $B \subseteq \mathbb{R}^n$ is Σ-definable if and only if there exists an effective sequence of formulas in the language σ_0 with existential quantifiers over the reals, $\{\Phi_s(x)\}_{s\in\omega}$, such that $x \in B \leftrightarrow \mathbb{R} \models \bigvee_{s\in\omega} \Phi_s(x)$.
4. A set $B \subseteq \mathbb{R}^n$ is Π-definable if and only if there exists an effective sequence of formulas in the language σ_0 with universal quantifiers over the reals, $\{\Phi_s(x)\}_{s\in\omega}$, such that $x \in B \leftrightarrow \mathbb{R} \models \bigwedge_{s\in\omega} \Phi_s(x)$.

Proof. The parts *1.-2.* can be easy proved by technique developed in [3,9,20]. The parts *3.-4.* immediately follow from the part *2.* □

Below we will write $(\exists n \in \mathbb{N})\, \Phi(n, x)$ instead of $\bigvee_n \Phi(\underline{n}, x)$ and $(\exists r \in \mathrm{D}_2)\, \Phi(r, x)$ instead of $\bigvee_{r,m} \left(\Phi(\frac{n}{2^m}, x) \vee \Phi(\frac{-n}{2^m}, x)\right)$, where $\underline{0} = 0 \ldots \underline{n+1} = \underline{n} + 1$.

Let us recall the notion of majorant-computability for real-valued functions proposed and investigated in [19,20,21,22]. We use the class of Σ- and Π-definable sets as the basic classes. A real-valued function is said to be *majorant-computable* if we can construct a special kind of nonterminating process computing approximations closer and closer to the result.

Definition 14. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called majorant-computable if there exist an effective sequence of Σ-formulas $\{\Phi_s(\mathbf{x}, y)\}_{s\in\omega}$ and an effective sequence of Π-formulas $\{G_s(\mathbf{x}, y)\}_{s\in\omega}$ such that the following conditions hold.*

1. *For all $s \in \omega$, $\mathbf{x} \in \mathbb{R}^n$, the formulas Φ_s and G_s define nonempty intervals $< \alpha_s, \beta_s >$ and $< \delta_s, \gamma_s >$.*
2. *For all $\mathbf{x} \in \mathbb{R}^n$, the sequences $\{< \alpha_s, \beta_s >\}_{s\in\omega}$ and $\{< \delta_s, \gamma_s >\}_{s\in\omega}$ decrease monotonically and $< \alpha_s, \beta_s > \subseteq < \delta_s, \gamma_s >$ for all $s \in \omega$.*
3. *For all $\mathbf{x} \in \mathrm{dom}(f)$, $f(\mathbf{x}) = y \leftrightarrow \bigcap_{s\in\omega} < \alpha_s, \beta_s >= \{y\} \leftrightarrow \bigcap_{s\in\omega} < \delta_s, \gamma_s >= \{y\}$ holds; for all $\mathbf{x} \notin \mathrm{dom}(f)$, $||\cap_{s\in\omega} < \delta_s, \gamma_s > || > 1$.*

The sequence $\{\Phi_s\}_{s\in\omega}$ in Definition 14 is called a *sequence of Σ-approximations* for f. The sequence $\{G_s\}_{s\in\omega}$ is called a *sequence of Π-approximations* for f. As we can see, the process which carries out the computation is represented by

two effective procedures. These procedures produce Σ-formulas and Π-formulas which define approximations closer and closer to the result. Below we will write $A(\mathbf{x},\cdot) < B(\mathbf{x},\cdot)$ if $\mathbf{HF}(\mathbb{R}) \models A(\mathbf{x},y) \wedge B(\mathbf{x},y) \to y < z$ for all real numbers y, z. The following theorem connects a majorant-computable function with validity of finite formulas in the set of hereditarily finite sets, $\mathbf{HF}(\mathbb{R})$.

Proposition 7. *For all functions $f : \mathbb{R}^n \to \mathbb{R}$ the following assertions are equivalent:.*

1. *The function f is majorant-computable.*
2. *There exist Σ–formulas $A(\mathbf{x},y)$, $B(\mathbf{x},y)$ such that $A(\mathbf{x},\cdot) < B(\mathbf{x},\cdot)$ and*

$$f(\mathbf{x}) = y \leftrightarrow (A(\mathbf{x},\cdot) < y < B(\mathbf{x},\cdot) \wedge \{z \mid A(\mathbf{x},z)\} \cup \{z \mid B(\mathbf{x},z)\} = \mathbb{R} \setminus \{y\}).$$

Proof. $\to$)Let $f : \mathbb{R}^n \to \mathbb{R}$ be majorant-computable. By Definition 14, there exist a sequence $\{F_s\}_{s\in\omega}$ of Σ-approximations for f and a sequence $\{G_s\}_{s\in\omega}$ of Π-approximations for f. Put

$$A(\mathbf{x},y) \rightleftharpoons (\exists s \in \omega)\,(y \notin < \delta_s, \gamma_s > \wedge\, (\exists z \in < \alpha_s, \beta_s >)\,(y < z))$$

and

$$B(\mathbf{x},y) \rightleftharpoons (\exists s \in \omega)\,(y \notin < \delta_s, \gamma_s > \wedge\, (\exists z \in < \alpha_s, \beta_s >)\,(y > z))\,.$$

By construction, A and B are the sought formulas.
$\leftarrow$) Let A and B satisfy the requirements of the theorem. Let us construct approximations in the following way.

$$F_s(\mathbf{x},y) \rightleftharpoons \exists z \exists v\,(A(\mathbf{x},z) \wedge B(\mathbf{x},v) \wedge y \in (z,v) \wedge v - z < 1/s)\,,$$

$$G_s(\mathbf{x},y) \rightleftharpoons \forall z\,(A(\mathbf{x},z) \to z - y < 1/s) \wedge \forall z\,(B(\mathbf{x},z) \to y - z < 1/s)\,.$$

□

As a corollary we note that a total real-valued function is majorant-computable if and only if its epigraph and ordinate set are Σ-definable (i.e. effective sets). The same proposition holds for a total function $f : [a,b]^n \to \mathbb{R}$ for some compact n-cube $[a,b]^n$.

Definition 15. *A real-valued function f is said to be shared by Σ-formulas φ_1, φ_2 if*

$$f|_{[x_1,x_2]} > c \leftrightarrow \mathbf{HF}(\mathbb{R}) \models \varphi_1(x_1,x_2,c),$$
$$f|_{[x_1,x_2]} < c \leftrightarrow \mathbf{HF}(\mathbb{R}) \models \varphi_2(x_1,x_2,c).$$

Proposition 8. *A real-valued function is majorant-computable if and only if it is shared by two Σ-formulas.*

Proof. The claim immediate follows from Proposition 7.

Theorem 3. *The class of computable real-valued functions coincides with the class of majorant-computable real-valued functions.*

Proof. Without loss of generality we consider a function $f : \mathbb{IR} \to [0,1]$. Let $f^* : \mathcal{I} \to \mathcal{I}_{[0,1]}$ be computable and $f^*(\{x\}) = \{f(x)\}$. For $n \in \omega$, we define $A_n = \{x \in \mathbb{IR} \mid \mu(f^*(\{x\})) < \frac{1}{n}\}$, where μ is Borel measure. It is easy to see that A_n is a Σ-definable open set, and $\mathrm{dom}(f) = \bigcap_{n\in\omega} A_n$.
Because each Σ-definable subset of $\mathbb{IR}$ is an effective union of open intervals, we can denote $A_1 = \bigcup_{i\in\omega}(\alpha_i, \beta_i)$, where $\alpha_i,\ \beta_i \in \mathrm{D}_2$ and $\alpha_i \leq \beta_i$.

The following formulas satisfy the conditions of Proposition 7 :

$$A(x,z) \rightleftharpoons x \in A_1 \wedge (\exists a \in \mathrm{D}_2)\,(\exists b \in \mathrm{D}_2)\,(\exists y \in \mathrm{D}_2)\,(x \in (a,b) \wedge y > z \wedge [y, y+1] \ll f^*([a,b]),$$

$$B(x,z) \rightleftharpoons x \in A_1 \wedge (\exists a \in \mathrm{D}_2)\,(\exists b \in \mathrm{D}_2)\,(\exists y \in \mathrm{D}_2)\,(x \in (a,b) \wedge y < z \wedge [y, y+1] \ll f^*([a,b])$$

By Proposition 6, f is majorant-computable.

Let f be majorant-computable and A and B satisfy the properties from Proposition 7. We construct a computable function $f^* : \mathcal{I} \to \mathcal{I}$ such that $f^*(\{x\}) = \{f(x)\}$.

Put $f^*([a,b]) = \bigcup_{x\in[a,b]} f^{**}(x)$, where the auxiliary function f^{**} is defined in the following way:

$$f^*([a,b]) = \begin{cases} \cap\{[u,v] \mid u,v \in \mathrm{D}_2,\ <x,u> \in A,\ <x,v> \in B\} & \text{if such } u,v \text{ exist} \\ \bot & \text{elsewhere} \end{cases}$$

It is easy to see that f is continuous and the set $E = <a,b,c,d> \mid a,b,c,d \in \mathrm{D}_2,\ [c,d] \ll f^*([a,b])$ is Σ-definable by the following Σ-formula

$$\exists x \in (a,b)\,(<x,c> \in A \wedge <x,d> \in B)\,.$$

So the function f is computable. □

To introduce generalised computability of operators and functionals we extend the language σ by two 3-ary predicates U_1 and U_2.

Definition 16. *A total operator $F^* : \mathcal{I}_f[a,b] \to \mathcal{I}_f[c,d]$ is said to be shared by two Σ-formulas φ_1 and φ_2 if the following assertions hold. If $F^*(\langle u^1, u^2\rangle) = \langle h^1, h^2\rangle$ then*

$$h^1|_{[x_1,x_2]} > z \leftrightarrow \mathbf{HF}(\mathbb{IR}) \models \varphi_1(U_1, U_2, x_1, x_2, z);$$
$$h^2|_{[x_1,x_2]} < z \leftrightarrow \mathbf{HF}(\mathbb{IR}) \models \varphi_2(U_1, U_2, x_1, x_2, z),$$

where $U_1(x_1,x_2,c) \rightleftharpoons u^1|_{[x_1,x_2]} > c, U_2(x_1,x_2,c) \rightleftharpoons u^2|_{[x_1,x_2]} < c$ and the predicate U_1 and U_2 positively occur in $\varphi_1,\ \varphi_2$.

Definition 17. *An operator $F : C[a,b] \to C[c,d]$ is said to be generalised computable, if there exists an operator $F^* : \mathcal{I}_f[a,b] \to \mathcal{I}_f[c,d]$ which is shared by two Σ-formulas and $F(f) = g \leftrightarrow F^*(\hat{f}) = \hat{g}$, where $\hat{f}(x) = \{f(x)\}$, $\hat{g}(x) = \{g(x)\}$.*

Definition 18. *A functional $F : C[a,b] \times [c,d] \to \mathbb{R}$ is said to be generalised computable, if there exists a computable operator $\tilde{F} : C[a,b] \to C[c,d]$ such that $F(f,x) = \tilde{F}(f)(x)$.*

Definition 19. *A functional $F : C[a,b] \times \mathbb{R} \to \mathbb{R}$ is said to be generalised computable, if there exists an effective sequence $\{\tilde{F}_n\}_{n\in\omega}$ of computable operators $\tilde{F}_n : C[a,b] \to C[-n,n]$ such that*

$$F(f,x) = y \leftrightarrow \forall n \left(-n \le x \le n \to \tilde{F}_n(f)(x) = y \right).$$

Theorem 4. *An operator $F : C[a,b] \to C[c,d]$ is computable if and only if it is generalised computable.*

Proof. Let $F : C[a,b] \to C[c,d]$ be computable. To show generalised computability of its corresponding operator $F^* : \mathcal{I}_f[a,b] \to \mathcal{I}_f[c,d]$, we construct two Σ-formulas φ_1, φ_2 satisfying the conditions of Definition 17. Let $\mathcal{I}_{f,0}([a,b]) = \{b_i\}_{i\in\omega}$ and $\mathcal{I}_{f,0}([c,d]) = \{c_i\}_{i\in\omega}$ be effective bases constructed as in Proposition 1 for $\mathcal{I}_f([a,b])$ and $\mathcal{I}_{f,0}([c,d])$.
Suppose $F^*(u) = h$. By Proposition 7 and Corollary 8 the relation $b_n \ll u$ is definable by Σ-formulas with positive occurrences of U_1 and U_2, where $U_1(r_1,r_2,c) \rightleftharpoons u^1|_{[r_1,r_2]} > c$, $U_2(r_1,r_2,c) \rightleftharpoons u^2|_{[r_1,r_2]} < c$. Therefore the set $\{(n,m) | u \gg c_n \wedge F^*(b_n) \gg b_m\}$ is definable by some Σ-formula $\Phi(n,m,U_1,U_2)$. Then $F^*(u) \gg c_m \leftrightarrow \mathbf{HF}(\mathbb{R}) \models \exists n \Phi(n,m,U_1,U_2)$.

Put

$$\varphi_1(U_1,U_2,x_1,x_2,z) \rightleftharpoons \exists m \exists n \left(b^1_m|_{[x_1,x_2]} > z \right) \wedge \Phi(n,m,U_1,U_2),$$

$$\varphi_2(U_1,U_2,x_1,x_2,z) \rightleftharpoons \exists m \exists n \left(b^2_m|_{[x_1,x_2]} < z \right) \wedge \Phi(n,m,U_1,U_2).$$

Clearly, φ_1, φ_2 are required formulas.

Let $F : C[a,b] \to C[c,d]$ be generalised computable. We prove computability of its corresponding operator $F^* : \mathcal{I}_f[a,b] \to \mathcal{I}_f[c,d]$. Monotonicity of F^* follows from positive occurrences of U_1 and U_2 in the formulas φ_1 and φ_2.

Because $\mathcal{I}_f[a,b]$ and $\mathcal{I}_f[c,d]$ are ω-continuous domains, it is enough to prove that F^* preserves suprema of countable directed sets.

Let $A = \{< u^1_n, u^2_n >\}_{n\in\omega}$ and $\bigvee^{\uparrow} A =< u^1, u^2 >$. Put $U_{1n}(x_1,x_2,c) \rightleftharpoons u^1_n|_{[x_1,x_2]} > c$ and $U_{2n}(x_1,x_2,c) \rightleftharpoons u^2_n|_{[x_1,x_2]} < c$ for $n \in \omega$ and $U_1(x_1,x_2,c) \rightleftharpoons u^1|_{[x_1,x_2]} > c$, $U_2(x_1,x_2,c) \rightleftharpoons u^2|_{[x_1,x_2]} < c$.

By Lemma 11, if $u^1|_{[x_1,x_2]} > c$ then there exists n such that $u^1_n|_{[x_1,x_2]} > c$, and if $u^2|_{[x_1,x_2]} < c$ then there exists n such that $u^2_n|_{[x_1,x_2]} < c$.

So

$$U_1(x_1,x_2,c) = \bigvee_{n\in\omega} U_{1n}(x_1,x_2,c) \text{ and } U_2(x_1,x_2,c) = \bigvee_{n\in\omega} U_{2n}(x_1,x_2,c).$$

By the properties of Σ-formulas and positive occurrences of U_1 and U_2 in φ_1 and φ_2,

$$\varphi_1(U_1, U_2, x_1, x_2, c) \leftrightarrow \bigvee_{n\in\omega} \varphi_{1n}(U_1, U_2, x_1, x_2, c),$$
$$\varphi_2(U_1, U_2, x_1, x_2, c) \leftrightarrow \bigvee_{n\in\omega} \varphi_{2n}(U_1, U_2, x_1, x_2, c).$$

Hence it is clear that $F^*(\bigvee^{\uparrow} A) = \bigvee^{\uparrow} F^*(A)$.

Now we show that the set $\{(n, m)|F^*(b_n) \gg c_m\}$ is Σ-definable and, as a consequence, is computable enumerable in n and m. Let $F^*(< b_n^1, b_n^2 >) =< h^1, h^2 >$. Since b_n^1, b_n^2, c_m^1 and c_m^2 are piecewise linear, it is obvious that the sets $b_n^1|_{[x_1,x_2]} > c$, $b_n^2|_{[x_1,x_2]} < c$ and $c_m^1|_{[x_1,x_2]} > c$, $c_m^2|_{[x_1,x_2]} < c$ are Σ-definable. As is evident from the definition of F^*, the sets $h^1|_{[x_1,x_2]} > c$, $h^2|_{[x_1,x_2]} < c$ are Σ-definable too. By Proposition 2, there exist upper semicontinuous step functions s^1 and s^2 such that $c_m^1(x) < s^1(x) < h^1(x)$ and $c_m^2(x) > s^2(x) > h^2(x)$ for $x \in [c, d]$.

As one can see, the following Σ-formula

$$\exists x_0 \ldots \exists x_n \exists y_1 \ldots \exists y_n \exists z_1 \ldots \exists z_n \bigwedge_{i\le n} \left((c_m^1|_{[x_i,x_{i+1}]} < y_i) \wedge \left(h^1|_{[x_i,x_{i+1}]} > y_i\right) \wedge \left(c_m^2|_{[x_i,x_{i+1}]} > z_i\right) \wedge \left(h^2|_{[x_i,x_{i+1}]} < z_i\right)\right.$$

defines the set $\{(n, m)|F^*(b_n) \gg c_m\}$. As a consequence this set is computable enumerable in n and m. □

Note that using the previous theorem one can elegantly prove computability of such functions as $\sup_{x\in[x_1,x_2]} f(x)$, $\inf_{x\in[x_1,x_2]} f(x)$ and Riemann integral on $[x_1, x_2]$.

Corollary 3. *A functional $F : C[a, b] \times [c, d] \to \mathbb{R}$ is computable if and only if it is generalised computable.*

Proof. The claim follows from generalised computability of its corresponding operators. □

Corollary 4. *A functional $F : C[a, b] \times \mathbb{R} \to \mathbb{R}$ is computable if and only if it is generalised computable.*

Proof. The claim follows from the property of Σ- formulas: an effective sequence of Σ-formulas is equivalent to a Σ-formula. □

3.2 Semantic Characterisation of Computable Functions and Functionals

After mentioning the main properties of majorant-computable real-valued functions and generalised computable operators and real-valued functionals, we pass to computable ones.

Corollary 5. *For a function $f : \mathbb{R}^n \to \mathbb{R}$ the following assertions are equivalent*:

1. *The function f is computable.*
2. *There exist Σ–formulas $A(\mathbf{x}, y)$ and $B(\mathbf{x}, y)$ such that $A(\mathbf{x}, \cdot) < B(\mathbf{x}, \cdot)$ and*

$$f(\mathbf{x}) = y \leftrightarrow (A(\mathbf{x}, \cdot) < y < B(\mathbf{x}, \cdot) \wedge \\ \{z \mid A(\mathbf{x}, z)\} \cup \{z \mid B(\mathbf{x}, z)\} = \mathbb{IR} \setminus \{y\}).$$

Proof. The claim follows from Proposition 7 and Theorem 3. □

Corollary 6. *A real-valued function is computable if and only if it is shared by two Σ-formulas.*

Proof. The claim follows from Proposition 8 and Theorem 3. □

Proposition 9. *Let f be a computable function such that $[a, b] \subseteq \mathrm{dom}(f)$ and g be a computable function such that $[b, c] \subseteq \mathrm{dom}(g)$ and $f(b) = g(b)$. Then the function $h(x) = \begin{cases} f(x) & \text{if } x \leq b, \\ g(x) & \text{if } x \geq b \end{cases}$ is computable.*

Proof. From Theorem 26 in [7] (cf. [21]) it follows that there exist an effective modulus of continuity w_f for f and an effective modulus of continuity w_g for g. In other words, for every $s \in \omega$ for all $x_1, x_2 \in [a, b]$ and $x_3, x_4 \in [c, d]$ we have

$$|x_1 - x_2| < w_f(\frac{1}{s}) \to |f(x_1) - f(x_2)| < \frac{1}{s} \text{ and}$$
$$|x_3 - x_4| < w_g(\frac{1}{s}) \to |g(x_3) - g(x_4)| < \frac{1}{s}.$$

Put $w_h(\epsilon) = \min\{w_f(\epsilon), w_g(\epsilon)\}$. The following Σ-formula defines the epigraph of the function h.

$$y > h(x) \leftrightarrow (x < b \wedge y > f(x)) \vee (x > b \wedge y > g(x)) \vee \\ \left(\left[\exists \epsilon \in \mathrm{D}_2^+\right] (|x - b| < w_h(\epsilon) \wedge ([\exists t < b]\, |x - t| < w_h(\epsilon) \wedge y > f(t) + \epsilon)\right)$$

Analogously, the ordinate set of h is Σ–definable. By Corollary 6, the function h is computable. □

Corollary 7. *A functional $F : C[a, b] \times [c, d] \to \mathbb{IR}$ is computable if and only if there exists an operator $F^* : \mathcal{I}_f[a, b] \to \mathcal{I}_f[c, d]$ which is shared by two Σ-formulas and $F(f, x) = y \leftrightarrow F^*(\hat{f})(x) = \{y\}$, where $\hat{f}(x) = \{f(x)\}$.*

Proof. It follows from Theorem 4. □

Corollary 8. *If a computable operator $F : C[a, b] \to C[c, d]$ is defined in a computable function f, then the function $F(f)$ is computable.*

Proof. We only note that if a function u is computable, then the following relations $u^1|_{[x_1,x_2]} > z$ and $u^2|_{[x_1,x_2]} < z$ are Σ-definable. This follows from Proposition 8. □

Corollary 9. *A total computable operator $F : C[a,b] \to C[c,d]$ maps computable functions to computable functions.*

Proof. It follows from Corollary 8. □

Corollary 10. *The composition of computable operators is computable.*

Proof. The claim follows from properties of Σ-formulas and Theorem 4. □

Now we introduce a useful recursion scheme which permits us to describe the behaviour of complex systems such as hybrid systems.

Let $\mathcal{F} : C[a,b] \times C[0,1] \times \mathbb{R} \to \mathbb{R}$ and $G : C[a,b] \times [0,1] \to \mathbb{R}$ be computable functionals. Then $F : C[a,b] \times [0,+\infty) \to \mathbb{R}$ is defined by the following scheme:

$$\begin{cases} F(f,t)|_{t\in[0,1]} = G(f,t), \\ F(f,t)|_{t\in(n,n+1]} = \mathcal{F}(f,t,\lambda y.F(f,y+n-1)) \end{cases}$$

Proposition 10. *If F is continuous then F is computable, with F defined above.*

Proof. We prove that there exists an effective sequence of generalised computable operators $F_n^* : C[a,b] \to C[0,n]$. For this we state that for each k there exist two Σ-formulas τ_1 and τ_2 which share F_k^*. Clearly, on the m-th step of computation via the recursion scheme, we obtain a computable functional where t ranges over the interval $[m, m+1]$. Hence, there exist two effective sequences of Σ-formulas $\{\tau_1^m\}_{m\in\omega}$ and $\{\tau_2^m\}_{m\in\omega}$ such that for $m \le x_1 \le x_2 \le m+1$ and $F^*(< u^1, u^2 >) =< h^1, h^2 >$ we have

$$h^1|_{[}x_1, x_2] > c \leftrightarrow \tau_1^m(U_1, U_2, x_1, x_2, c),$$
$$h^2|_{[}x_1, x_2] < c \leftrightarrow \tau_2^m(U_1, U_2, x_1, x_2, c),$$

where $U_1(x_1,x_2,c) \rightleftharpoons u^1|_{[x_1,x_2]} > c$, $U_2(x_1,x_2,c) \rightleftharpoons u^2|_{[x_1,x_2]} < c$ and the predicate U_1 and U_2 positively occur in φ_1 and φ_2. Let us denote

$$\begin{aligned} &\varphi(U_1,U_2,x_1,x_2,c) \rightleftharpoons \\ &(\exists i,j \in \mathbb{N})\, i \le j \wedge i < x_1 < i+1 \wedge j < x_2 < j+1 \wedge \tau_1^i(U_1,U_2,x_1,i+1,c) \wedge \\ &\tau_1^j(U_1,U_2,j,x_2,c) \wedge \bigwedge_{i+1\le m\le j-1} \tau_1^m(U_1,U_2,m,m+1,c); \end{aligned}$$

$$\begin{aligned} &\theta(U_1,U_2,x,c) \rightleftharpoons \\ &(\exists n \in \mathbb{N}) \left(x < n \wedge \bigwedge_{0\le m\le n-1} \tau_1^m(U_1,U_2,m,m+1,c) \right) \vee \\ &(\exists i \in \mathbb{N}) \left(i < x < i+1 \wedge \bigwedge_{0\le m\le i} \tau_1^m(U_1,U_2,m,m+1,c) \wedge \tau_1^i(U_1,U_2,i,x,c) \right). \end{aligned}$$

The required formula τ_1 can be defined as follows:

$$\tau_1(U_1, U_2, x_1, x_2, c) \rightleftharpoons$$
$$\exists y_1 \exists y_2 \, (y_1 < y_2 \wedge y_1 < x_1 \wedge x_2 < y_2 \wedge \varphi(U_1, U_2, y_1, y_2, c)) \vee$$
$$(\exists y > x_2) \wedge \theta(U_1, U_2, y, c)$$

The required formula τ^2 can be defined in the similar way. □

We would note that the recursion scheme is a useful tool for formalisation of hybrid systems. Indeed, in this framework the trajectories of the continuous component of hybrid systems (the performance specifications) can be represented by computable functionals which can be constructed by the specifications **SHS** of hybrid systems proposed in [23]. □

3.3 Comparison with Similar Domain-Theoretic Approaches

We compare approaches to higher order computability over the reals via function domains presented here and continuous domains proposed in [10,29]. Actually a function domain is a retract of the standard continuous domain $\left[\mathcal{I}_{[a,b]} \Rightarrow \mathcal{I}\right]$. The injection $\triangle : \mathcal{I}_f([a,b]) \to \left[\mathcal{I}_{[a,b]} \Rightarrow \mathcal{I}\right]$ is defined by the rule:

$$f^{\triangle}([\alpha, \beta]) = \bigcup_{x \in [\alpha, \beta]} f(x) \text{ for all } f \in \mathcal{I}_f([a,b]).$$

To show that $f^{\triangle} : \mathcal{I}_{[a,b]} \to \mathcal{I}$ is continuous let us suppose the contrary. Assume that for some decreasing sequence $\{[\alpha_i, \beta_i]\}_{i \in \omega}$ with $\bigvee^{\uparrow}[\alpha_i, \beta_i] = [\alpha, \beta]$, there exists $c \in \mathbb{R}$ such that $\inf_{\mathbb{R}} f^{\triangle}([\alpha_i, \beta_i]) < c < \inf_{\mathbb{R}} f^{\triangle}([\alpha, \beta])$ for all $i \in \omega$. It contradicts to continuity of f in α and β. Indeed, since $c < f(\alpha)$ and $c < f(\beta)$, there exists i_0 such that $\inf_{\mathbb{R}} f^{\triangle}([\alpha_{i_0}, \beta_{i_0}]) > c$. So $\triangle$ is correctly defined.

The prove of continuity of $\triangle$ is based on Lemma 1. Let $\bigvee_i^{\uparrow} f_i = f$ and for some $[\alpha, \beta] \in \mathcal{I}_{[a,b]}$ we have $\bigvee_i^{\uparrow} f_i^{\triangle}([\alpha, \beta]) = f^{\triangle}([\alpha, \beta]) = [\gamma, \delta]$. Without a loss of generality suppose that $\sup_{\mathbb{R}} \bigvee_i^{\uparrow} f_i^{\triangle}([\alpha, \beta]) > \delta_1 > \delta$. By Lemma 1, $\exists i_0 \forall i \geq i_0 \forall x \in [\alpha, \beta] f_i(x) \leq \delta_1$, so $f_i^{\triangle}([\alpha, \beta]) \leq \delta_1$, a contradiction.

So $\bigvee_i^{\uparrow} f_i = f$ and $\triangle$ is continuous.

A surjection $\bigtriangledown : \left[\mathcal{I}_{[a,b]} \to \mathcal{I}\right] \Rightarrow \mathcal{I}_f([a,b])$ is defined by the rule: $h^{\bigtriangledown}(x) = h(\{x\})$ for all $h \in \left[\mathcal{I}_{[a,b]} \Rightarrow \mathcal{I}\right]$. Its correctness is clear. The proof of continuity of the mapping $\bigtriangledown$ is standard as well as the proof of the equality $f^{\triangle \bigtriangledown} = f$ for all $f \in \mathcal{I}_f([a,b])$.

So, the function domain $\mathcal{I}_f([a,b])$ is a retract of $\left[\mathcal{I}_{[a,b]} \Rightarrow \mathcal{I}\right]$ considered in [10]. In Section 2.3 it was shown that every element of the functional domain is described in terms of semicontinuous functions. Therefore we believe that in practice it is easier to deal with elements of our domain.

The basic function space $C([a,b])$ is embedded in our function domain via natural injection em defined by the rule $f^{em} = \{f(x)\}$ for all $x \in [a,b]$, $f \in C([a,b])$.

It is worth to note, that a standard basis of the domain $[\mathcal{I}_{[a,b]} \Rightarrow \mathcal{I}]$ is generated by the subbasis consisting of the elements $a_i \Rightarrow b_j$, where a_i ranges over the basis of $\mathcal{I}_{[a,b]}$ and b_j ranges over the basis of $\mathcal{I}$ (e.g. [14,10]). In general, $\alpha \Rightarrow \beta$ is defined by the rule $(\alpha \Rightarrow \beta)(x) = \begin{cases} \beta & \text{if } \alpha \ll x, \\ \perp & \text{otherwise .} \end{cases}$

So, this standard subbasis is a subset of the image of $\triangle$ since, in general, $a \Rightarrow b = f^{\triangle}$ for the element $f \in \mathcal{I}_f([a,b])$ defined by the rule

$f(x) = \begin{cases} \beta & \text{if } \alpha \ll \{x\}, \\ \perp & \text{otherwise .} \end{cases}$

So, some standard basis of our domain is mapped onto some decidable subset of the basis of the domain $[\mathcal{I}_{[a,b]} \Rightarrow \mathcal{I}]$ via $\triangle$. Moreover, it is easy to compute the element $g = (\bigsqcup_{i=1}^{n} a_i \Rightarrow b_i)^{\bigtriangledown}$ for all $x \in [a,b]$ by the rule

$$g(x) = \begin{cases} \bigsqcup_{i:a_i \ll x} b_i \text{ , if there exists ,} \\ \perp \qquad\qquad\qquad \text{, otherwise .} \end{cases}$$

In particular, the operators $\triangle$ and $\bigtriangledown$ are computable.
Using these facts and descriptions of the relation '$\ll$' on $[\mathcal{I}_{[a,b]} \Rightarrow \mathcal{I}]$ (e.g.[14,11]), it is easy to show that every computable operator H from $[\mathcal{I}_{[a,b]} \Rightarrow \mathcal{I}]$ to $[\mathcal{I}_{[c,d]} \Rightarrow \mathcal{I}]$ induces a computable operator $F : \mathcal{I}_f([a,b]) \to \mathcal{I}_f([c,d])$ by the rule $F(f)(x) = H(f^{\triangle})^{\bigtriangledown}$. Moreover, $H(g^{em\triangle})^{\bigtriangledown} = F(g^{em})$ for all $g \in C([a,b])$. In particular, the operators H and F induce the same (partial) operator from $C([a,b])$ to $C([c,d])$. In similar manner one can extend every computable operator $F : \mathcal{I}_f([a,b]) \to \mathcal{I}_f([c,d])$ to some computable operator $H : [\mathcal{I}_{[a,b]} \Rightarrow \mathcal{I}] \to [\mathcal{I}_{[c,d]} \Rightarrow \mathcal{I}]$ by the rule $H(h) = F(h^{\triangle})^{\bigtriangledown}$. Again, the operators F and H induce the same (partial) operator from $C([a,b])$ to $C([c,d])$. So, notions of computability for operators on $C([a,b])$ defined via considered domains coincide. It is worth to remark that some semantic characterisation for higher order computability over the reals was considered in [10] in terms of the programming language PCF with additional distinguished functional Max that maps (f,a,b) to $\max_{x \in [a,b]} f(x)$. Moreover, some absoluteness principle and completeness of this language was obtained in [10]. By this reason it is not surprising that considered notions of computability coincide. It is likely that expressive power of extended PCF is the same as one of the language of Σ-formulas.

4 Conclusions

In this work we analyse computability of operators and functionals defined on the class of continuous functions. Using the small effective ω-continuous domains presented here we introduce several notions of computability. We take into consideration semantic and topological characterizations of computable objects. It was shown that domain theory, in particular the effective theory of continuous domains, can be useful for analysing computability of higher order objects over the reals. Moreover this theory can be useful for analysing computability of complex systems which is a subject for further investigation.

Acknowledgement

We are grateful to Yu. L. Ershov and K. Weihrauch for stimulating discussions. Many thanks to referees for helpful comments and suggestions for revision of the earlier version of this work.

References

1. S. Abramsky, A. Jung, Domain theory, Handbook of Logic in Computer Science, v. 3, Clarendon Press, 1994.
2. J. Blanck, Domain representability of metric space, Annals of Pure and Applied Logic, 83, 1997, pages 225–247.
3. J. Barwise, Admissible sets and structures, Berlin, Springer–Verlag, 1975.
4. A. Brown, C. Pearcy, Introduction to Analysis, Springer-Verlag, Berlin, 1989.
5. L. Blum and M. Shub and S. Smale, On a theory of computation and complexity over the reals: NP-completeness, recursive functions and universal machines, Bull. Amer. Math. Soc., (N.S.) , v. 21, no. 1, 1989, pages 1–46.
6. A. Edalat, Domain Theory and integration, Theoretical Computer Science, 151, 1995, pages 163–193.
7. A. Edalat, P. Sünderhauf, A domain-theoretic approach to computability on the real line, Theoretical Computer Science, 210, 1998, pages 73–98.
8. Yu. L. Ershov, Computable functionals of finite types, Algebra and Logic, 11(4), 1996 pages 367-437.
9. Yu. L. Ershov, Definability and computability, Plenum, New York, 1996.
10. M. H. Escardó, PCF extended with real numbers: a domain-theoretic approach to hair-order exact real number computation, PhD thesis, Imperial College, University of London, London, 1997.
11. T. Erker, M. H. Escardó,K. Keimel, The way-below relation of function spaces over semantic domain, Topology and its Applications, 89(1-2), pages 61-74, 1998.
12. M. H. Escardó, Function-space compactifications of function spaces, To appear in Topology and its Applications, 2000.
13. H. Freedman and K. Ko, Computational complexity of real functions, Theoret. Comput. Sci. , v. 20, 1982, pages 323–352.
14. G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M.W. Mislove, D.S. Scott, A Compendium Of Continuous Lattices, Springer Verlag, Berlin, 1980.
15. A. Grzegorczyk, On the definitions of computable real continuous functions, Fund. Math., N 44, 1957, pages 61–71.
16. T.A. Henzinger, Z. Manna, A. Pnueli, Towards refining Temporal Specifications into Hybrid Systems, LNCS N 736, 1993, pages 36–60.
17. T.A. Henzinger, V. Rusu, Reachability Verification for Hybrid Automata, LNCS N 1386, 1998, pages 190–205.
18. A. Jung, Cartesian Closed Categories of Domains, CWI Tract. Centrum voor Wiskunde en Informatica, Amsterdam v. 66, 1989.
19. M. Korovina, Generalized computability of real functions, Siberian Advance of Mathematics, v. 2, N 4, 1992, pages 1–18.
20. M. Korovina, O. Kudinov, A New Approach to Computability over the Reals, SibAM, v. 8, N 3, 1998, pages 59–73.
21. M. Korovina, O. Kudinov, Characteristic Properties of Majorant-Computability over the Reals, Proc. of CSL'98, LNCS, 1584, 1999, pages 188–204.
22. M. Korovina, O. Kudinov, Computability via Approximations, Bulletin of Symbolic Logic, v. 5, N 1, 1999

23. M. Korovina, O. Kudinov, A Logical approach to Specifications of Hybrid Systems, Proc. of PSI'99, to appear in LNCS, 2000, pages 10–16.
24. M. Korovina, O. Kudinov, Computability over the reals without equality, Proceedings of Mal'sev conference on Mathematical Logic, Novosibirsk, p 47, 1999.
25. Z. Manna, A. Pnueli, Verifying Hybrid Systems, LNCS N 736, 1993, pages 4–36.
26. R. Montague, Recursion theory as a branch of model theory, Proc. of the third international congr. on Logic, Methodology and the Philos. of Sc., 1967, Amsterdam, 1968, pages 63–86.
27. Y. N. Moschovakis, Abstract first order computability, Trans. Amer. Math. Soc., v. 138, 1969, pages 427–464.
28. A. Nerode, W. Kohn, Models for Hybrid Systems, Automata, Topologies, Controllability, Observability, LNCS N 736, 1993, pages 317–357.
29. Pietro Di Gianantonio, Real number computation and domain theory, Information and Computation, N 127, 1996, pages 11-25.
30. M. B. Pour-El, J. I. Richards, Computability in Analysis and Physics, Springer-Verlag, 1988.
31. D. Scott, Outline of a mathematical theory of computation, In 4th Annual Princeton Conference on Information Sciences and Systems, 1970, pages 169–176.
32. D. Scott, Continuous lattices, Lecture Notes in Mathematics, 274, Toposes, Algebraic geometry and Logic, Springer-Verlag, 1972, pages 97–136.
33. E.Schechter, Handbook of Analysis and Its Foundations, Academic Pressbook, 1996.
34. V. Stoltenberg-Hansen and J. V. Tucker, Complete local rings as domains, Journal of Symbolic Logic, 53, 1988, pages 603-624.
35. V. Stoltenberg-Hansen and J. V. Tucker, Effective algebras, Handbook of Logic in computer Science, v. 4, Clarendon Press, 1995, pages 375–526.
36. H. Tong, Some characterizations of normal and perfectly normal space, Duke Math. J. N 19, 1952, pages 289-292.
37. B.A. Trakhtenbrot, Yu.Barzdin, Finite automata: Behaviour and Syntheses, North-Holland, 1973.
38. K. Weihrauch, Computability, volume 9 of EATCS Monographs on Theoretical Computer Science, Springer, Berlin, 1987.
39. K. Weihrauch, A simple introduction to computable analysis, Informatik Berichte 171, FernUniversitat, Hagen, 1995, 2-nd edition.
40. C. Kreitz, K. Weihrauch, Complexity Theory on Real Numbers and Functions, LNCS, 145, 1983, pages 165-175.
41. K. Weihrauch, X. Zheng Computability on Continuous, Lower Semi-Continuous and Upper Semi-Continuous real Functions, LNCS, 1276, 1997, pages 166-186.

Computing a Required Absolute Precision from a Stream of Linear Fractional Transformations

Marko Krznarić

Department of Computing, Imperial College, London SW7 2BZ, UK
marko@doc.ic.ac.uk

1 Introduction

A real number can be represented as a sequence of nested, closed intervals whose lengthes tend to zero. In the LFT approach to Exact Real Arithmetic the sequence of intervals is generated by a sequence of one–dimensional linear fractional transformations (1–LFTs) applied to a base interval, [9,13,11,4,12,7].

These intervals (1–LFTs applied to the base interval) are better and better approximations to the real number. Knowing the lengthes of the intervals we can say how good the approximations are. Here, we show how to determine the lengthes of the intervals and in particular, how to obtain a required decimal precision of a real number, extending a result in [12].

2 Representation of Real Numbers

We will give a brief review of the LFT approach to Exact Real Arithmetic; for more details see [4,12,7]. Note that we will work in the one–point compactification $\mathbb{R}^* = \mathbb{R} \cup \{\infty\}$ of the real line, which is usually represented by the unit circle and the stereographic projection.

Let us denote the set of matrices by:

$$\mathbb{M} = \left\{ \begin{pmatrix} a & c \\ b & d \end{pmatrix} \;\middle|\; a, b, c, d \in \mathbb{R} \right\} \ .$$

A matrix of the form $\begin{pmatrix} a & c \\ b & d \end{pmatrix}$ induces a 1–dimensional linear fractional transformation (1–LFT), $\Theta \begin{pmatrix} a & c \\ b & d \end{pmatrix}$, which is a function from $\mathbb{R}^*$ to $\mathbb{R}^*$ given by:

$$\Theta \begin{pmatrix} a & c \\ b & d \end{pmatrix} (x) = \frac{ax + c}{bx + d} \ .$$

We can identify a 1–LFT $\Theta(M)$ with the matrix M and this identification is unique up to scaling by a non–zero number. The composition of two 1–LFTs corresponds to matrix multiplication. A non–singular matrix M maps an interval to an interval: the interval $[p, q]$ is mapped to $[Mp, Mq]$ for $\det M > 0$ and

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 169–186, 2001.

$[Mq, Mp]$ for $\det M < 0$. In $\mathbb{R}^*$, represented by the unit circle and the stereographic projection, the interval $[p, q]$ is the set of all points which belong to the arc starting from p and going anti–clockwise to q on the unit circle. For example, $[1, -1] = \{x \mid |x| \geq 1\}$.

One can easily verify that for any two intervals I and J, there exists a 1–LFT M such that $M(I) = J$. This implies that all intervals can be encoded as a 1–LFT applied to a fixed interval, which is called the BASE INTERVAL. Although the choice of the base interval is essentially not relevant (whatever holds for one, will hold, with minor adjustments, for any other base interval), we should choose it in a way to make computations as efficient as possible. There are two base intervals which have been used, namely $[-1, 1]$ and $[0, \infty]$. See [2,4,12,7] for more details.

For a base interval $[a, b]$, we say that a matrix M is REFINING if $M[a, b] \subseteq [a, b]$. The set of all refining matrices is denoted by $\mathbb{M}^+$. We also define the INFORMATION of a 1–LFT M by $\mathbf{Info}_{[a,b]}(M) = M[a, b]$. When the choice of the base interval $[a, b]$ is clear, we write $\mathbf{Info}_{[a,b]}(M)$ simply as $\mathbf{Info}(M)$.

As we have mentioned earlier, a real number can be represented as a shrinking sequence of nested, closed intervals whose length tends to zero. Or, using the facts given above, as a sequence of 1–LFTs:

$$\{x\} = \bigcap_n [p_n, q_n] \ , \qquad p_n, q_n \in \mathbb{R} \ ,$$

$$[p_n, q_n] = M_0 M_1 \dots M_n [a, b] \ , \qquad \forall n \in \mathbb{N} \ ,$$

$$\Rightarrow \qquad \{x\} = \bigcap_n M_0 M_1 \dots M_n [a, b] \ ,$$

where $[a, b]$ is the base interval, $M_0 \in \mathbb{M}$, and $M_n \in \mathbb{M}^+$, $n > 0$. This representation is called NORMAL PRODUCT. The first matrix, M_0, determines an interval which contains the real number x, while all other matrices refine that interval further and further. By analogy with the usual representation of real numbers, the first matrix of the normal product, $M_0 \in \mathbb{M}$, is called a SIGN matrix, while the matrices $M_n \in \mathbb{M}^+$ are called DIGIT matrices.

Note that most previous research has been done on LFTs (matrices) with integer coefficients, [2,4]. In such cases, the generated sequences of intervals have rational end–points.

Edalat and Potts in [2,4] proposed a standard form, called EXACT FLOATING POINT, EFP, where both sign and digit matrices belong to predetermined finite sets of matrices. The information in the sign matrices should overlap and cover $\mathbb{R}^*$. The four sign matrices proposed by Edalat and Potts correspond to rotations of the unit circle by $0°$ (i.e. identity), $90°$, $180°$ and $270°$, and form a cyclic group. Digit matrices should overlap, cover the base interval $[a, b]$, and contract distances in $[a, b]$.

3 The Base Interval $[0, \infty]$

The main reason to choose $[0, \infty]$ as a base interval is that it allows us to calculate the information of a matrix $M = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ very efficiently: $\mathbf{Info}M = [\frac{c}{d}, \frac{a}{b}]$ if $\det M > 0$ or $\mathbf{Info}M = [\frac{a}{b}, \frac{c}{d}]$ if $\det M < 0$. In the base interval $[0, \infty]$, the four sign matrices are named as follows:

$$S_+ = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = S_\infty^{4k} = \mathrm{Id} \ , \qquad \mathbf{Info}(S_+) = [0, \infty] \ ,$$

$$S_\infty = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} = S_\infty^{4k+1} \ , \qquad \mathbf{Info}(S_\infty) = [1, -1] \ ,$$

$$S_- = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = S_\infty^{4k+2} \ , \qquad \mathbf{Info}(S_-) = [\infty, 0] \ ,$$

$$S_0 = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} = S_\infty^{4k+3} \ , \qquad \mathbf{Info}(S_0) = [-1, 1] \ ,$$

for any $k \in \mathbb{N}$. It is easy to check that, for example, $S_0 S_\infty = S_\infty S_0 = S_+ = \mathrm{Id}$.

Edalat and Potts have shown, [4], that for any base $b > 1$, the signed digit representation on $[-1, 1]$ in base b induces via the homeomorphism $S_0 : [0, \infty] \to [-1, 1]$ a suitable set of digit matrices. The signed digit system in base $b > 1$ in $[-1, 1]$ is generated by the contracting maps $A_d : [-1, 1] \to [-1, 1]$, given by:

$$A_d(x) = \begin{pmatrix} 1 & d \\ 0 & b \end{pmatrix}(x) = \frac{x + d}{b} \ ,$$

with $d \in \mathsf{Dig}(b) = \{-b + n, b - n \mid n \in \mathbb{N},\ 1 \leq n \leq \lfloor b \rfloor\}$. We now define the digit matrices in base b:

$$D_d = S_0^{-1} A_d S_0 = S_\infty \begin{pmatrix} 1 & d \\ 0 & b \end{pmatrix} S_0 = \begin{pmatrix} b + d + 1 & b + d - 1 \\ b - d - 1 & b - d + 1 \end{pmatrix} \ ,$$

where $d \in \mathsf{Dig}(b)$. For example, $\mathsf{Dig}(2) = \{-1, 0, 1\}$, $\mathsf{Dig}(\phi) = \{-\phi + 1, \phi - 1\}$, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio.

Products of digit matrices can be obtained using the following:

Proposition 1. *The product, $D_{d_1} D_{d_2} \ldots D_{d_n}$, of digit matrices in base b is given by:*

$$D_{d_1} D_{d_2} \ldots D_{d_n} = \begin{pmatrix} b^n + c + 1 & b^n + c - 1 \\ b^n - c - 1 & b^n - c + 1 \end{pmatrix} =: \mathfrak{D}_c^n \ ,$$

where $c = c(d_1, d_2, \ldots, d_n) = \sum_{i=1}^n d_i b^{n-i}$. Furthermore, $\mathfrak{D}_c^n D_d = \mathfrak{D}_{bc+d}^{n+1}$, $D_d \mathfrak{D}_c^n = \mathfrak{D}_{d \cdot b^n + c}^{n+1}$ and $\mathfrak{D}_c^n \mathfrak{D}_{c'}^{n'} = \mathfrak{D}_{c \cdot b^{n'} + c'}^{n+n'}$.

Proof. As $D_d = S_\infty \begin{pmatrix} 1 & d \\ 0 & b \end{pmatrix} S_0$ and $S_0 S_\infty = \mathrm{Id}$ we have:

$$\begin{aligned} D_{d_1} D_{d_2} \dots D_{d_n} &= S_\infty \begin{pmatrix} 1 & d_1 \\ 0 & b \end{pmatrix} S_0 S_\infty \begin{pmatrix} 1 & d_2 \\ 0 & b \end{pmatrix} S_0 \dots S_\infty \begin{pmatrix} 1 & d_n \\ 0 & b \end{pmatrix} S_0 \\ &= S_\infty \begin{pmatrix} 1 & d_1 \\ 0 & b \end{pmatrix} \begin{pmatrix} 1 & d_2 \\ 0 & b \end{pmatrix} \dots \begin{pmatrix} 1 & d_n \\ 0 & b \end{pmatrix} S_0 \\ &= S_\infty \begin{pmatrix} 1 & c \\ 0 & b^n \end{pmatrix} S_0 \\ &= \begin{pmatrix} b^n + c + 1 & b^n + c - 1 \\ b^n - c - 1 & b^n - c + 1 \end{pmatrix} , \end{aligned}$$

where $c = \sum_{i=1}^{n} d_i b^{n-i}$. Let us check the second claim:

$$\begin{aligned} \mathfrak{D}_c^n D_d &= \begin{pmatrix} b^n + c + 1 & b^n + c - 1 \\ b^n - c - 1 & b^n - c + 1 \end{pmatrix} \begin{pmatrix} b + d + 1 & b + d - 1 \\ b - d - 1 & b - d + 1 \end{pmatrix} \\ &= \begin{pmatrix} 2b^{n+1} + 2bc + 2d + 2 & 2b^{n+1} + 2bc + 2d - 2 \\ 2b^{n+1} - 2bc - 2d - 2 & 2b^{n+1} - 2bc - 2d + 2 \end{pmatrix} \\ &= \begin{pmatrix} b^{n+1} + (bc + d) + 1 & b^{n+1} + (bc + d) - 1 \\ b^{n+1} - (bc + d) - 1 & b^{n+1} - (bc + d) + 1 \end{pmatrix} \\ &= \mathfrak{D}_{bc+d}^{n+1} . \end{aligned}$$

Similarly, we prove the last two equalities. □

The number c provides a compact representation for this product of digit matrices. However, the original sequence of digits usually cannot be recovered from $\mathfrak{D}_c^n$.

The most common base in both theory and practise is the base $b = 2$. As $\mathsf{Dig}(2) = \{-1, 0, 1\}$ we get the following three digit matrices:

$$D_{-1} = S_\infty \begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix} S_0 = \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix} , \qquad \mathbf{Info}(D_{-1}) = [0, 1] ,$$

$$D_0 = S_\infty \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} S_0 = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix} , \qquad \mathbf{Info}(D_0) = [\tfrac{1}{3}, 3] ,$$

$$D_1 = S_\infty \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix} S_0 = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} , \qquad \mathbf{Info}(D_1) = [1, \infty] .$$

Any sequence of n digits in base 2, $D_{d_1}, D_{d_2}, \dots, D_{d_n}$ can be compressed into the number $c(d_1, d_2, \dots, d_n)$, which can be represented by $n+1$ bits of memory.

The fact that for $1 < b < 2$ we get only two digit matrices can be exploited in practise as we need only one bit of memory to denote a digit matrix. The golden ratio $\phi = \frac{1+\sqrt{5}}{2}$, which is the positive root of the equation $x^2 - x - 1 = 0$, satisfies the above property and has been extensively studied [1,8]. It provides an interesting base because the rules for addition and subtraction using

the positional representation for the real numbers is extremely elegant. Using identities obtained from $\phi^2 = \phi + 1$, we have $\mathsf{Dig}(\phi) = \{-\frac{1}{\phi}, \frac{1}{\phi}\}$ and

$$D_{-\frac{1}{\phi}} = \begin{pmatrix} 1 & 0 \\ \frac{1}{\phi} & \phi \end{pmatrix} \quad , \qquad \mathbf{Info}(D_{-\frac{1}{\phi}}) = [0, \phi] \ ,$$

$$D_{\frac{1}{\phi}} = \begin{pmatrix} \phi & \frac{1}{\phi} \\ 0 & 1 \end{pmatrix} \quad , \qquad \mathbf{Info}(D_{\frac{1}{\phi}}) = [\tfrac{1}{\phi}, \infty] \ .$$

3.1 $S_0 D_{d_1} D_{d_2} \ldots$

The information of the sign matrix S_0 is the interval $[-1, 1]$, i.e. $\mathbf{Info}(S_0) = [-1, 1]$. Any digit in base b will contract distances on the real line by a factor $\frac{1}{b}$. We have the following:

Proposition 2.

$$\mathsf{width}(\mathbf{Info}(S_0 D_{d_1} D_{d_2} \ldots D_{d_n})) = \frac{2}{b^n} \ ,$$

where D_{d_i}, $i = 1 \ldots n$, are digits in base b.

Proof. Using Proposition 1 we get:

$$\begin{aligned} \mathsf{width}(\mathbf{Info}(S_0 D_{d_1} D_{d_2} \ldots D_{d_n})) &= \mathsf{width}(S_0 \mathfrak{D}_c^n [0, \infty]) \\ &= \mathsf{width}\left(\begin{pmatrix} c+1 & c-1 \\ b^n & b^n \end{pmatrix} [0, \infty]\right) \\ &= \mathsf{width}\left(\left[\frac{c-1}{b^n}, \frac{c+1}{b^n}\right]\right) \\ &= \frac{2}{b^n} \ . \end{aligned}$$

□

With the above proposition, it is easy to calculate a sufficient number of digits which will achieve any required accuracy.

3.2 $S_+ D_{d_1} D_{d_2} \ldots$

All real numbers represented by the product $S_+ D_{d_1} D_{d_2} \ldots = D_{d_1} D_{d_2} \ldots$ are points of the interval $\mathbf{Info}_{[0,\infty]}(S_+) = [0, \infty]$. An infinite product of digits D_{b-1} in base b represents ∞. Any other infinite product can be written as a finite product of D_{b-1} (called the exponent), followed by an infinite product of digits starting with D_d where $-(b-1) \leq d \leq (b-2)$ (called the mantissa). Any such product represents a non–negative real number $x \in [0, \infty)$.

If the product represents ∞, as it happens with $\frac{1}{0}$, the computation of the interval lengthes will not finish in finite time (unless we put a time limit). In all other cases, using Proposition 4 below, we will be able to determine how many digit matrices will be sufficient to satisfy the required accuracy. Proposition 4 is a generalisation of Proposition 8, [12], which we present later.

Lemma 3. *Let $d_1, d_2, \ldots, d_n$ be real numbers such that $|d_i| \leq (b-1)$, $i = 2, \ldots, n$ and $-(b-1) \leq l \leq d_1 \leq u \leq (b-2)$. Then, $(l-1)b^{n-1}+1 \leq c \leq (u+1)b^{n-1}-1$, where $c = \sum_{i=1}^{n} d_i b^{n-i}$.*

Proof. We have:

$$\begin{aligned} c &\geq lb^{n-1} - (b-1)b^{n-2} - \ldots - (b-1)b^0 \\ &= lb^{n-1} - (b^{n-1}-1) \\ &= (l-1)b^{n-1}+1 \ , \end{aligned}$$

$$\begin{aligned} c &\leq ub^{n-1} + (b-1)b^{n-2} + (b-1)b^{n-3} + \ldots + (b-1)b^0 \\ &= ub^{n-1} + (b^{n-1}-1) \\ &= (u+1)b^{n-1} - 1 \ . \end{aligned}$$

□

Proposition 4. *Let $D^e_{b-1}D_{d_1}D_{d_2}\ldots D_{d_n}$ be a finite product of digits in base b with $d_1 \neq (b-1)$. Then:*

$$b^{e-n} < \mathsf{width}(\mathbf{Info}(D^e_{b-1}D_{d_1}D_{d_2}\ldots D_{d_n})) < 4b^{e-n+2} \ .$$

Proof. The product $D^e_{b-1}D_{d_1}D_{d_2}\ldots D_{d_n} = D^e_{b-1}\mathfrak{D}^n_c$ can be compressed into $\mathfrak{D}^{n'}_{c'}$ where

$$\begin{aligned} n' &= e+n \ , \\ c' &= \left[(b-1)b^{e+n-1} + \ldots + (b-1)b^n\right] + \left[d_1 b^{n-1} + \ldots + d_{n-1}b + d_n\right] \\ &= b^n(b-1)\frac{b^e-1}{b-1} + c \\ &= b^{e+n} - b^n + c \ . \end{aligned}$$

The information of $D^e_{b-1}\mathfrak{D}^n_c$ is given by:

$$\begin{aligned} \mathbf{Info}(D^e_{b-1}\mathfrak{D}^n_c) &= \begin{pmatrix} b^{n'}+c'+1 & b^{n'}+c'-1 \\ b^{n'}-c'-1 & b^{n'}-c'+1 \end{pmatrix}[0,\infty] \\ &= \begin{pmatrix} 2b^{e+n}-(b^n-c-1) & 2b^{e+n}-(b^n-c+1) \\ b^n-c-1 & b^n-c+1 \end{pmatrix}[0,\infty] \\ &= \left[\frac{2b^{e+n}}{b^n-c+1}-1, \frac{2b^{e+n}}{b^n-c-1}-1\right] \ . \end{aligned}$$

Therefore,

$$\begin{aligned} \mathsf{width}(\mathbf{Info}(D^e_{b-1}\mathfrak{D}^n_c)) &= \frac{2b^{e+n}}{b^n-c-1} - \frac{2b^{e+n}}{b^n-c+1} \\ &= \frac{4b^{e+n}}{(b^n-c)^2-1} \ . \end{aligned} \tag{1}$$

This holds for any sequence of digits, even if all the n digits compressed into $\mathfrak{D}_c^n$ are D_{b-1}. Then,

$$\begin{aligned}\frac{4b^{e+n}}{(b^n-c)^2-1} &= \frac{4b^{e+n}}{(b^n-(b^n-1))^2-1}\\ &= \frac{4b^{e+n}}{0}\\ &= \infty\ ,\end{aligned}$$

which corresponds to the length of the interval:

$$\begin{aligned}\mathbf{Info}(D_{b-1}^e\mathfrak{D}_{b^n-1}^n) &= \left[\frac{2b^{e+n}}{b^n-c+1}-1, \frac{2b^{e+n}}{b^n-c-1}-1\right]\\ &= [b^{e+n}-1, \infty]\ .\end{aligned}$$

For $d_1 \neq b-1$, i.e. $-(b-1) \leq d_1 \leq (b-2)$, using Lemma 3 we get that $-b^n+1 \leq c \leq (b-1)b^{n-1}-1$. Note that the (1) is a monotonic function in c. Therefore, the upper and lower bounds of $\mathsf{width}(\mathbf{Info}(D_{b-1}^e\mathfrak{D}_c^n))$ are respectively obtained at $c=(b-1)b^{n-1}-1$ and $c=-b^n+1$:

$$\begin{aligned}\mathsf{width}(\mathbf{Info}(D_{b-1}^e\mathfrak{D}_c^n)) &= \frac{4b^{e+n}}{(b^n-c)^2-1}\\ &\leq \frac{4b^{e+n}}{(b^n-(b-1)b^{n-1}+1)^2-1}\\ &= \frac{4b^{e+n}}{b^{2(n-1)}+2b^{n-1}}\\ &= \frac{4b^{e+1}}{b^{n-1}+2}\\ &< 4b^{e-n+2}\ ,\end{aligned}$$

$$\begin{aligned}\mathsf{width}(\mathbf{Info}(D_{b-1}^e\mathfrak{D}_c^n)) &\geq \frac{4b^{e+n}}{(b^n+b^n-1)^2-1}\\ &= \frac{4b^{e+n}}{4b^{2n}-4b^n}\\ &= \frac{b^e}{b^n-1}\\ &> b^{e-n}\ .\end{aligned}$$

□

Therefore, as soon as we get a digit in base b different from D_{b-1}, i.e. when we have determined a non–negative integer e as in Proposition 4, we can use the above formula to calculate the total number of digit matrices which will guarantee the absolute tolerance we wish to achieve.

3.3 $S_- D_{d_1} D_{d_2} \ldots$

This case is basically the same as in the previous section. In base b, the sequence $S_- D_{-(b-1)} D_{-(b-1)} D_{-(b-1)} \ldots$ will represent ∞. Hence, any attempt to obtain an absolute precision will be futile. For any other sequence we can use a variation of Proposition 4, in which $D_{-(b-1)}$ takes the position of D_{b-1}.

3.4 $S_\infty D_{d_1} D_{d_2} \ldots$

As we have seen earlier in Sections 3.2 and 3.3 the information of an EFP may contain ∞. Of course, that will prevent us from obtaining any absolute precision. That is also the case when the sign matrix of an EFP is S_∞. We have the following:

Proposition 5. *The information of an EFP $S_\infty \mathfrak{D}_c^n$ will contain ∞ if and only if $|c| \leq 1$.*

Proof. We have:

$$\begin{aligned}
\mathbf{Info}(S_\infty \mathfrak{D}_c^n) &= \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} b^n + c + 1 & b^n + c - 1 \\ b^n - c - 1 & b^n - c + 1 \end{pmatrix} [0, \infty] \\
&= \begin{pmatrix} b^n & b^n \\ -1 - c & 1 - c \end{pmatrix} [0, \infty] \\
&= \left[\frac{b^n}{1-c}, \frac{b^n}{-1-c} \right] .
\end{aligned}$$

If $c > 1$ then $-1 - c < 1 - c < 0$ which implies

$$-\infty < \frac{b^n}{1-c} < \frac{b^n}{-1-c} < 0 .$$

Similarly, if $c < -1$ the interval $\mathbf{Info}(S_\infty \mathfrak{D}_c^n)$ does not contain ∞. On the other hand, for $|c| \leq 1$ we have $-1 - c \leq 0 \leq 1 - c$ implying that the interval

$$\left[\frac{b^n}{1-c}, \frac{b^n}{-1-c} \right] ,$$

contains ∞. This completes the proof. □

If $|c| > 1$, then the interval

$$\mathbf{Info}(S_\infty \mathfrak{D}_c^n) = \left[\frac{b^n}{1-c}, \frac{b^n}{-1-c} \right] ,$$

is bounded, with width given by:

$$\begin{aligned}
\mathsf{width}(\mathbf{Info}(S_\infty \mathfrak{D}_c^n)) &= \tfrac{b^n}{-1-c} - \tfrac{b^n}{1-c} \\
&= \tfrac{2b^n}{c^2-1} .
\end{aligned}$$

Every subsequent digit matrix will contract the interval by $\frac{1}{b}$, which helps us to determine how many more digits we need for required accuracy. For a general real number $b > 1$ we need c in order to express lower and upper bounds. However, we are able to determine the bounds for integer bases $b \in \mathbb{N},\ b \geq 2$, as well as for $b = \phi$ without computing c explicitly. In this section we will consider integer bases and in Section 3.6 we study $b = \phi$.

Assume throughout this section that b is an integer base, i.e. $b \in \mathbb{N},\ b \geq 2$. For such b the set $\mathsf{Dig}(b) = \{1-b, 2-b, \cdots, -1, 0, 1, \cdots, b-2, b-1\}$ contains integer values only. We have that $c(d_1, d_2, \ldots, d_n) = 0$ if and only if $d_1 = d_2 = \ldots = d_n = 0$. Furthermore, $c(d_1, d_2, \ldots, d_n)$ is 1, respectively -1, iff the product of digit matrices $D_{d_1} D_{d_2} \ldots D_{d_n}$ is of the form $D_0^e D_1 D_{-(b-1)}^{n-e-1}$, respectively $D_0^e D_{-1} D_{b-1}^{n-e-1}$, for an integer b and some $e \in \mathbb{N},\ e < n$. We use these facts for the following two Propositions:

Proposition 6. *Let $S_\infty D_0^e D_{-1} D_{b-1}^f D_{d_1} D_{d_2} \ldots D_{d_n}$ be a finite product of matrices in an integer base b with $d_1 \neq (b-1)$ (that is, the information of such a product is a bounded interval). Then:*

$$\tfrac{1}{2} b^{e+f-n+1} < \mathsf{width}(\mathbf{Info}(S_\infty D_0^e D_{-1} D_{b-1}^f \mathfrak{D}_c^n)) < 2b^{e+f-n+3} \ ,$$

where $\mathfrak{D}_c^n = D_{d_1} D_{d_2} \ldots D_{d_n}$.

Proof. Since $D_{-1} D_{b-1} = \mathfrak{D}_{-1}^2 = D_0 D_{-1}$ we obtain $D_{-1} D_{b-1}^f = D_0^f D_{-1}$. Hence we can assume $f = 0$ and at the end, replace e by $e + f$. Using Proposition 1 we have:

$$\begin{aligned}
S_\infty D_0^e D_{-1} \mathfrak{D}_c^n &= S_\infty D_0^e \mathfrak{D}_{-1 \cdot b^n + c}^{n+1} \\
&= S_\infty \mathfrak{D}_{0 - b^n + c}^{e+n+1} \\
&= \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} b^{e+n+1} - b^n + c + 1 & b^{e+n+1} - b^n + c - 1 \\ b^{e+n+1} + b^n - c - 1 & b^{e+n+1} + b^n - c + 1 \end{pmatrix} \\
&= \begin{pmatrix} b^{e+n+1} & b^{e+n+1} \\ b^n - c - 1 & b^n - c + 1 \end{pmatrix} \ .
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathsf{width}(\mathbf{Info}(S_\infty D_0^e D_{-1} \mathfrak{D}_c^n)) &= \mathsf{width}\left(\begin{pmatrix} b^{e+n+1} & b^{e+n+1} \\ b^n - c - 1 & b^n - c + 1 \end{pmatrix} [0, \infty] \right) \\
&= \mathsf{width}\left(\left[\frac{b^{e+n+1}}{b^n - c + 1}, \frac{b^{e+n+1}}{b^n - c - 1} \right] \right) \\
&= \frac{b^{e+n+1}}{b^n - c - 1} - \frac{b^{e+n+1}}{b^n - c + 1} \\
&= \frac{2b^{e+n+1}}{(b^n - c)^2 - 1} \ .
\end{aligned}$$

As $-(b-1) \leq d_1 \leq (b-2)$, using Lemma 3 we have that $-(b^n-1) \leq c \leq (b-1)b^{n-1}-1$, which implies:

$$\begin{aligned}
\mathsf{width}(\mathbf{Info}(S_\infty D_0^e D_{-1} \mathfrak{D}_c^n)) &= \frac{2b^{e+n+1}}{(b^n-c)^2-1} \\
&\leq \frac{2b^{e+n+1}}{(b^n-(b-1)b^{n-1}+1)^2-1} \\
&= \frac{2b^{e+n+1}}{b^{2(n-1)}+2b^{n-1}} \\
&= \frac{2b^{e+2}}{b^{n-1}+2} \\
&< 2b^{e-n+3} \ , \\
\mathsf{width}(\mathbf{Info}(S_\infty D_0^e D_{-1} \mathfrak{D}_c^n)) &\geq \frac{2b^{e+n+1}}{(b^n+b^n-1)^2-1} \\
&= \frac{2b^{e+n+1}}{4b^{2n}-4b^n} \\
&= \frac{b^{e+1}}{2(b^n-1)} \\
&> \frac{1}{2}b^{e-n+1} \ .
\end{aligned}$$

□

Proposition 7. *Let $S_\infty D_0^e D_1 D_{-(b-1)}^f D_{d_1} D_{d_2} \dots D_{d_n}$ be a finite product of matrices in base b with $d_1 \neq -(b-1)$ (that is, the information of such a product is a bounded interval). Then:*

$$\tfrac{1}{2}b^{e+f-n+1} < \mathsf{width}(\mathbf{Info}(S_\infty D_0^e D_1 D_{-(b-1)}^f \mathfrak{D}_c^n)) < 2b^{e+f-n+3} \ ,$$

where $\mathfrak{D}_c^n = D_{d_1} D_{d_2} \dots D_{d_n}$.

3.5 Absolute Decimal Precision in Base $b = 2$

Usually we want to obtain the decimal precision of a given number. Yet, base $b = 2$ is the most common base in both theory and practice. Therefore, the issue of computing the absolute decimal precision in base $b = 2$ is of great importance [5].

Of course, we can use the above propositions for a general base b, but in base 2 we obtain better bounds obtained by taking more digits into account.

$S_0 D_{d_1} D_{d_2} \dots$. If the sign digit is S_0, we do not have any problems to yield any required absolute precision. The information of S_0 is a bounded interval and

every digit matrix will halve the previous interval. We use 3.322 as an upper bound for $\log_2 10$. If we choose

$$n = \lceil 1 + 3.322k \rceil \ ,$$

where k is the required decimal accuracy, we have:

$$\begin{aligned} & n = \lceil 1 + 3.322k \rceil \ , \\ \Rightarrow \quad & n > 1 + k \log_2 10 \ , \\ \Rightarrow \quad & -k \log_2 10 > 1 - n \ , \\ \Rightarrow \quad & 10^{-k} > 2^{1-n} \\ & \qquad = \mathsf{width}(\mathbf{Info}(S_0 D_{d_1} D_{d_2} \ldots D_{d_n})) \ . \end{aligned}$$

$S_+ D_{d_1} D_{d_2} \ldots$, $S_- D_{d_1} D_{d_2} \ldots$. In the case when the sign digit is S_+ we wait until we get a digit other than D_1 (recall that $S_+ D_1 D_1 \ldots$ represents ∞). Then we will get a bounded interval (one which does not contain ∞) and every subsequent digit will refine that interval. Basically, we require that e, the number of leading digits D_1, is finite. We call such a digit an EFFECTIVE digit. Similarly, if the sign matrix is S_- we wait until we get a digit other that D_{-1}.

The following proposition, which we use below to determine a sufficient number of digits in base 2 for a required absolute decimal precision, is given in [12], page 129.

Proposition 8. *For any $e \in \mathbb{N}$, $\alpha \in \{-1, 0\}$, $\beta \in \{-1, 0, 1\}$ and $\gamma \in \mathbb{Z}$, if*

$$n = e - 1 - \gamma + (1 + \alpha)(1 + \beta) \ ,$$

then,

$$2^{\gamma - 1} < \mathsf{width}(\mathbf{Info}(D_1^e D_\alpha D_\beta \mathfrak{D}_c^n)) < 2^{\gamma + 1} \ ,$$

for all $c \in \mathbb{Z}$, $|c| < 2^n$.

Suppose that we have a real number whose EFP representation is given by $S_+ D_{d_1} D_{d_2} \ldots$. Provided that e, the number of leading digit matrices D_1 is finite, we can calculate correctly the number up to the required decimal accuracy 10^{-k} by emitting not more than $e + 2 + n$ digit matrices, where

$$n = \lceil e + 3.322k + 2 \rceil \ .$$

The reasoning is as follows:

$$\begin{aligned} & n = \lceil e + 3.322k + 2 \rceil \ , \\ \Rightarrow \quad & n > e + k \log_2 10 + 2 \ , \\ \Rightarrow \quad & -k \log_2 10 > e - n + 2 \ , \\ \Rightarrow \quad & 10^{-k} > 2^{e-n+2} \\ & \qquad \geq 2^{e-n+(1+\alpha)(1+\beta)} \\ & \qquad = 2^{\gamma+1} \ , \end{aligned}$$

where $\gamma = e - n - 1 + (1+\alpha)(1+\beta)$. By Proposition 8:

$$\mathsf{width}(\mathbf{Info}(D_1^e D_\alpha D_\beta \mathfrak{D}_c^n)) < 2^{\gamma+1} < 10^{-k} \ ,$$

for all $c \in \mathbb{Z},\ |c| < 2^n$.

The same conclusion can be used in the case $S_- D_{d_1} D_{d_2} \ldots$. The only difference is that e is the number of leading digit matrices D_{-1}.

$S_\infty D_{d_1} D_{d_2} \ldots$. Using Proposition 5 it is easy to check that in the case when the leading matrix is S_∞, we need two effective digits. The first digit is D_{-1}, respectively D_1, while the second one is either D_{-1} or D_0, respectively D_0 or D_1. The reason is that all of the sequences:

$$\begin{array}{ll} S_\infty D_0 D_0 \ldots & (\Leftrightarrow c = 0) \ , \\ S_\infty D_0^e D_{-1} D_1 D_1 \ldots & (\Leftrightarrow c = -1) \ , \\ S_\infty D_0^e D_1 D_{-1} D_{-1} \ldots & (\Leftrightarrow c = 1) \ , \end{array}$$

represent ∞.

Once the information of $S_\infty D_{d_1} D_{d_2} \ldots$ is either positive or negative (i.e. after getting the first effective digit) we can use the proposition below.

Proposition 9. *For $e, f \in \mathbb{N}$ we have:*

$$\begin{aligned} \mathsf{width}(\mathbf{Info}(S_\infty D_0^e D_{-1} D_1^f \mathfrak{D}_c^n)) &= \mathsf{width}(\mathbf{Info}(S_+ D_1^{e+f} \mathfrak{D}_c^n)) \ , \\ \mathsf{width}(\mathbf{Info}(S_\infty D_0^e D_1 D_{-1}^f \mathfrak{D}_c^n)) &= \mathsf{width}(\mathbf{Info}(S_- D_{-1}^{e+f} \mathfrak{D}_c^n)) \ . \end{aligned}$$

Proof. Let us first prove the first equality. From the proofs of Proposition 4 and Proposition 6 putting $b - 1 = 1$ we get:

$$\begin{aligned} \mathbf{Info}(S_+ D_1^{e+f} \mathfrak{D}_c^n) &= [A - 1, B - 1] \ , \\ \mathbf{Info}(S_\infty D_0^{e+f} D_{-1} \mathfrak{D}_c^n) &= [A, B] \ , \end{aligned}$$

where

$$A = \frac{2^{n+e+f+1}}{2^n - c + 1} \ , \qquad B = \frac{2^{n+e+f+1}}{2^n - c - 1} \ .$$

This implies the first equality. Similarly, we prove the second one. □

The proposition above enables us to use Proposition 8 in order to determine the number of necessary digits which will produce the required absolute decimal precision. For finite e and f, in order to achieve absolute decimal precision of 10^{-k} ($k \in \mathbb{N}$), we need less than $e + f + 2 + n$ digit matrices, where

$$n = \lceil e + f + 3.322k + 2 \rceil \ .$$

3.6 Absolute Decimal Precision in Base $b = \phi$

As we have already mentioned, ϕ is another interesting candidate for the base. Proposition 2 and Proposition 4 are proven for a general b (not only integers), hence they can be used to determine the required number of digits when the sign matrix is either $S_0,\ S_+$ or S_-.

$\boldsymbol{S_0 D_{d_1} D_{d_2} \ldots}$. The information of S_0 is an interval $[-1, 1]$ and every digit matrix will shrink the previous interval by a factor $\frac{1}{\phi}$. For

$$n = \left\lceil \frac{\log 2 + k}{\log \phi} \right\rceil \ ,$$

where k is the required decimal accuracy, we have the following:

$$\begin{aligned}
& n = \left\lceil \tfrac{\log 2 + k}{\log \phi} \right\rceil \ , \\
\Rightarrow \quad & n \geq \tfrac{\log 2 + k}{\log \phi} \ , \\
\Rightarrow \quad & -k \geq \log 2 - n \log \phi \ , \\
\Rightarrow \quad & -k \geq \log \tfrac{2}{\phi^n} \\
\Rightarrow \quad & 10^{-k} \geq \tfrac{2}{\phi^n} \\
\Rightarrow \quad & 10^{-k} \geq \mathsf{width}(\mathbf{Info}(S_0 D_{d_1} D_{d_2} \ldots D_{d_n})) \ .
\end{aligned}$$

$\boldsymbol{S_+ D_{d_1} D_{d_2} \ldots}$, $\boldsymbol{S_- D_{d_1} D_{d_2} \ldots}$. If the sign digit is S_+ we wait until we get a digit $D_{-\frac{1}{\phi}}$ (otherwise, the number represented will be ∞). Let e be a finite number of leading digit matrices $D_{\frac{1}{\phi}}$. Calculating $e + n$ digit matrices, where

$$n = \left\lceil \frac{\log 4\phi^{e+2} + k}{\log \phi} \right\rceil \ ,$$

we will yield the required decimal accuracy 10^{-k}:

$$\begin{aligned}
& n = \left\lceil \tfrac{\log 4\phi^{e+2} + k}{\log \phi} \right\rceil \ , \\
\Rightarrow \quad & n \geq \tfrac{\log 4\phi^{e+2} + k}{\log \phi} \ , \\
\Rightarrow \quad & -k \geq \log 4\phi^{e+2} - n \log \phi \ , \\
\Rightarrow \quad & 10^{-k} \geq 4\phi^{e-n+2} \ , \\
\Rightarrow \quad & 10^{-k} \geq \mathsf{width}(\mathbf{Info}(S_+ D_{\frac{1}{\phi}}^e D_{d_1} D_{d_2} \ldots D_{d_n})) \ .
\end{aligned}$$

The same conclusion can be drawn in the case $S_- D_{d_1} D_{d_2} \ldots$. The only difference is that e is the number of leading digit matrices $D_{-\frac{1}{\phi}}$.

$\boldsymbol{S_\infty D_{d_1} D_{d_2} \ldots}$. Using identity $\phi^2 = \phi + 1$, i.e. $\phi = 1 + \frac{1}{\phi}$, we have that $D_{-\frac{1}{\phi}} D_{\frac{1}{\phi}} D_{\frac{1}{\phi}} = D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}} = \mathfrak{D}_0^3$. Therefore, without loss of generality we can assume that the product $D_{-\frac{1}{\phi}} D_{\frac{1}{\phi}} D_{\frac{1}{\phi}}$ does not appear in $S_\infty D_{d_1} D_{d_2} D_{d_3} \cdots$ as it can be replaced by its equivalent $D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}}$.

An infinite product of matrices $S_\infty (D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})(D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}}) \cdots$ represents ∞ and, of course, we cannot compute any absolute precision at all. Any other product may be given in the form $S_\infty (D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})^e D_{d_1} D_{d_2} D_{d_3} \cdots$,

where e is a finite number and $(d_1, d_2, d_3) \neq (\frac{1}{\phi}, -\frac{1}{\phi}, -\frac{1}{\phi})$. Such a product will represent a finite number and we will be able to yield the number of digits required for any absolute accuracy, as shown in the proposition below. Note that $(D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})^e$ represents the exponent in $S_\infty (D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})^e D_{d_1} D_{d_2} D_{d_3} \cdots$, like, for example, D_{b-1}^e in $S_+ D_{b-1}^e D_{d_1} D_{d_2} D_{d_3} \cdots$.

Proposition 10. *Let $S_\infty (D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})^e D_{d_1} D_{d_2} D_{d_3} \mathfrak{D}_c^n$ be a finite product of matrices such that $(d_1, d_2, d_3) \neq (\frac{1}{\phi}, -\frac{1}{\phi}, -\frac{1}{\phi})$ and $(d_1, d_2, d_3) \neq (-\frac{1}{\phi}, \frac{1}{\phi}, \frac{1}{\phi})$. Then:*

$$\tfrac{1}{2}\phi^{3e-n+1} < \mathsf{width}(\mathbf{Info}(S_\infty (D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})^e D_{d_1} D_{d_2} D_{d_3} \mathfrak{D}_c^n)) < \phi^{3e-n+6} \ .$$

Proof. As $(D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})^e = \mathfrak{D}_0^{3e}$, using Proposition 1 we have:

$$\begin{aligned}
\mathbf{Info}(S_\infty (D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})^e D_{d_1} D_{d_2} D_{d_3}) &= \mathbf{Info}(S_\infty \mathfrak{D}_0^{3e} \mathfrak{D}_{c'}^3) \\
&= (S_\infty \mathfrak{D}_{c'}^{3e+3})[0, \infty] \\
&= \begin{pmatrix} \phi^{3e+3} & \phi^{3e+3} \\ -1-c' & 1-c' \end{pmatrix} [0, \infty] \\
&= \left[\tfrac{\phi^{3e+3}}{1-c'}, \ \tfrac{\phi^{3e+3}}{-1-c'} \right] \ ,
\end{aligned}$$

where $c' = \sum_{i=1}^3 d_i \phi^{3-i}$. Because $(d_1, d_2, d_3) \neq (\frac{1}{\phi}, -\frac{1}{\phi}, -\frac{1}{\phi})$ and $(d_1, d_2, d_3) \neq (-\frac{1}{\phi}, \frac{1}{\phi}, \frac{1}{\phi})$, it is easy to check that:

$$\frac{2}{\phi} \leq |c'| \leq 2\phi \ ,$$

as the six remaining combinations for (d_1, d_2, d_3) give $c' \in \{\pm\frac{2}{\phi}, \pm 2, \pm 2\phi\}$. Then,

$$\mathsf{width} \left[\frac{\phi^{3e+3}}{1-c'} \ , \ \frac{\phi^{3e+3}}{-1-c'} \right] = \frac{2\phi^{3e+3}}{(c')^2 - 1} \ .$$

Since the function $x \mapsto \frac{1}{x-1}$ is monotonically decreasing for $x > 1$, we have:

$$\begin{aligned}
\mathsf{width}(\mathbf{Info}(S_\infty (D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})^e D_{d_1} D_{d_2} D_{d_3})) &= \tfrac{2\phi^{3e+3}}{(c')^2-1} \\
&\leq \tfrac{2\phi^{3e+3}}{\frac{4}{\phi^2}-1} \\
&< \tfrac{2\phi^{3e+3}}{\frac{2}{\phi^3}} \\
&= \phi^{3e+6} \ ,
\end{aligned}$$

$$\begin{aligned}
\mathsf{width}(\mathbf{Info}(S_\infty (D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})^e D_{d_1} D_{d_2} D_{d_3})) &= \tfrac{2\phi^{3e+3}}{(c')^2-1} \\
&\geq \tfrac{2\phi^{3e+3}}{4\phi^2-1} \\
&> \tfrac{2\phi^{3e+3}}{4\phi^2} \\
&= \tfrac{1}{2}\phi^{3e+1} \ .
\end{aligned}$$

The subsequent n digit matrices contract the information of the product by the factor $\frac{1}{\phi^n}$, proving the claim. □

The above result can be used to compute the number of digits sufficient to obtain the required precision. Computing $3e+3+n$ digits, where

$$n = \left\lceil \frac{\log \phi^{3e+6} + k}{\log \phi} \right\rceil \ ,$$

we will yield the required decimal accuracy 10^{-k}:

$$\begin{aligned} & n = \left\lceil \tfrac{\log \phi^{3e+6}+k}{\log \phi} \right\rceil \ , \\ \Rightarrow \quad & n > \tfrac{\log \phi^{3e+6}+k}{\log \phi} \ , \\ \Rightarrow \quad & -k > \log \phi^{3e+6} - n \log \phi \ , \\ \Rightarrow \ & 10^{-k} > \phi^{3e-n+6} \ , \\ \Rightarrow \ & 10^{-k} > \mathsf{width}(\mathbf{Info}(S_\infty (D_{\frac{1}{\phi}} D_{-\frac{1}{\phi}} D_{-\frac{1}{\phi}})^e D_{d_1} D_{d_2} D_{d_3} \mathfrak{D}_c^n)) \ . \end{aligned}$$

4 The Base Interval $[-1, 1]$

Compared with $[0, \infty]$, the base interval $[-1, 1]$ has an obvious disadvantage in the amount of effort required to calculate the information of a matrix $M = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$: $\mathbf{Info}M = [\frac{c-a}{d-b}, \frac{c+a}{d+b}]$ if $\det M > 0$ or $\mathbf{Info}M = [\frac{c+a}{d+b}, \frac{c-a}{d-b}]$ if $\det M < 0$. Furthermore, testing the refining property is more complex with $[-1, 1]$ as the base interval ($|M(-1)| = |\frac{c-a}{d-b}| \leq 1$ and $|M(1)| = |\frac{c+a}{d+b}| \leq 1$), comparing it with $[0, \infty]$ as the base interval (a, b, c and d are of the same sign).

Despite the above drawbacks, it seems that taking $[-1, 1]$ as the base interval pays off at later stages. This is because the digit matrices, given below, the representation of functions by tensors, emission and finally absorption, [4,12], are simpler due to the appearance of more zeros in the representation. Pros and cons are discussed in more detail in [7].

The four possible sign matrices are given as follows:

$$\begin{aligned} S^0 &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathrm{Id} \ , & \mathbf{Info}(S^0) &= [-1, 1] \ , \\ S^1 &= \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \ , & \mathbf{Info}(S^1) &= [0, \infty] \ , \\ S^2 &= \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \ , & \mathbf{Info}(S^2) &= [1, -1] \ , \\ S^3 &= \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \ , & \mathbf{Info}(S^3) &= [\infty, 0] \ . \end{aligned}$$

The indices may be considered as exponents as $S^i S^j = S^t$, where $t = (i + j) \bmod 4$. Digit matrices in base b are given by:

$$A_d = \begin{pmatrix} 1 & d \\ 0 & b \end{pmatrix} \ ,$$

for $d \in \mathbb{Z}(b)$. The matrix A_d maps $[-1, 1]$ onto the interval $[\frac{d-1}{b}, \frac{d+1}{b}]$ with length $2/b$. A product of digits $A_{d_1} A_{d_2} \dots A_{d_n}$ corresponds to an interval $[r - \frac{1}{b^n}, r + \frac{1}{b^n}] \subseteq [-1, 1]$, where $r = \sum_{i=1}^{n} d_i b^{-i}$.

When $b = 2$ we get three matrices ($d = -1, 0, 1$):

$$A_{-1} = \begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix} \ , \qquad \mathbf{Info}(A_{-1}) = [-1, 0] \ ,$$

$$A_0 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \ , \qquad \mathbf{Info}(A_0) = [-\tfrac{1}{2}, -\tfrac{1}{2}] \ ,$$

$$A_1 = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix} \ , \qquad \mathbf{Info}(A_1) = [0, 1] \ .$$

Then, the product of digits $A_{d_1} A_{d_2} \dots A_{d_n}$ corresponds to the well–known signed binary representation of numbers in $[-1, 1]$.

It is easy to check that there exists an isomorphism between the representations with base intervals $[0, \infty]$ and $[-1, 1]$ in the following sense. Every EFP with base interval $[-1, 1]$ can be easily translated into an EFP with base $[0, \infty]$:

$$\begin{aligned} S^i A_{d_1} \dots A_{d_n} &= S^i (S^3 S^1) A_{d_1} (S^3 S^1) \dots (S^3 S^1) A_{d_n} (S^3 S^1) \\ &= (S^i S^3)(S^1 A_{d_1} S^3)(S^1 \dots S^3)(S^1 A_{d_n} S^3) S^1 \\ &= S^{i+3} D_{d_1} \dots D_{d_n} S^1 \ . \end{aligned}$$

We used the identities $S^3 S^1 = S^0 = \mathrm{Id}$ and $D_{d_i} = S^1 A_{d_i} S^3$. As $S^1[-1, 1] = [0, \infty]$ we have that

$$\begin{aligned} \mathbf{Info}_{[-1,1]}(S^i A_{d_1} \dots A_{d_n}) &= (S^i A_{d_1} \dots A_{d_n})[-1, 1] \\ &= S^{i+3} D_{d_1} \dots D_{d_n} S^1 [-1, 1] \\ &= \mathbf{Info}_{[0,\infty]}(S^{i+3} D_{d_1} \dots D_{d_n}) \ . \end{aligned}$$

Therefore, trying to obtain any absolute accuracy from an EFP given by $S^i A_{d_1} A_{d_2} \dots$ with the base interval $[-1, 1]$ is equivalent to the problem of obtaining the same absolute accuracy from $S^{i+3} D_{d_1} D_{d_2} \dots$ with $[0, \infty]$ as the base interval. We have:

$$\begin{aligned} S^0 A_{d_1} A_{d_2} \dots A_{d_n} [-1, 1] &\Longleftrightarrow S_0 D_{d_1} D_{d_2} \dots D_{d_n} [0, \infty] \ , \\ S^1 A_{d_1} A_{d_2} \dots A_{d_n} [-1, 1] &\Longleftrightarrow S_+ D_{d_1} D_{d_2} \dots D_{d_n} [0, \infty] \ , \\ S^2 A_{d_1} A_{d_2} \dots A_{d_n} [-1, 1] &\Longleftrightarrow S_\infty D_{d_1} D_{d_2} \dots D_{d_n} [0, \infty] \ , \\ S^3 A_{d_1} A_{d_2} \dots A_{d_n} [-1, 1] &\Longleftrightarrow S_- D_{d_1} D_{d_2} \dots D_{d_n} [0, \infty] \ . \end{aligned}$$

4.1 Alternative Approaches with Base Interval $[-1, 1]$

We will mention two other approaches, namely the MANTISSA–EXPONENT APPROACH and the INTEGER PART–FRACTIONAL PART APPROACH. In the former, any real number $x \in \mathbb{R}$ is represented in base b as $x = b^e u$, where e is a non–negative integer and $u = \sum d_i b^{-i} \in [-1, 1]$. This corresponds to the product of matrices $\begin{pmatrix} b^e & 0 \\ 0 & 1 \end{pmatrix} A_{d_1} A_{d_2} \ldots$. Obtaining any required absolute precision in this approach is straightforward. As each of the digit matrices, A_{d_i}, in base b, contracts distances by $1/b$, the product

$$\begin{pmatrix} b^e & 0 \\ 0 & 1 \end{pmatrix} A_{d_1} A_{d_2} \ldots A_{d_{e+k}} \ ,$$

will induce an interval whose length is $2b^{-k}$, for any $k \in \mathbb{N}$.

In the integer part–fractional part approach, a real number $x \in \mathbb{R}$ is represented as $x = e + u$, where e is an integer and $u = \sum d_i b^{-i} \in [-1, 1]$. Computing an absolute precision for x is even simpler. Calculating u within the same absolute precision will solve the problem. The product of k digit matrices, $A_{d_1} A_{d_2} \ldots A_{d_k}$ will induce an interval of length $2b^{-k}$. Hence, $x = e + u$ will be calculated with error not greater than $2b^{-k}$.

Acknowledgements

I would like to thank Reinhold Heckmann and Abbas Edalat for many useful discussions.

References

1. Di Gianantonio, P.: A Golden Ratio Notation for the Real Numbers. CWI Technical Report (1991).
2. Edalat, A.: Domains for Computation in Mathematics, Physics and Exact Real Arithmetic. Bulletin of Symbolic Logic, Vol. 3 (1997).
3. Edalat, A., Heckmann R.: Computation with Real Numbers. Applied Semantics Summer School, APPSEM 2000 (2000).
4. Edalat, A., Potts, P. J.: A New Representation for Exact Real Numbers. Electronic Notes in Theoretical Computer Science, Vol. 6 (1997).
5. Errington, L., Heckmann, R.: Using the C–LFT Library (2000).
6. Gosper, R. W.: Continued Fraction Arithmetic. HAKMEM Item 101B, MIT AI Memo 239, MIT (1972).
7. Heckmann, R.: How Many Argument Digits are Needed to Produce n Result Digits? Electronic Notes in Theoretical Computer Science, Vol. 24 (2000).
8. Kelsey, T. W.: Exact Numerical Computation Via Symbolic Computation. Proceedings of Computability and Complexity in Analysis 2000, 187–197 (2000).
9. Kornerup, P., Matula, D. W.: Finite Precision Lexicographic Continued Fraction Number Systems. Proceedings of 7th IEEE Symposium on Computer Arithmetic, 207–214 (1985).

10. Menissier-Morain, V.: Arbitrary Precision Real Arithmetic: Design and Algorithms. Submitted to Journal Symbolic Computation (1996).
11. Nielsen, A., Kornerup P.: MSB–First Digit Serial Arithmetic. Journal of Univ. Comp. Science, 1(7):523–543 (1995).
12. Potts, P. J.: Exact Real Arithmetic Using Möbius Transformations. PhD Thesis, University of London, Imperial College (1998).
13. Vuillemin, J. E.: Exact Real Computer Arithmetic with Continued Fractions. IEEE Transactions on Computers, 39(8):1087–1105 (1990).

δ-Approximable Functions*

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

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

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

1 Introduction

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

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

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

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

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 187–199, 2001.

There is a strict relation between δ-uniform computability and TTE, that is, recursive analysis; indeed, for any Archimedean field the halting sets of δ-uniform BSS machines with coefficients in $\mathbf{T}$ (the field of Turing computable reals) or $\mathbf{Q}$ are exactly the halting sets of Type 2 Turing machines [2]. Thus, the restriction of δ-uniformity reduces the full power of the BSS model, making it closer to Turing machines.

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

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

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

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

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

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

Suitable types of analytic machines compute classes of functions identified by older computational models, such as BSS computable, Type 2 computable, and weakly (in the sense of Freund) computable functions. Since two of the main classes defined in this paper turn out to coincide with the latter two, we obtain correspondingly an equivalence with classes of functions computed by suitable analytic machines. Note however that the accent in our treatment is on *equality*, whereas in analytic machines also operations are approximated. It is particularly

interesting to note that the resulting classes are the same, as this shows once again that the main problem in making a real computation feasible is testing for exact equality, and not computing exact operations.

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

2 The Computation Models

2.1 BSS Machines

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

2.2 (Weak) Type 2 Turing Machines

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

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

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

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

These machines have been studied by Freund [8], who showed that they are more powerful than standard Type 2 machines. To be true, the model used in

the abovementioned reference has a single tape: nonetheless, it is not difficult to see that the single-tape restriction does not change the power of the model. Moreover, in [8] the function computed by weak Type 2 machines are defined using a positive notation. This could in fact make a difference, and this issue is discussed in Section 10.

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

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

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

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

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

3 Degrees of Real Numbers and Jumps

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

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

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

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

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

It should be clear that, for any nondyadic real number α, there exists exactly one set in $\rho^{-1}(\alpha)$ (which is neither finite nor cofinite), and we define the degree of α, denoted by $\deg \alpha$, as the degree of unsolvability of $\rho^{-1}(\alpha)$ [12,10,3]; moreover, we let $\deg \alpha = \mathbf{0}$ for every dyadic rational α. When the distinction is irrelevant, we shall confuse real numbers, subsets of $\mathbf{N}$ and degrees, omitting the map ρ; this will happen in particular when using real numbers as oracles, or when specifying an arbitrary real number of given degree; note that in particular it is equivalent to think of a Turing machine as using oracles $\alpha_1, \ldots, \alpha_r$ or the single oracle $\alpha_1 \vee \cdots \vee \alpha_r$.

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

4 δ-Approximable Functions

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

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

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

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

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

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

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

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

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

– uniformly δ-*approximates* f *iff*

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

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

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

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

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

5 δ-Approximability vs. Turing Computability

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

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

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

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

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

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

Conversely, let f be a function pointwise δ-approximated by a BSS machine M. It is possible for a Type 2 machine, given an integer k, to compute $\varphi_M^\delta(\boldsymbol{x})$ up to the first k fractional digits, for some $\delta \in (0, 2^{-k})$. Indeed, such a machine can clearly compute $\varphi_M^\delta(\boldsymbol{x})$ with any given precision, provided that all tests along the accepting path of $\boldsymbol{x}$ for threshold δ are strict inequalities, so all we have to do is dovetail these computations for all dyadic thresholds in $(0, 2^{-k})$ and, due to Lemma 1, one of the computations will necessarily halt. We thus obtain a sequence of dyadics (indexed by k) converging to $f(\boldsymbol{x})$ that can be fed into the machine of Proposition 1. This completes the proof. □

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

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

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

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

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

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

6 Some Properties of δ-Approximable Functions

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

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

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

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

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

Another interesting property follows from uniformity:

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

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

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

7 Separation Results

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

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

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

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

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

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

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

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

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

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

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

8 Generalizations: (Weak) Turing Closures

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

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

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

Definition 3. *Let R be an Archimedean field, and $R \subseteq F$ a field extension with F Archimedean. An element $x \in F$ is said to be* Turing over R *iff there are $n \in \mathbf{N}$, $\boldsymbol{y} \in R^n$ and a Turing (Type 2) machine M (with n input tapes) such*

that $M(\boldsymbol{y}) = x$. If every $a \in F$ is Turing over R, then $R \subseteq F$ is said to be a Turing extension *of R. A field R is* Turing closed *iff it does not have any proper Turing extension. The* Turing closure *of a field R is the intersection of all Turing closed fields containing R, and will be denoted*[3] *by $T(R)$.*

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

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

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

Proof. For each $j, k \in \mathbf{N}$ let S_{jk} be the Turing machine (with oracles $\alpha_1, \ldots, \alpha_r$) that emulates M until the j-th output cell has been changed k times and then stops. Of course, for each j there is a $k > 1$ such that S_{jk} does not stop. If we have $(\alpha_1 \vee \cdots \vee \alpha_r)'$ as an oracle, for each j we can clearly compute the least such k, emulate M up to the $(k-1)$-th cell change and output the resulting value. This gives as a result a (non-weak) Type 2 machine with oracle $(\alpha_1 \vee \cdots \vee \alpha_r)'$ that outputs α, hence the thesis. □

We can finally prove our characterization:

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

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

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

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

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

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

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

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

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

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

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

9 A Remark on Signedness

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

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

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

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

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

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

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

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

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

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

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

10 Open Problems

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

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

$$\begin{array}{ccccccc} \mathscr{A}_c^\infty & \longrightarrow & \mathscr{A}_u^\infty & \longrightarrow & \mathscr{A}_p^\infty & \longrightarrow & \mathrm{BSS} \\ \downarrow & & \downarrow & & \downarrow & & \\ \mathscr{T}_2 = \mathscr{A}_c & \longrightarrow & \mathscr{A}_u & \longrightarrow & \mathscr{A}_p = \mathscr{W}_2 & & \end{array}$$

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

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

References

1. Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc. (N.S.)*, 21:1–46, 1989.
2. Paolo Boldi and Sebastiano Vigna. δ-uniform BSS machines. *J. Complexity*, 14(2):234–256, 1998.
3. Paolo Boldi and Sebastiano Vigna. The Turing closure of an Archimedean field. *Theoret. Comput. Sci.*, 231:143–156, 2000.
4. Vasco Brattka. Personal electronic communication, 2001.
5. Vasco Brattka and Peter Hertling. Feasible real random access machines. *J. Complexity*, 14(4):490–526, 1998.
6. Thomas Chadzelek and Günter Hotz. Analytic machines. *Theoret. Comput. Sci.*, 219(1–2):151–167, 1999.
7. Abbas Edalat and Philipp Sünderhauf. A domain-theoretic approach to computability on the real line. *Theoret. Comput. Sci.*, 210(1):73–98, 1999.
8. Rudolf Freund. Real functions and numbers defined by Turing machines. *Theoret. Comput. Sci.*, 23(3):287–304, May 1983.
9. Armin Hemmerling. On approximate and algebraic computability over the real numbers. *Theoret. Comput. Sci.*, 219:185–223, 1999.
10. Ker-I Ko. Reducibilities on real numbers. *Theoret. Comput. Sci.*, 31(1–2):101–123, May 1984.
11. Joseph R. Shoenfield. *Degrees of Unsolvability*. North–Holland, Amsterdam, 1971.
12. Robert I. Soare. Recursion theory and Dedekind cuts. *Trans. Amer. Math. Soc.*, 140:271–294, 1969.
13. John V. Tucker and Jeffery I. Zucker. Computation by 'While' programs on topological partial algebras. *Theoret. Comput. Sci.*, 219(1–2):379–420, 1999.
14. Sebastiano Vigna. On the relations between distributive computability and the BSS model. *Theoret. Comput. Sci.*, 162:5–21, 1996.
15. Robert F.C. Walters. An imperative language based on distributive categories. *Math. Struct. Comp. Sci.*, 2:249–256, 1992.
16. Klaus Weihrauch. *Introduction to Computable Analysis*. Texts in Theoretical Computer Science. Springer–Verlag, 2000.

Computabilities of Fine-Continuous Functions

Takakazu Mori

Faculty of Science, Kyoto Sangyo University
Kamigamo-Motoyama Kita-ku Kyoto 603-8555, Japan
morita@cc.kyoto-su.ac.jp

Abstract. We propose a sequential-based definition of locally uniformly Fine-computable functions together with a definition of effective locally uniform convergence. This definition of computability makes some discontinuous functions, which may diverge, computable. It is proved that a locally uniformly Fine-computable function can be approximated effectively locally uniformly by a Fine-computable sequence of binary step functions on the unit interval $[0, 1)$ with respect to the Fine metric. We also introduce effective integrability for locally uniformly Fine-computable functions, and prove that Walsh-Fourier coefficients of an effectively integrable function f form a computable sequence of reals. It is also proved that $S_{2^n} f$, where $S_n f$ is the partial sum of the Walsh-Fourier series, Fine-converges effectively locally uniformly to f.
Keywords: Locally uniform Fine-computable function, Effectively integrable function, Walsh-Fourier coefficients, Walsh-Fourier Series.

1 Introduction

The definition of computable real valued functions (on $[0, 1]$, say) proposed by Grzegorczyk and Lacombe is formulated in [PR] as follows:

(i) (Sequential computability) If $\{x_n\}$ is a computable sequence of reals then $\{f(x_n)\}$ is also a computable sequence of reals.
(ii) f is effectively uniformly continuous.

Pour-El and Richard's theory of the computability structures has been extended by Yasugi, Mori and Tsujii to compact metric spaces ([MTY]) and to σ-compact metric spaces ([YMT]). The computability theory of metric spaces is also investigated by Weihrauch and others based on representations and the Type-2 machine. For compact metric spaces, the equivalence between the former definitions and the latter has been proved by Kamo and Brattka independently.

The usual computability theory of the unit interval $[0, 1]$ is obtained regarding it as the metric space $([0, 1], d_E)$, where d_E is the Euclidean metric. In this formulation, a computable function must be a continuous function in the ordinary sense, and so neither the Haar functions nor the Walsh functions can be computable although they are simple in definition and calculation.

For definitions of computable functions, it seems to be natural and necessary to require that they are continuous. If we employ a stronger metric than the

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 200–221, 2001.

Euclidean metric, then the class of all continuous functions may become larger, and we can expect that some discontinuous functions become computable with respect to the new metric.

As an example, Mori ([M]) used the Fine metric d_F and proved that the Walsh functions are uniformly computable with respect to d_F. The following figures illustrate some graphs of Walsh functions.

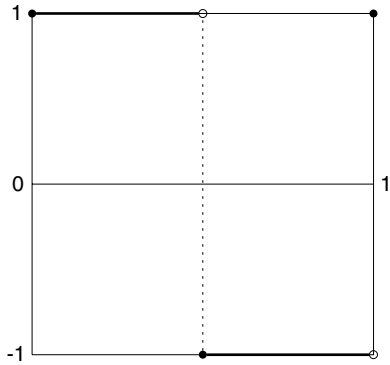


Figure 1: The graph of $W_1(x)$.

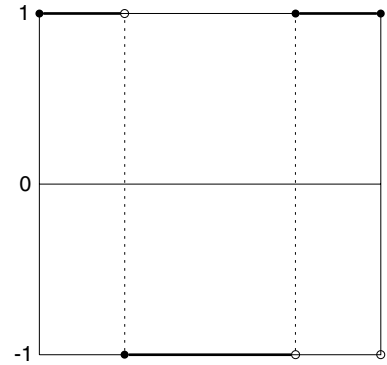


Figure 2: The graph of $W_2(x)$.

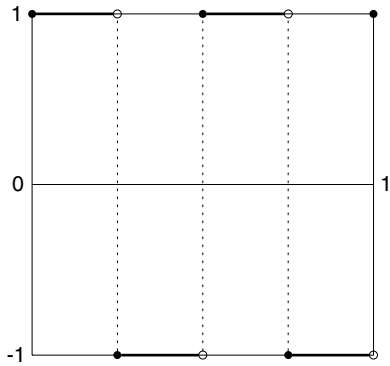


Figure 3: The graph of $W_3(x)$.

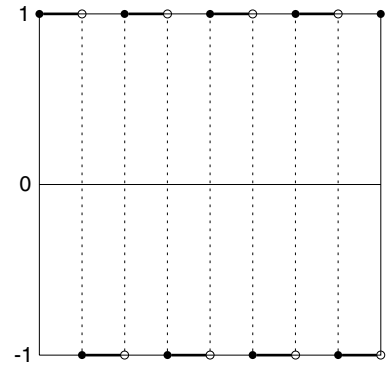


Figure 4: The graph of $W_4(x)$.

Since uniform continuity on a totally bounded metric space implies boundedness, $f(x)$ defined by

$$f(x) = \begin{cases} \frac{1}{1-2x} - 1 & \text{if } x < \frac{1}{2} \\ 0 & \text{if } x \geq \frac{1}{2} \end{cases} \tag{1}$$

is not effectively uniformly continuous. But it is still a simple function in definition and calculation. Furthermore, it is a Fine-continuous total function on $[0, 1]$.

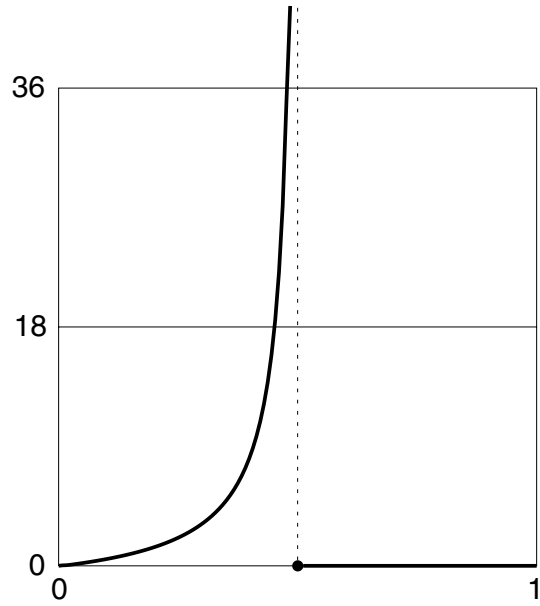


Figure 5: The graph of $f(x)$ defined by Equation (1).

One of the important problems in analysis is to investigate the behavior of functions near singularities of them and it is a usual way to change the space suitably. For example, it is practiced to take a completion or a compactification and restrict functions to those which can be extended smoothly to the new space.

To obtain a definition of computable functions, which includes $f(x)$ defined by Equation (1), effective uniform continuity must be replaced by a weaker condition of effective continuity. We also need a weaker concept of effective convergence of functions, which preserves the new continuity to develop analysis.

We propose effective locally uniform Fine-continuity as such a continuity notion and effective locally uniform Fine-convergence as the corresponding notion of convergence.

The basic definitions of computable reals and of a metric space with computability structure are summarized in §2.

In §3, we deal with the dyadic field Ξ which was introduced by Fine ([F2]). Ξ is defined to be the set of all doubly infinite sequences $\boldsymbol{\sigma} = (\ldots, \sigma_{-1}, \sigma_0; \sigma_1, \sigma_2, \ldots)$ which satisfies that there exists n such that $\sigma_\ell = 0$ for $\ell \leq n$. For each pair $(\boldsymbol{\sigma}, \boldsymbol{\tau})$ consisted of elements of Ξ , the metric $d_C(\boldsymbol{\sigma}, \boldsymbol{\tau})$ is defined by $\sum_{\ell=-\infty}^{\infty} 2^{-\ell}|\sigma_\ell - \tau_\ell|$.

The metric space (Ξ, d_C) is a σ-compact metric space. On this space, we have obtained two definitions of computable functions. One is effective uniform computability. The other is effective compact uniform continuity, which was defined by Definition 4.3 of [YMT]. We prove that a function is effectively compact-uniformly computable if and only if it is approximated effectively compact-uniformly by some computable sequence of cylinder functions.

In §4, we define locally uniformly Fine-computable functions on the unit interval $[0, 1)$. This definition is obtained by requiring effective local uniform Fine-continuity instead of the effective uniform Fine-continuity. We say that a function is locally uniformly Fine-computable if it is locally uniformly computable with respect to the Fine-metric.

It is proved that a function f is locally uniformly Fine-computable, if and only if there exists a Fine-computable sequence of binary step functions which Fine-converges effectively locally uniformly to f (Theorem 4). It is also proved that Walsh-Fourier coefficients of an effectively integrable function f form a computable sequence of reals, and $S_{2^n}f$, where S_nf is the partial sums of the Walsh-Fourier series, Fine-converges effectively locally uniformly to f (Proposition 4.2).

In §5 we discuss briefly the extension of the results on $[0, 1)$ to R_+ and prove that the Gauss function and the generalized Walsh function are uniformly Fine-computability.

2 Preliminaries

In this section we summarize the definitions in [PR], [MTY] and [YMT] which we will need. We assume separability for a metric space in this article. The set of all real numbers is denoted by R.

Definition 2.1 (*Computable Sequences of Binary Rationals and Reals*).

(i) A double sequence of binary rationals $\{r_{n,m}\}$ is said to be computable if there exist recursive functions $\alpha(n, m)$ and $\beta(n, m)$ such that

$$r_{n,m} = \frac{\beta(n, m)}{2^{\alpha(n,m)}}. \tag{2}$$

(ii) A double sequence $\{x_{n,m}\}$ of reals is said to converge effectively to a sequence of reals $\{x_n\}$ if there exists a recursive function $\alpha(n, k)$ such that,

$$|x_{n,m} - x_n| \leq \frac{1}{2^k} \quad \text{for all } k \text{ and } \quad m \geq \alpha(n, k).$$

(iii) A sequence of reals, say $\{x_n\}$, is said to be computable, if there exists a computable double sequence of binary rationals $\{r_{n,m}\}$ which converges effectively to $\{x_n\}$.

A real number x is called computable if $\{x, x, \cdots\}$ is a computable sequence.

Definition 2.2 (*Effective Convergence in Metric Spaces*). Let $\langle X, d\rangle$ be a metric space, $\{x_n\}$ be a sequence from X and $\{x_{n,m}\}$ be a double sequence from X.

$\{x_{n,m}\}$ is said to converge effectively to $\{x_n\}$ (with respect to the metric d) if there exists a recursive function β such that,

$$d(x_{n,m}, x_n) \leq \frac{1}{2^k} \quad \text{for all } k \text{ and } \quad m \geq \beta(n,k).$$

Definition 2.3 (*Effectively Cauchy Sequence*). A computable sequence $\{x_n\}$ is said to be effectively Cauchy if there exists a recursive function α such that,

$$d(x_m, x_n) \leq \frac{1}{2^p} \quad \text{for all } p \text{ and } \quad m, n \geq \alpha(p).$$

Definition 2.4 (*Computability Structure*). $\mathcal{S}$ will be called a computability structure on $\langle X, d\rangle$ if it satisfies the three axioms below. A sequence in $\mathcal{S}$ is called a computable sequence (relative to $\mathcal{S}$).

Axiom M1 (*Metrics*). If $\{x_n\}, \{y_m\} \in \mathcal{S}$, then $\{d(x_n, y_m)\}_{n,m}$ forms a computable double sequence of reals.

Axiom M2 (*Reenumerations*). If $\{x_n\} \in \mathcal{S}$, then $\{x_{\alpha(n)}\} \in \mathcal{S}$ for any recursive function $\alpha(n)$.

Axiom M3 (*Limits*). If $\{x_{n,m}\} \in \mathcal{S}$, $\{x_n\} \subset X$ and $\{x_{n,m}\}$ converges effectively to $\{x_n\}$, then $\{x_n\} \in \mathcal{S}$.

Definition 2.5 (*Effective Separability*). $\langle X, d, \mathcal{S}\rangle$ is said to be effectively separable if there exists a sequence $\{e_n\}$ in $\mathcal{S}$ which is dense in X.

Remark 2.1. We call a computable sequence $\{e_n\}$, which satisfies the requirement of Definition 2.5, an effective separating set. From the effective density lemma (Proposition 8 of [MTY]), any computable sequence which is dense in X becomes an effective separating set.

We define $B_d(a, r) = \{x | d(a, x) < r\}$.

Definition 2.6 (*Effective Compactness*). Let $\langle X, d, \mathcal{S}\rangle$ be an effective separable metric space with an effective separating set $\{e_n\}$.

(i) X is said to be effectively totally bounded if there exists a recursive function $\alpha(p)$ such that

$$X = \bigcup_{n=1}^{\alpha(p)} B_d(e_n, \frac{1}{2^p}) \quad \text{for all } \quad p. \tag{3}$$

(ii) We say that $\langle X, d, \mathcal{S}\rangle$ is effectively compact if it is effectively totally bounded and complete.

We define computable functions, computable sequences of functions and effective convergence of functions on an effectively compact metric space. We use the term function as a mapping from some metric space to R with the Euclidean metric. Therefore the convergence of the sequence $\{f(x_n)\}$ is the ordinary convergence as a sequence of reals.

Definition 2.7 (*Uniformly Computable Sequence of Functions*). A sequence $\{f_n\}$ of functions from X to R is said to be uniformly computable if

(i) (Sequential computability) the double sequence $\{f_n(x_m)\}$ is computable for any $\{x_m\} \in \mathcal{S}$

and

(ii) (Effective uniform continuity) there exists a recursive function $\alpha(n,k)$ such that, for all n,k and all $x,y \in X$,

$$d(x,y) \leq \frac{1}{2^{\alpha(n,k)}} \text{ implies } |f_n(x) - f_n(y)| \leq \frac{1}{2^k}.$$

A function f is said to be uniformly computable if $\{f,f,f,\ldots\}$ is an uniformly computable sequence.

Definition 2.8 (*Effective Uniform Convergence of Functions*). A sequence of functions $\{f_n\}$ is said to converge effectively uniformly to a function f if there exists a recursive function $\alpha(k)$ such that, for all n and k,

$$n \geq \alpha(k) \text{ implies } |f_n(x) - f(x)| \leq \frac{1}{2^k}.$$

Definition 2.9 (*r.e. Open Sets and Co-r.e. Closed Sets*). A sequence of open subsets of X, say $\{U_n\}$ is said to be sequentially recursively enumerable (r.e. open) if there are a double sequence $\{a_{n,m}\}$ in $\mathcal{S}$ and a computable double sequence of positive rationals $\{r_{n,m}\}$ such that $U_n = \bigcup_m B_d(a_{n,m}, r_{n,m})$. We call such a sequence $\{U_n\}$ r. e. open.

A sequence of closed sets $\{V_n\}$ is said to be sequentially co-r.e. if $\{X - V_n\}$ is sequentially r.e. open. Such a sequence $\{V_n\}$ is called co-r.e. closed.

An open set U is called r.e. open if $\{U,U,\cdots\}$ is r.e. open. Similarly, a closed set V is called co-r.e. closed if $\{V,V,\cdots\}$ is co-r.e. closed.

We define effectively σ-compact metric spaces, compact-uniform computable functions, compact-uniform computable sequences of functions and effective compact-uniform convergence of functions.

Definition 2.10 (*Effectively Compact Subset*). Let $\langle X,d,\mathcal{S}\rangle$ be a complete metric space with a computability structure. A co-r.e. closed subset $K \subset X$ is called effectively compact if there exist a computable sequence $\{\xi_i\}_{i=1}^{\infty}$ of points and a recursive function $\alpha(p)$ such that $\overline{\{\xi_i\}_{i=1}^{\infty}} = K$ and $\bigcup_{i=1}^{\alpha(p)} \mathbf{B}(\xi_i, \frac{1}{2^p}) \supset K$ for all positive integer p.

Definition 2.11 (*Effectively σ-Compact Metric Space*). A complete metric space $\langle X, d, \mathcal{S}\rangle$ is called effectively σ-compact if there exist a sequence $\{K_n\}_{n=1}^{\infty}$ of effectively compact subsets, a computable double sequence $\{\xi_{n,i}\}_{n,i}$ and a recursive function $\alpha(n,p)$ such that $\overline{\{\xi_{n,i}\}_{i=1}^{\infty}} = K_n$, $\bigcup_{i=1}^{\alpha(n,p)} B_d(\xi_{n,i}, \frac{1}{2^p}) \supset K_n$ and $\bigcup_{n=1}^{\infty} K_n = X$.

Note that the definition of effective σ-compactness depends on a choice of $\{K_n\}$ and $\{\xi_{n,i}\}$. In the rest of this section, we assume that $\langle X, d, \mathcal{S}\rangle$ is an effectively σ-compact metric space with respect to some $\{K_n\}$ and $\{\xi_{n,i}\}$.

Definition 2.12 (*Compact-Uniformly Computable Sequence of Functions*). A sequence of functions $\{f_m\}$ on X is called compact-uniformly computable if

(i) (*Sequential computability*) for any $\{x_n\} \in \mathcal{S}$, the double sequence $\{f_m(x_n)\}_{m,n}$ is a computable double sequence of reals

and

(ii) (*Effective compact-uniform continuity*) there exists a recursive function $\beta(n,m,p)$ such that

$$|f_m(x) - f_m(y)| \leq \frac{1}{2^p} \quad \text{for} \quad x, y \in K_n \quad \text{and} \quad d(x,y) \leq \frac{1}{2^{\beta(n,m,p)}}$$

A function f is called compact-uniformly computable if the sequence $\{f, f, \ldots\}$ is compact-uniformly computable.

Definition 2.13 (*Effective Compact-Uniform Convergence*). A sequence $\{f_m\}$ of functions on X is said to converge effectively compact-uniformly to a function f if there exists a recursive function $\alpha(n,p)$ such that

$$m \geq \alpha(n,p) \quad \text{implies} \quad |f_m(x) - f(x)| \leq \frac{1}{2^p} \quad \text{for} \quad x \in K_n.$$

The following proposition follows from the above two definitions easily.

Proposition 2.1. If a compact-uniformly computable sequence of functions $\{f_m\}$ converges effectively compact-uniformly to f, then f is compact-uniformly computable.

3 Dyadic Field and Fine Metric

Fine defined the dyadic field and treated the generalized Walsh function ([F2]). Let Ξ be the set of all doubly infinite sequences $\boldsymbol{\sigma} = (\ldots, \sigma_{-2}, \sigma_{-1}, \sigma_0; \sigma_1, \sigma_2, \ldots)$ such that each element of $\{\sigma_\ell \,|\, \ell \leq 0\}$ is zero except for finitely many ℓ's.

The metrics d_C on Ξ, which is also introduced by Fine, is defined by

$$d_C(\boldsymbol{\sigma}, \boldsymbol{\tau}) = \sum_{k=-\infty}^{\infty} \frac{|\sigma_k - \tau_k|}{2^k} \tag{4}$$

The lower limit of the summation of the above equation is finite although it may depend on $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$.

Lemma 3.1. *If $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ satisfies $d_C(\boldsymbol{\sigma}, \boldsymbol{\tau}) < \frac{1}{2^k}$, then for $\ell \leq k$ $\sigma_\ell = \tau_\ell$. On the other hand, if $\sigma_\ell = \tau_\ell$ for $\ell \leq k$, then $d_C(\boldsymbol{\sigma}, \boldsymbol{\tau}) \leq \frac{1}{2^k}$.*

Let us denote the set of all nonnegative reals by R_+. Then the mappings $\varphi : \Xi \longrightarrow R_+$ and $\psi : R_+ \to \Xi$ are defined as follows;

$$\varphi(\boldsymbol{\sigma}) = \sum_{\ell=-\infty}^{\infty} \frac{\sigma_\ell}{2^\ell}, \tag{5}$$

$$\psi(x) = (\ldots, 0, \sigma_{-k}, \ldots, \sigma_0; \sigma_1, \sigma_2, \ldots),$$

where, $(\sigma_{-k}, \ldots, \sigma_0)$ $(\sigma_{-k} \neq 0)$ is the binary expansion of the integer part $[x]$ of x and $(\sigma_1, \sigma_2, \ldots) = \psi(x - [x])$ is the binary expansion of $x - [x]$ with the convention that finitely many 1's for a binary rational are used. If $\boldsymbol{\sigma} = (\ldots, \sigma_{-k}, \ldots, \sigma_0; \sigma_1, \sigma_2, \ldots)$ and $\sigma_\ell = 0$ for $\forall \ell < -k$, we call $(\sigma_{-k}, \ldots, \sigma_0)$ or $[\varphi(\boldsymbol{\sigma})]$ the integral part of $\boldsymbol{\sigma}$. Similarly, we call $(\sigma_1, \sigma_2, \ldots)$ or $\varphi(\boldsymbol{\sigma}) - [\varphi(\boldsymbol{\sigma})]$ the decimal part of $\boldsymbol{\sigma}$.

We use the following notation:

$$\begin{aligned}
\Xi_n &= \{\boldsymbol{\sigma} = (\sigma_\ell) \in \Xi \,|\, \sigma_\ell = 0 \text{ for } \forall \ell \leq n\} \\
d_{C,n} &= \text{the restriction of } d_C \text{ to } \Xi_n \\
\varphi_n &= \text{the restriction of } \varphi \text{ to } \Xi_n \\
\psi_n &= \text{the restriction of } \psi \text{ to } [0, 2^{-n}) \\
\Xi^0 &= \{\boldsymbol{\sigma} = (\sigma_\ell) \in \Xi \,|\, \sigma_\ell = 1 \text{ for finitely many } \ell\text{'s}\} \\
\Xi^1 &= \{\boldsymbol{\sigma} = (\sigma_\ell) \in \Xi \,|\, \sigma_\ell = 0 \text{ for finitely many } \ell\text{'s}\} \\
Q^2 &= \text{the set of all nonnegative binary rationals} \\
Q^2_n &= \text{the set of all nonnegative binary rationals less than } 2^{-n} \\
\Xi^{0,p} &= \{\boldsymbol{\sigma} = (\sigma_\ell) \in \Xi \,|\, \sigma_\ell = 0 \text{ for } \forall \ell \geq p\} \\
\Xi^0_n &= \Xi^0 \cap \Xi_n \\
\Xi^{0,p}_n &= \Xi^{0,p} \cap \Xi_n \\
\Xi^1_n &= \Xi^1 \cap \Xi_n
\end{aligned}$$

For $\boldsymbol{\sigma} \in \Xi^0$, we define $\boldsymbol{\sigma}^*$ to be the element $\boldsymbol{\tau} \in \Xi^1$ such that $\varphi(\boldsymbol{\sigma}) = \varphi(\boldsymbol{\tau})$.

Ξ_n is a compact subset and $\Xi = \cup_{\ell=n}^{\infty} \Xi_{-\ell}$, so (Ξ, d_C) is a σ-compact metric space. $(\Xi_0, d_{C,0})$ is identical with (Ω, d_C), which was treated in [M].

By the definitions, $\cup_p \Xi^{0,p} = \Xi^0$, $\varphi(\Xi^{0,0})$ is the set of all nonnegative integers, $\varphi(\Xi^0_{-n})$ is the set of all binary rationals which is less than 2^n and $\varphi(\Xi^{0,p}_{-n}) = \{\frac{\ell}{2^p} \,|\, 0 \leq j < 2^{n+p}\}$.

On Ξ, two operations $\oplus$ and $\otimes$ are defined as follows:

$$(\boldsymbol{\sigma} \oplus \boldsymbol{\tau})_k = \sigma_k + \tau_k \pmod 2,$$

$$(\boldsymbol{\sigma} \otimes \boldsymbol{\tau})_k = \sum_{i+j=k} \sigma_i \tau_j \pmod 2.$$

The following equation is an easy consequence of the definitions.

$$d_C(\boldsymbol{\sigma}, \boldsymbol{\tau}) = \varphi(\boldsymbol{\sigma} \oplus \boldsymbol{\tau}) \tag{6}$$

Definition 3.1 (*Computability Structure on* (Ξ, d_C)).
$\mathcal{S}_C = \{\{\boldsymbol{\sigma}_n\} \mid \exists \alpha(n,k), \beta(n,k), \gamma(n)$ recursive functions such that
$\boldsymbol{\sigma}_n = (\ldots, 0, \beta(n,\gamma(n)), \beta(n,\gamma(n)-1), \ldots, \beta(n,0); \alpha(n,1), \alpha(n,2), \ldots)\}$

Let $\{\xi_{n,i}\}$ be an effective enumeration of all elements of Ξ_n^0, then $\langle \Xi, d_C, \mathcal{S}_C \rangle$ becomes an effectively σ-compact metric space with respect to $\{\Xi_{-n}\}$ and $\{\xi_{-n,i}\}$. From the definition of Ξ and that of the operations $\oplus$ and $\otimes$, $(\boldsymbol{\sigma} \oplus \boldsymbol{\tau})_k$ and $(\boldsymbol{\sigma} \otimes \boldsymbol{\tau})_k$ are determined by the finite bits of $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. Therefore, we obtain the following proposition.

Proposition 3.1. $\oplus$ and $\otimes$ are computable mappings from $\langle \Xi, d_C, \mathcal{S}_C \rangle \times \langle \Xi, d_C, \mathcal{S}_C \rangle$ to $\langle \Xi, d_C, \mathcal{S}_C \rangle$.

Definition 3.2 (*Cylinder Functions*). (i) A function f on Ξ_n is called a cylinder function if there exists an integer $m > n$ such that $\sigma_\ell = \tau_\ell$ for all $\ell \leq m$ implies $f(\boldsymbol{\sigma}) = f(\boldsymbol{\tau})$.

(ii) A function f on Ξ is called a cylinder function if there exists an integer n such that the restriction of f to Ξ_n is a cylinder function on Ξ_n and $f(\boldsymbol{\sigma}) = 0$ for every $\boldsymbol{\sigma} \notin \Xi_n$.

If a function f satisfies the requirement of the Definition 3.2 (i) then f takes at most 2^{m-n} values.

For each triplet of integers (n, m, j), where $n < m$, let

$$\mathit{\Gamma}_{n,m,j} = \{\boldsymbol{\sigma} \in \Xi_n \mid \sum_{\ell=n+1}^{m} 2^{m-\ell}\sigma_\ell = j\}.$$

Then, f is a computable cylinder function on Ξ if and only if there exist positive integers $n < m$ and a finite sequence of computable reals $c_0, \ldots, c_{2^{m-n}-1}$ such that $f(\boldsymbol{\sigma}) = c_j$ if $\boldsymbol{\sigma} \in \mathit{\Gamma}_{n,m,j}$ and $f(\boldsymbol{\sigma}) = 0$ for every $\boldsymbol{\sigma} \notin \Xi_n$.

Proposition 3.2 (Computability of Cylinder Functions). *Every cylinder function which takes only computable values is a uniformly computable function.*

Definition 3.3 (*Computable Sequence of Cylinder Functions*). (i) A sequence of cylinder functions $\{f_m\}$ on Ξ_n is said to be computable if there exist a recursive functions $\alpha(m)$ and a computable double sequence of reals $\{c_{m,j}\}$ such

that

$$f_m(\boldsymbol{\sigma}) = c_{m,j} \text{ if } \boldsymbol{\sigma} \in \Gamma_{n,\alpha(m),j} \text{ and } 0 \leq j < 2^{\alpha(m)-n}.$$

(ii) A sequence of cylinder functions $\{f_m\}$ on Ξ is said to be computable if there exist a recursive functions $\alpha(m)$, an integer valued recursive function $\beta(m)$ and a computable double sequence of reals $\{c_{m,j}\}$ such that

$$f_m(\boldsymbol{\sigma}) = \begin{cases} c_{m,j} \text{ if } \boldsymbol{\sigma} \in \Gamma_{\beta(m),\alpha(m),j} \text{ and } 0 \leq j < 2^{\alpha(m)-\beta(m)}. \\ 0 \quad \text{if } \boldsymbol{\sigma} \notin \Xi_{\beta(m)} \end{cases}$$

Let $d_{C,n}$ be the restriction of d_C to Ξ_n and $\mathcal{S}_{C,n}$ be the set of all computable sequences which are consisted of elements of Ξ_n. Then $\langle \Xi_n, d_{C,n}, \mathcal{S}_{C,n} \rangle$ is an effectively compact metric space with respect to $\{\xi_{n,i}\}_i$, where $\{\xi_{n,i}\}$ is an effective enumeration of all elements of Ξ_n^0.

Theorem 1 (Necessary and Sufficient Condition for Uniformly Computable Functions on Ξ_n). *A function f on Ξ_n is uniformly computable if and only if there exists a computable sequence of cylinder functions $\{f_m\}$ on Ξ_n which converges effectively uniformly to f.* (Theorem 1 of [M])

Theorem 2 (Necessary and Sufficient Condition for Compact-Uniformly Computable Functions on Ξ). *A function f is compact-uniformly computable if and only if there exists a computable sequence of cylinder functions $\{f_m\}$ which converges effectively compact-uniformly to f.*

Proof. The if part is an easy consequence of the definitions. To prove the only if part, for each n, let $\{f_{n,m}\}$ be the approximating computable sequence of cylinder functions on Ξ_{-n} obtained by the previous theorem, which converges effectively uniformly to the restriction of f. We denote the modulus of convergence by $\alpha_n(p)$. We construct an approximate sequence by a 'patching method' as follows;

$$f_1(\boldsymbol{\sigma}) = f_{1,1}(\boldsymbol{\sigma})$$
$$f_m(\boldsymbol{\sigma}) = \begin{cases} f_{1,m}(\boldsymbol{\sigma}) \text{ if } \boldsymbol{\sigma} \in \Xi_{-1} \\ f_{\ell,m}(\boldsymbol{\sigma}) \text{ if } \boldsymbol{\sigma} \in \Xi_{-\ell} \backslash \Xi_{-\ell+1}, \ 2 \leq \ell \leq m \\ 0 \quad \text{if } \boldsymbol{\sigma} \notin \Xi_{-m} \end{cases}$$

The modulus of convergence $\alpha(n,p)$ can be taken as $\max\{n, \alpha_1(p), \ldots, \alpha_n(p)\}$. □

Example 3.1 (Generalized Walsh Functions). The Walsh functions $w_n(x)$ on $\Omega = \{0,1\}^{\{1,2,\ldots\}}$ are defined by

$$w_n(\boldsymbol{\sigma}) = (-1)^{\sum_{i=0}^{k} \sigma_{i+1} n_i}, \tag{7}$$

where, $n = n_0 + 2n_1 + \cdots + 2^k n_k$ $(n_k \neq 0)$ is the binary representation of n.

The generalized Walsh functions $w_{\boldsymbol{\tau}}(\boldsymbol{\sigma})$ on Ξ are defined by

$$w_{\boldsymbol{\tau}}(\boldsymbol{\sigma}) = (-1)^{\sum\limits_{i+j=1}^{\sigma_j \tau_i}}. \tag{8}$$

If we take

$$\tau_\ell = \begin{cases} n_{-\ell} \text{ if } -k \le \ell \le 0 \\ 0 \quad \text{otherwise} \end{cases},$$

then $\varphi(\boldsymbol{\tau}) = n$ and the right-hand side of Equation (8) is equal to that of Equation (7). Therefore, $w_{\boldsymbol{\tau}}(\boldsymbol{\sigma}) = w_{\varphi(\boldsymbol{\tau})}((\sigma_1, \sigma_2, \ldots))$. Usually, this equation is regarded as the definition of $w_n(\boldsymbol{\sigma})$. Furthermore, it is well known that $w_{\boldsymbol{\tau}}(\boldsymbol{\sigma}) = w_1(\boldsymbol{\sigma} \otimes \boldsymbol{\tau})$.

We define the Fine metric on the positive reals.

Definition 3.4. For every pair of nonnegative real numbers x and y, the Fine metric $d_F(x, y)$ is defined by

$$d_F(x, y) = d_C(\psi(x), \psi(y)). \tag{9}$$

The restriction of d_F to $[0, 2^{-n})$ is denoted by $d_{F,n}$.

The following inequality is an easy consequence of the definitions.

$$|x - y| \le d_F(x, y). \tag{10}$$

Example 3.2. For a binary rational $r = \frac{j}{2^k}$ $(j > 0)$, let $x_n = r - \frac{1}{2^{k+n}}$. Then, $\{x_n\}$ is an effective Cauchy sequence with respect to the Fine metric and $d_{F,0}(r, x_n) \ge \frac{1}{2^{k+1}}$. On the other hand, $|r - x_n| = \frac{1}{2^{k+n}} \longrightarrow 0$. If $\{x_n\}$ Fine-converges to x, then x must be r from Inequality (10). Therefore, $\{x_n\}$ does not Fine converge. In this case, $\{\psi_0(x_n)\}$ converges to $\psi_0(r)^*$.

This example shows that $([0, 1), d_{F,0})$ is not complete.

We define Fine-computable cequences and Fine-computability structure $\mathcal{S}_F$ on R_+ in the same way as Definition 2.1.

Definition 3.5 (*Fine-Computability Structure*).
(i) A double sequence $\{x_{n,m}\}$ of nonnegative reals is said to Fine-converge effectively to a sequence of nonnegative reals $\{x_n\}$ if there exists a recursive function $\alpha(n, k)$ such that,

$$d_F(x_{n,m}, x_n) \le \frac{1}{2^k} \text{ for all } k \text{ and } m \ge \alpha(n, k).$$

(ii) A sequence of nonnegative reals $\{x_n\}$ is said to be Fine-computable, if there exists a computable double sequence of nonnegative binary rationals $\{r_{n,m}\}$ which Fine-converges effectively to $\{x_n\}$.

(iii) The Fine-computability structure $\mathcal{S}_F$ is defined to be the set of all Fine-computable sequences of nonnegative reals.

(iv) The computability structure $\mathcal{S}_{F,n}$ is defined to be the set of all Fine-computable sequences consisted of reals in $[0, 2^{-n})$.

In the rest of this section, we investigate computable functions on the metric space with a computability structure $\langle [0, 2^{-n}), d_{F,n}, \mathcal{S}_{F,n} \rangle$. We had considered $[0, 1]$ instead of $[0, 1)$ in [M], because $\varphi((1, 1, \ldots)) = 1$. In this case, 1 is an isolated point.

$\mathcal{S}_{F,n}$ is essentially equal to $\mathcal{S}_{C,n}$ in the following sense.

Proposition 3.3. (i) *If* $\{\boldsymbol{\sigma}_n\} \in \mathcal{S}_{C,0}$ *and* $\{\boldsymbol{\sigma}_n\}$ *does not contain* $(\ldots, 0, 0; \underbrace{0, \ldots 0}_{n}, 1, 1, \ldots)$, *then* $\{\varphi_n(\boldsymbol{\sigma}_n)\} \in \mathcal{S}_{F,n}$.

(ii) *If* $\{x_m\} \in \mathcal{S}_{F,n}$, *then* $\{\psi_0(x_m)\} \in \mathcal{S}_{C,n}$. (Proposition 3.1 of [M].)

The metric space with a computability structure $\langle [0, 1), d_{F,0}, \mathcal{S}_{F,0} \rangle$ is effectively separable and effectively totally bounded. But it is not complete by Example 3.2. We denote by $\{e_i\}$ an effective enumeration of Q_0^2 and take it as an effective separating set unless otherwise stated.

We say that a function f on $[0, 2^{-n})$ is Fine-continuous if it is continuous with respect to the metric $d_{F,n}$. For Fine-continuity, the following two propositions are well known.

Proposition 3.4 (Necessary and Sufficient Condition for Fine-Continuity). *A function* f *on* $[0, 2^{-n})$ *is Fine-continuous if and only if it satisfies the following two conditions.*

(i) f *is continuous at* $x \notin Q_n^2$.

(ii) f *is right continuous at* $x \in Q_n^2$.

Proposition 3.5. *For a Fine-continuous function* f *on* $[0, 2^{-n})$ *the following conditions are equivalent.*

(i) f *has left limits at* $x \in Q_n^2$.

(ii) *There exists a continuous function* g *on* $(\Xi_n, d_{C,n})$ *such that* $f(x) = g(\psi_n(x))$ *for* $x \in [0, 2^{-n})$.

(iii) f *is uniformly Fine-continuous on* $[0, 2^{-n})$.

We say that a function on $[0, 2^{-n})$ is uniformly Fine-computable if it is uniformly computable with respect to the Fine metric $d_{F,n}$. In [M], the uniformly Fine-computable functions were treated. We first summarize the results. A left-closed and right open interval with binary rational endpoints is said to be a dyadic interval.

Definition 3.6 (*Binary Step Functions*). A function f on $[0, 2^{-n})$ is called a binary step function if there exist dyadic intervals $I_1, \ldots, I_k$ such that they

are mutually disjoint, the union of them is $[0, 2^{-n})$ and f is constant on each interval I_ℓ.

Definition 3.7 (*Fine-Computable Sequences of Binary Step Functions*). A sequence of functions $\{f_m\}$ on $[0, 2^{-n})$ is said to be a Fine-computable sequence of binary step functions if there exist a recursive function $\alpha(m)$ and a computable double double sequence of reals $\{s_{m,j}\}$ such that

$$f_m(x) = s_{m,j} \ \textit{if} \ \ x \in \Delta_{\alpha(m),j} \ \textit{and} \ \ 0 \le j < 2^{\alpha(m)}, \tag{11}$$

where, $\Delta_{k,j} = [\frac{j}{2^k}, \frac{j+1}{2^k})$ for $0 \le j < 2^k$.

Theorem 3 (Necessary and Sufficient Condition for Uniformly Fine-Computable Function). *For a function f on $[0, 2^{-n})$, the following conditions are equivalent.*

(i) *f is a uniformly Fine-computable function.*

(ii) *There exists a computable function on $(\Xi_n, d_{C,n})$ such that $f(x) = g(\psi_n(x))$ for all $x \in [0, 2^{-n})$.*

(iii) *There exists a Fine-computable sequence of binary step functions which Fine-converges effectively uniformly to f.*

The equivalence of (i) and (ii) is a consequence of Theorem 2 of [M] and that of (i) and (iii) is a consequence of Theorem 3 of [M].

Remark 3.1. The metric space $(\Xi_0, d_{C,0})$ can be thought as a completion of the metric space $([0, 1), d_{F,0})$. Let $\mathcal{D}$ be the class of all uniformly Fine-continuous functions on $[0, 1)$, then it is easy to prove that

$$\sup_{x\in[0,1)} |f(x)| = \sup_{\boldsymbol{\sigma}\in\Xi_0} |g(\boldsymbol{\sigma})|$$

for every $f \in \mathcal{D}$ and a continuous function g on Ξ_0, which satisfies $f(x) = g(\psi_0(x))$ for $x \in [0, 1)$. Since $(\Xi_0, d_{C,0})$ is a compact metric space $\{\sup_{\boldsymbol{\sigma}\in\Xi_0} |g_n(\boldsymbol{\sigma})|\}$ is a computable sequence of reals for every uniformly computable sequence $\{g_n\}$ ([MTY]). Let $||f||_{\mathcal{D}} = \sup_{x\in[0,1)} |f(x)|$ and $\mathcal{S}_{\mathcal{D}}$ be the set of all uniformly Fine-computable sequences of functions on $[0, 1)$. Then $\langle \mathcal{D}, ||\ ||_{\mathcal{D}}, \mathcal{S}_{\mathcal{D}} \rangle$ becomes a Banach space with a computability structure in the sense of Pour-El and Richards ([PR]).

Example 3.3. Walsh functions $\{W_n(x)\}$ on $[0, 1)$ are defined by

$$W_n(x) = w_n(\psi_0(x))$$

and they form a Fine-computable sequence of binary step functions.

4 Locally Uniformly Fine-Computable Functions on the Unit Interval $[0, 1)$

In this section, we consider locally uniformly Fine-computable functions on the metric space with computability structure $\langle [0, 1), d_{F,0}, \mathcal{S}_{F,0} \rangle$.

Let $B_{F,0}(a,r)$ be the set $\{x \mid d_{F,0}(x,a) < r\} \cap [0,1)$. In general, $B_{F,0}(a,r)$ is not an interval. For example, $B_{F,0}(\frac{1}{2},\frac{3}{4}) = [0,\frac{1}{4}) \cup [\frac{1}{2},1)$. If a is a binary rational, say $\frac{j}{2^k}$, then a sufficient condition for $B_{F,0}(a,r)$ being an interval is that $r = \frac{1}{2^\ell}$ and $\ell \geq k$. In this case, $B_{F,0}(\frac{j}{2^k},\frac{1}{2^\ell}) = [\frac{j}{2^k},\frac{j}{2^k}+\frac{1}{2^\ell})$ and $B_{F,0}(a,r) = \{x \mid d_{F,0}(x,a) < r\}$. Therefore, we adapt the following constraint for $B_{F,0}(a,r)$.

Constraint: a is a binary rational $\frac{j}{2^k}$ and $r = \frac{1}{2^\ell}$ for some $\ell \geq k$.

Since any dyadic interval in $[0,1)$ can be represented as a finite disjoint union of constrained $B_{F,0}(a,r)$'s, this constraint does not cause a loss of generality.

This constraint yields also that $B_{F,0}(a,r) = \{x \mid d_{F,0}(x,a) < r\} = a + [0,2^{-\ell})$, where $a + [0,2^{-\ell}) = \{a + x \mid x \in [0,2^{-\ell})\}$.

Definition 4.1 (*Locally Uniformly Fine-Computable Sequences of Functions on* $[0,1)$). A sequence of functions $\{f_n\}$ on $[0,1)$ is said to be locally uniformly Fine-computable if

(i) f_n is sequentially computable

and

(ii) there exist a recursive function $\gamma(n,i)$ and a recursive function $\alpha(n,i,k)$ such that

$$\bigcup_{i=1}^{\infty} B_{F,0}(e_i, 2^{-\gamma(n,i)}) = [0,1) \quad \text{for each} \quad n$$

$$|f_n(x) - f_n(y)| \leq \frac{1}{2^k} \quad \text{for} \quad x,y \in B_{F,0}(e_i, 2^{-\gamma(n,i)}) \quad \text{and} \quad d_{F,0}(x,y) \leq \frac{1}{2^{\alpha(n,i,k)}}.$$

A function is called locally uniformly Fine-computable if the sequence $\{f,f,\ldots\}$ is computable.

Definition 4.2 (*Effective Locally Uniform Convergence on* $[0,1)$). A sequence of functions $\{f_n\}$ on X is said to Fine-converge effectively locally uniformly to a function f if there exist a recursive function $\gamma(i)$ and a recursive function $\beta(i,k)$ such that $\{f_n\}$ is computable sequence with $\gamma(n,i) = \gamma(i)$,

$$\bigcup_{i=1}^{\infty} B_d(e_i, 2^{-\gamma(i)}) = [0,1)$$

and

$$|f_n(x) - f(x)| \leq \frac{1}{2^k} \quad \text{for} \quad x \in B_{F,0}(e_i, 2^{-\gamma(i)}) \quad \text{and} \quad n \geq \beta(i,k).$$

From these definitions, the following proposition is obtained easily.

Proposition 4.1. *If a locally uniformly Fine-computable sequence of functions* $\{f_m\}$ *Fine-converges effectively locally uniformly to* f*, then* f *is locally uniformly Fine-computable.*

We can define the notion of locally uniform computability and locally uniform convergence on an effectively separable metric space with a computability structure in the same way.

Example 4.1. Let $f(x)$ be the function defined by Equation (1), that is, $f(x) = \frac{1}{1-2x} - 1$ if $x < \frac{1}{2}$ and $f(x) = 0$ if $x \geq \frac{1}{2}$. Then f diverges at $\frac{1}{2}$ from the left-hand side. Therefore, f is not a uniformly Fine-continuous function.

If e_i is not equal to zero, let $e_i = \frac{p_i}{2^{q_i}}$, where p_i is an odd integer smaller than $2^{q_i} - 1$.

Case (i) ($e_i = 0$). Take $\gamma(i) = 2$ and $\alpha(i,k) = k + 2$.

Case (ii) ($p_i \geq 2^{q_i - 1}$). Take $\gamma(i) = q_i$ and $\alpha(i,k) = 1$.

Case (iii) ($p_i \leq 2^{q_i-1} - 2$). Take also $\gamma(i) = q_i$. Since $f'(e_i + 2^{-\gamma(i)}) = \frac{2^{2q_i+1}}{(2^{q_i} - 2(p_i+1))^2}$, it is sufficient to take $\alpha(i,k) = k + 2q_i + 1 - 2\log_2(2^{q_i} - 2(p_i+1))$.

Final Case ($p_i = 2^{q_i-1} - 1$). In this case, $e_i = \frac{1}{2} - \frac{1}{2^{q_i}}$. Take $\gamma(i) = q_i + 1$, then $f'(e_i + 2^{-\gamma(i)}) = 2^{2q_i+1}$. It is sufficient to take $\alpha(i,k) = k + 2q_i + 1$.

Therefore, f is locally uniformly Fine-computable. It diverges at $\frac{1}{2}$, and also the improper integral $\displaystyle\int_0^1 f(x)dx$ diverges.

Example 4.2. Let $f(x)$ be the function defined in the previous Example. Define

$$f_1(x) = f(x)$$
$$f_n(x) = \begin{cases} f(x) & \text{if } x < \frac{1}{2} \\ f_{n-1}(2x-1) & \text{if } x \geq \frac{1}{2} \end{cases} \quad \text{for} \quad n \geq 2,$$

then $\{f_n(x)\}$ is a locally uniformly Fine-computable sequence of functions.

Example 4.3. Let $g(x)$ be

$$g(x) = \begin{cases} \frac{1}{2^n(1-2x)-1} & \text{if } 1 - \frac{1}{2^{n-1}} \leq x < 1 - \frac{1}{2^n} \\ 0 & \text{if } x = 1 \end{cases}.$$

The sequence $\{f_n\}$ in the previous example Fine-converges effectively locally uniformly to g. $g(x)$ is also locally uniformly Fine-computable.

Theorem 4 (Necessary and Sufficient Condition for Locally Uniformly Fine-Computable Function). *A function f on $[0,1)$ is locally uniformly Fine-computable if and only if there exists a Fine-computable sequence of binary step functions which Fine-converges effectively locally uniformly to f.*

Proof. Let $f(x)$ be a locally uniformly Fine-computable function with respect to $\gamma(i)$ and $\alpha(i,k)$. We write $B_{F,0}(e_i, 2^{-\gamma(i)})$ as B_i in this proof.

If we define $\mathcal{S}_i = \{\, \{e_i + x_k\} \,|\, \{x_k\} \in \mathcal{S}_{F,2^{-\gamma(i)}} \}$ and d_i to be the restriction of d_F to B_i, then $\langle B_i, \mathcal{S}_i, d_i \rangle$ is an effectively separable metric space with a computability structure. Fine-computable sequences of binary step functions

on B_i are defined by translations of Fine-computable sequences of binary step functions on $[0, 2^{-\gamma(i)})$.

From (ii) of Definition 4.1, the restriction of $f(x)$ to B_i is effectively uniformly Fine-continuous with respect to $\alpha(i,k)$ for each i. There exists a Fine-computable sequence of binary step functions $\{f_{i,m}(x)\}$ which Fine-converges effectively uniformly to $f(x)$ on B_i by Theorem 3. By the definition of effective uniform Fine-convergence (Definition 2.8), there exists a recursive function $\beta_i(k)$ such that

$$|f_{i,m}(x) - f(x)| \leq \frac{1}{2^k} \text{ if } x \in B_i \text{ and } m \geq \beta_i(k).$$

We construct an approximating Fine-computable sequence of binary step functions $f_m(x)$ by the 'patching method', which was used in the proof of Theorem 2.

$$f_1(x) = f_{1,1}(x).$$

$$f_2(x) = \begin{cases} f_{1,2}(x) \text{ if } x \in B_1 \\ f_{2,2}(x) \text{ if } x \notin B_1 \end{cases}.$$

$$f_m(x) = \begin{cases} f_{1,m}(x) \text{ if } x \in B_1 \\ f_{k,m}(x) \text{ if } x \in B_k \backslash \bigcup_{\ell=1}^{k-1} B_\ell, 2 \leq k \leq m \\ 0 \qquad \text{if } x \notin \bigcup_{\ell=1}^{m} B_\ell \end{cases}.$$

As stated in the beginning of this section, B_i is $[e_i, e_i + 2^{-\gamma(i)})$. The set of all finite disjoint union of binary intervals, which are left-closed and right-open, make a Boolean algebra, that is, it is closed under complement and finite union. Therefore, $\{f_m(x)\}$ just constructed is a Fine-computable sequence of binary step functions.

If we define $\delta(i,k) = \max\{i, \beta_1(k), \ldots, \beta_i(k)\}$, then $\{f_m\}$ Fine-converges locally effectively to f with respect to B_i and $\delta(i,k)$. □

Remark 4.1. Let $\Sigma = \{0, 1, *\}^\omega$. For $\boldsymbol{\sigma} = (\ldots, 0, \sigma_{-k}, \ldots, \sigma_0; \sigma_1, \sigma_2, \ldots) \in \Xi \backslash \Xi^1$, we define $\boldsymbol{\tau} = (\sigma_{-k}, \ldots, \sigma_0, *, \sigma_1, \sigma_2, \ldots) \in \Sigma$ and $\rho_F(\boldsymbol{\tau}) = \varphi(\boldsymbol{\sigma})$. Then, ρ_F is a representation of R_+ in the sense of Weihrauch, and ρ_F-computability of real numbers is equivalent to Fine-computability. In the same way, we obtain a representation $\rho_{F,n}$ of $[0, 2^n)$. Brattka has noted that the function defined by Equation (1) is $(\rho_{F,0}, \rho_E)$ computable, where ρ_E is some admissible standard representation of the real numbers.

We consider the Walsh-Fourier series. The Walsh-Fourier coefficients $\{c_k\}$ are defined by

$$c_k = \int_0^1 W_k(x) f(x) dx, \tag{12}$$

and the Walsh-Fourier series $\{S_n f\}$ is defined by

$$S_n f(x) = \sum_{k=0}^{n} c_k W_k(x). \tag{13}$$

Our aim is to obtain an extension of Proposition 4.5 of [M] to locally uniformly Fine-computable functions.

The crucial fact in the proof of the computability of the Walsh-Fourier coefficients $\{c_m\}$ of f is that $\int_0^1 |f_n(x) - f(x)|dx$ converges effectively to 0 for a uniformly Fine-computable sequence of binary step functions $\{f_n\}$ which Fine-converges effectively uniformly to f.

The proof of the effective convergence of $\{S_{2^n} f\}$ is based on the following lemma.

Lemma 4.1. *If f is integrable and Fine-continuous then*

$$S_{2^n} f(x) - f(x) = 2^n \int_0^{2^{-n}} (f(x \oplus t) - f(x))dt, \tag{14}$$

holds, where $x \oplus t = \varphi_0(\psi_0(x) \oplus \psi_0(t))$ and dt is the Lebesgue measure on $[0, 1)$.

Every uniformly Fine-computable function f on $[0, 1)$ is bounded and integrable. But, Example 4.1 shows that there exists a locally uniformly Fine-computable function which diverges and the integral on $[0, 1)$ also diverges. Therefore, we need an additional condition concerning improper integrals.

Definition 4.3 (*Effective Integrability*). Let f be a locally uniformly Fine-computable function with respect to $\gamma(i)$ and $\alpha(i, k)$. f is said to be effectively integrable if $\int_{E_n} |f|dx$ is an effective Cauchy sequence, where

$E_n = \bigcup_{i=1}^{n} B_{F,0}(e_i, 2^{-\gamma(i)})$.

Remark 4.2. If a locally uniformly Fine-computable function f is effectively integrable, then $|f|$ is integrable and $\int_{E_n} |f|dx$ converges effectively to $\int_0^1 |f|dx$.

Remark 4.3. A sufficient condition for effective integrability is that there exists an approximating Fine-computable sequence of binary step functions $f_m(x)$ such that $\int_0^1 |f_m(x)|dx$ forms an effective Cauchy sequence of reals.

We obtain the following proposition as a locally uniformly Fine-computable version of Proposition 4.5 in [M].

Proposition 4.2 (Convergence of $S_{2^n} f$ to f for Locally Uniformly Fine-Computable Functions). *If f is a locally uniformly Fine-computable function and effectively integrable, then it holds that*

(i) *the Walsh-Fourier coefficients* $\{c_k\}$ *form a computable sequences of reals and* $\{S_n f\}$ *is a Fine-computable sequence of binary step functions*

and

(ii) $\{S_{2^n} f\}$ *Fine-converges effectively locally uniformly to* f.

Proof. Let f be effectively integrable locally uniformly Fine-computable function with respect to $\gamma(i)$ and $\alpha(i,k)$. We define $B_i = B_{F,0}(e_i, 2^{-\gamma(i)})$ and $E_n = \bigcup_{i=1}^{n} B_i$ as in Definition 4.3. From Theorem 3, there exists an approximating Fine-computable sequence of binary step functions $\{f_m\}$. We can assume that $f_m(x) = 0$ outside E_m.

Computability of $\{c_k\}$: Note that the integrals $\int_{[0,1)\setminus E_n} |f| dx$ converges effectively to 0 and $|W_k(x)| = 1$. Let $c_{k,n} = \int_{E_n} W_k(x) f(x) dx$, then $\{c_{k,n}\}$ converges effectively to $\{c_k\}$ for each k by the effective integrability. If we also define $c_{k,n,m} = \int_{E_n} W_k(x) f_m(x) dx$, then $\{c_{k,n,m}\}$ converges effectively to $\{c_{k,n}\}$ for each pair of k and n by Proposition 4.5 of [M]. Therefore, $\{c_k\}$ is a computable sequence of reals.

Convergence of $\{S_{2^n} f\}$: Note that $d_{F,0}(x \oplus t, x) = t$ for $x, t \in [0,1)$ by Equations (6) and (9). Let $\beta(i,k) = 2^{\max\{\alpha(i,k),\gamma(i)\}+1}$. If $n \geq \beta(i,k)$ and $0 \leq t \leq 2^{-n}$, then $d_{F,0}(x \oplus t, x) \leq \dfrac{1}{2^{\alpha(i,k)}}$ and $x \oplus t \in B_i$ for all $x \in B_i$. These imply that $|f(x \oplus t) - f(x)| \leq \dfrac{1}{2^k}$. Using Equation (14) in Lemma 4.1, it follows that $|S_{2^n} f(x) - f(x)| \leq \dfrac{1}{2^k}$. □

There exist simple functions which are not Fine-continuous.

Example 4.4. Let $h(x)$ be

$$h(x) = \begin{cases} \frac{1}{|x - \frac{1}{2}|} - 1 & \text{if } x \neq \frac{1}{2} \\ 0 & \text{if } x = \frac{1}{2} \end{cases}.$$

h has not both limits at $x = \frac{1}{2}$. Therefore, h is not Fine-continuous. But it is continuous except $\frac{1}{2}$ with respect to the Euclidean metric.

In the rest of this section, we discuss about the computability of the function h in Example 4.4.

It seems natural that h is considered as a function on $D = [0,1]\backslash\{\frac{1}{2}\}$ with respect to the metric d_D, which is the restriction of the Euclidean metric d_E to D.

Let $K_n = [0, \frac{1}{2} - \frac{1}{n}] \cup [\frac{1}{2} + \frac{1}{n}, 1]$ and $\{\xi_{n,i}\}$ be the effective enumeration of all rationals in K_n. Then, D is effectively σ-compact with respect to $\{K_n\}$ and $\{\xi_{n,i}\}$ and h is compact-uniformly computable.

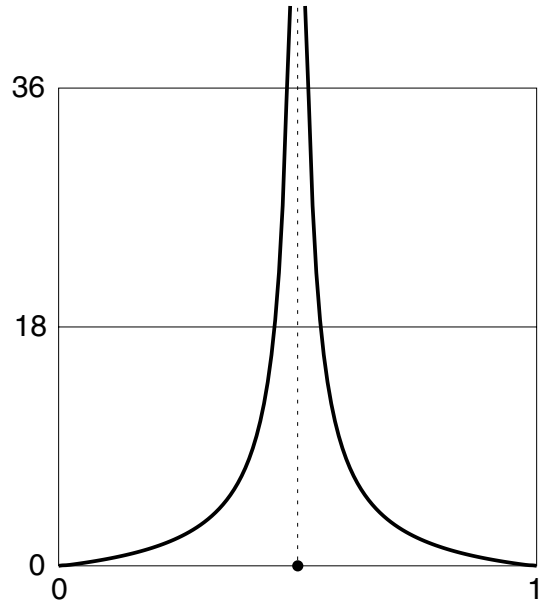


Figure 6: The graph of $h(x)$ in Example 4.4.

On the other hand, the notion of effective locally uniform computability can be defined with respect to d_D. In this case, we need no constraint on $B_D(a,r) = \{y \,|\, d_E(x,y) < r\} \cap D$ and$\gamma(i)$ is replaced by by a computable sequence of rationals $\{r_i\}$ in Definition 4.1. Then, h is an effectively locally uniformly computable function.

It seems that compact-uniformly computability and effective locally uniform computability are equivalent. But we have not succeeded in proving the equivalence between them nor found any counter example.

With respect to the Fine-metric, we cannot define the concept of compact–uniform computability, since the space D is not σ-compact.

In order to handle computabilities of a class of functions which include $h(x)$ in Example 4.4, it seems that we need to construct a theory of computable partial functions.

5 Computable Functions on R_+

In this section, we treat briefly computable functions on the metric space with computability structure $\langle R_+, d_F, \mathcal{S}_F\rangle$, where $\mathcal{S}_F$ is defined by Definition 3.5.

In the space $\langle R_+, d_F, \mathcal{S}_F\rangle$, we have two definitions of computability of functions. Both require sequential computability. Considering continuity, uniform Fine-computability requires effective uniform Fine-continuity and locally uniform Fine-computability requires effective locally uniform Fine-continuity. The definition of the latter is obtained by modifying Definition 4.1.

Proposition 5.1. If f is sequentially computable and Lipschitz continuous, that is, there exists c such that $|f(x) - f(y)| \leq c|x - y|$, then f is uniformly Fine-computable.

In this case, f is also uniformly computable with respect to d_E.

Definition 5.1 (Fine-Computable Sequence of Binary Step Functions). A sequence $\{f_n\}$ of functions on R_+ is said to be a Fine-computable sequence of binary step functions if there exist a recursive function $\alpha(n)$, an integer valued computable function $\beta(n)$ and computable double sequence $\{s_{n,j}\}$ of real numbers such that

$$f_n(x) = \begin{cases} s_{n,j} \text{ if } x \in \Delta_{\alpha(n),j},\ 0 \leq j < 2^{\alpha(n)-\beta(n)} \\ 0 \quad \text{if } x \notin [0, 2^{-\beta(n)}) \end{cases}$$

The following two theorems can be proved similarly to Theorems 3 and 4.

.

Theorem 5. *A function f is uniformly Fine-computable if and only if one of the following conditions is satisfied.*

(i) *There exists a Fine-computable sequence of binary step functions which Fine-converges effectively uniformly to f.*

(ii) *There exists a uniformly d_C-computable function g on Ξ such that $f(x) = g(\psi(x))$ for $\forall x \in R_+$.*

Theorem 6. *f is locally uniformly Fine-computable if and only if there exists a Fine-computable sequence of binary step functions which Fine-converges effectively locally uniformly to f.*

If $x \in [0, 2^n)$ then $\{y \mid d_F(x, y) < 2^n\} = [0, 2^n)$. Thus, we obtain the following proposition.

Proposition 5.2. Let g be a compact-uniformly computable function on Ξ. Then, a function on R_+, which satisfies that $f(x) = g(\psi(x))$ for all $x \in R_+$, is locally uniformly Fine-computable.

Example 5.1 (Gauss Function). On (R_+, d_F), the Gauss function $[x]$ is uniformly Fine-computable.

For $\Xi \ni \boldsymbol{\sigma} = (\ldots, \sigma_{-k}, \ldots, \sigma_0; \sigma_1, \sigma_2, \ldots)$, we define $g(\boldsymbol{\sigma}) = \sum_{\ell=0}^{k} \sigma_{-\ell} 2^{\ell}$.

If $\{\boldsymbol{\sigma}_n\}$ is a computable sequence in Ξ (definition 3.1), then

$$g(\boldsymbol{\sigma}_n) = \sum_{\ell=0}^{\gamma(n)} \beta(n, \ell) 2^{\ell}$$

is a computable sequence of nonnegative integers.

On the other hand, if $d_C(\boldsymbol{\sigma}, \boldsymbol{\tau}) < 1$ then $\sigma_\ell = \tau_\ell$ for $\ell \leq 0$ and $g(\boldsymbol{\sigma}) = g(\boldsymbol{\tau})$. This proves effective uniform continuity of g. Therefore g is a uniformly computable function on $\langle \Xi, d_C, \mathcal{S}_C \rangle$. It is obvious that $[x] = g(\psi(x))$ for $x \in R_+$. □

Example 5.2 (Generalized Walsh Functions). The generalized Walsh functions $W_y(x)$ on R_+ are defined by

$$W_y(x) = w_{\psi(y)}(\psi(x)).$$

The following formulas are well known.

$$W_n(x) = W_n(x - [x])$$
$$W_y(x) = W_1(x \otimes y)$$

where, $x \otimes y$ is defined to be $\varphi(\psi(x) \otimes \psi(y))$.

From the computability of $\otimes$ (Proposition 3.1), the following proposition follows easily.

Proposition 5.3 (Computability of Generalized Walsh Function).
(i) *$w_{\boldsymbol{\tau}}(\boldsymbol{\sigma})$ is a d_C uniformly computable function if $\boldsymbol{\tau}$ is computable.*
(ii) *$w_{\boldsymbol{\tau}}(\boldsymbol{\sigma})$ is a computable function on $\langle \Xi, d_C, \mathcal{S}_C \rangle^2$.*
(iii) *$W_y(x)$ is a uniformly Fine-computable function on $\langle R_+, d_F, \mathcal{S}_F \rangle$ if y is a Fine-computable real.*
(iv) *$W_y(x)$ is a uniformly Fine-computable function on $\langle R_+, d_F, \mathcal{S}_F \rangle^2$.*

Acknowledgements

The Author would like to express his gratitude to the referees and the editors, according to whose advices this article has been revised, and also Yasugi and Tsujii for their discussions.

References

[B] V. Brattka, *Some Notes on Fine Computability*, unpublished note.
[E] Y. Endo, *Walsh Analysis* (in Japanese), Tokyo Denki Univ. Press, 1993.
[F1] N.J. Fine, *On the Walsh Functions*, Trans. Amer. Math. Soc., 65(1949), 372-414.
[F2] N.J. Fine, *The generalized Walsh functions*, Trans. Amer. Math. Soc., 69(1950), 66-77.
[GES] B. Golubov, A. Efimov, and V. Skvortsov, *Walsh Series and Transforms*, Kluwer Academic, 1991.
[M] T. Mori. *On the computability of Walsh functions*, to appear in TCS.
[MTY] T. Mori, Y. Tsujii, and M. Yasugi, *Computability Structures on Metric Spaces*, Combinatorics, Complexity and Logic (Proceedings of DMTCS'96), ed. by Bridges et al., Springer(1996), 351-362.
[PR] M.B. Pour-El and J.I. Richards, *Computability in Analysis and Physics*, Springer-Verlag, 1989.
[SWS] F. Schipp, W.R. Wade, and P. Simon, *Walsh Series*, Adam Hilger, 1990.
[TYM] Y. Tsujii, M. Yasugi, and T. Mori, *Uniform topological space and computability of some discontinuous functions*, manuscript.
[W] K. Weihrauch, *Computable Analysis*, Springer-Verlag, 2000.
[YBT] M. Yasugi, V. Brattka, and M. Washihara, *Computability properties of the Gaussian function*, preprint.
[YMT] M. Yasugi, T. Mori, and Y. Tsujii, *Effective Properties of Sets and Functions in Metric Spaces with Computability Structure*, TCS 219(1999), 467-486.
[Y] K. Yoneda, *An application of Walsh Fourier Analysis to Weakly Stationary Processes*, Acta Math. Hungar., 76(1997), 303-335.

The iRRAM: Exact Arithmetic in C++

Norbert Th. Müller

Abteilung Informatik — Universität Trier
D-54286 Trier, Germany
mueller@uni-trier.de

Abstract. The iRRAM is a very efficient C++ package for error-free real arithmetic based on the concept of a Real-RAM. Its capabilities range from ordinary arithmetic over trigonometric functions to linear algebra even with sparse matrices. We discuss the concepts and some highlights of the implementation.
Keywords: Computable Real Analysis, Random Access Machines, Limits, Interval Arithmetic, Multiple Precision Arithmetic, Multi-valued functions, C++.

1 Introduction

In the last decade, several implementations of exact real arithmetic based on different theoretical approaches or programming languages have been discussed, see e.g. [BoCa90] (essentially based on decimal representation), [EdPo97] (linear fractional transformations), or [GL00] (using multiple precision arithmetic). All these approaches lack the possibility of full imperative programming that would facilitate the implementation of the usual numerical algorithms.

In [BrHe95], Brattka and Hertling considered a different approach: a Turing machine based simulation of random access machines working with reals. The basic idea was to iterate a finite precision approximation of the RAM, but to increase the precision from iteration to iteration.

In [Mu97], the author presented a prototype implementation based on the iterative structure of this simulator: the **i**nteractive (or **i**terative) **R**eal-**RAM** (**iRRAM**). A preliminary version had already been presented at the Second Workshop on Computability and Complexity in Analysis, August, 24/25th 1996, Trier, Germany. Since then, the package has been constantly enhanced and improved. Its stability has been proven by computations taking several weeks of time, producing just a few numbers for a small table in [HN00]. At a small competition between several systems for exact arithmetic organized at the 4th Workshop on Computability and Complexity in Analysis at Swansea (CCA 2000), the iRRAM was the clear winner. The sources of the iRRAM can be found at `http://www.informatik.uni-trier.de/iRRAM/index.html`.

A program for the iRRAM is coded in ordinary C++ but may use a special class `REAL`, that behaves like real numbers without any error. Only a small set of *intrinsic* operations is allowed to use the internal structure of this class: usual arithmetic operations, tests (with a special semantic defined below), output, and conversion to/from integer or other types. Of course on top of these operations, the programmer may use (almost) all programming methods from C++, like defining own data types (like real

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 222–252, 2001.

matrices or complex numbers, which are already implemented in this way) and functions (even returning real values).

The most important extension to the original simulator from [BrHe95] was the implementation of very powerful operators for the computation of limits of user-defined sequences. The operators even allow to compute multi-valued functional limits like the complex square root [Mu98]. With their use, e.g. the full set of trigonometric functions from *sin* to *acosech* could be implemented easily, currently using a Taylor series approach. The powerful AGM methods [Bo87,Bt76,Sch90] have been used for special values like π or $\log(2)$ or for an efficient implementation of $\log(x)$.

This ability of computing limits seems to be a unique feature of the iRRAM. The limited scope of algebraic or symbolic computations on real numbers is left as soon as these operators are used. From this point on, Type 2 Theory of Effectivity (TTE) [Ko91,We95,We97] is the best fitting theoretical model. This implies e.g. that computable functions must essentially be continuous, that it is no longer possible to check whether two real numbers are equal, etc.

A certain relaxation of the 'law' that computability implies continuity has been discussed e.g. in [BrHe94,BrHe95,Br96,Br99]: Instead of computing ordinary functions we may also compute set-valued functions F, e.g. $F : \mathbb{R} \to 2^{\mathbb{R}}$. Computing such a function F for an argument x means that the result of the computation of F on x must be one of the values in the set $F(x)$.

In the iRRAM the set-valued F appears like an ordinary (but *multi-valued*) partial function f, where we use the notation $f : \mathbb{R} \rightsquigarrow \mathbb{R}$ to express the multi-valuedness. Here the value $f(x)$ must be one of the elements of $F(x)$. The actual choice of the value is hidden from the user and appears to be nondeterministic: The same x may lead to different results $f(x)$ at different points of a computation.

The source code for the iRRAM is ordinary `C++`, as already said. However, some restrictions are necessary: The user should *not* use the normal IO operations, as this could lead to surprising and annoying results. Global real variables are not allowed. Dynamic allocation of memory with `malloc` should be used as rarely as possible, while the use of `alloca` should be possible without problems. The use of external libraries has to be handled with care, as they will normally not be suited for the special semantics of the iRRAM.

With these restrictions, `C++` loses a lot of its universal character, but more than enough is left to do numerical work. Figure 1 contains a simple example with some of the most important features of the implementation and should give a rough impression of the capabilities of the iRRAM. We will briefly explain this sample program in the following:

- Line 1 is the import of the necessary definitions.
- Lines 3-4 show an example of a simple self-defined real-valued function.
- Line 4 shows applications of the control structures in `C++` and of the overloading of operators that allows to build arithmetic expressions, here simply `x-y`. The function `positive(x-y,k)` is an intrinsic multi-valued test checking whether `x-y` is positive. In the interval $x-y \in (-2^k, 2^k)$, the result may be wrong, i.e. it may be `true` as well as `false`. In consequence, `maxapprox` returns a value that differs from the maximum of x and y by at most 2^k.

```
 1  #include "iRRAM.h"
 2
 3  REAL maxapprox (long k, const REAL& x, const REAL& y)
 4  { if ( positive(x-y,k) ) return x; else return y; };
 5
 6  REAL max (const REAL& x, const REAL& y)
 7  { return limit(maxapprox,x,y); };
 8
 9  void compute()
10  { REAL x1 =  3.14159, x2 = "3.14159", x3 =  pi();
11    rwrite(x1,63);          rprintf("\n");
12    rwrite(x2,63);          rprintf("\n");
13    rwrite(max(x3,x2),63);  rprintf("\n");  }
14
15  int main (int argc,char **argv)
16  { iRRAM_initialize(argc,argv); iRRAM_exec(compute); }
```

```
+.31415899999999998826183400524314492940902709960937500000E+0001
+.31415900000000000000000000000000000000000000000000000000E+0001
+.31415926535897932384626433832795028841971693993751058210E+0001
```

Fig. 1. A sample program for the iRRAM, with output.

- Lines 6-7 are an example for the limit computing capabilities of the iRRAM: The maximum of `x` and `y` is the limit of `maxapprox(k,x,y)` for $k \to -\infty$.
- Lines 9-13 define the core procedure `compute()` of this iRRAM example.
- Line 10 shows three of the possible initializations of `REAL` variables: from a double, from a string, and from a real-valued function (returning π).
- Lines 11-13 show I/O that has to be done through special functions. In this example, the values are printed with a width of exactly 63 characters including the exponent, where only an error corresponding to the rounding of the last decimal is allowed. The results are as expected: The first line of the output shows the effect of the initialization from a `C++` double. This data type uses a mantissa of 53 bits. In consequence, we get nonzero digits up to the 53rd decimal where only the first 16 are correct. The second line comes from the initialization from a string, where we get the precise intended value of the string constant. The third line shows the result for π (which is computed using an AGM iteration, see e.g. [Ba88,Bo87]).
- Lines 15-16 define `main`, which is optional. If `main` is missing, exactly this two line definition of `main` is used as default. This allows to use the iRRAM in two ways: (i) As a package for exact arithmetic with special semantics explained in the next section, and (ii) as a tool for approximative arithmetic with 'usual' semantics explained in section 13.

The paper is structured as follows: In the following two sections we give a sketch of the semantics and of the corresponding simulation. In sections 4 to 9 we will explain this simulation in more detail and with emphasis on those implementational details that have great influence on efficiency. Section 10 is devoted to the limit computing capabilities

of the iRRAM. The rest of the paper aims at applications and gives an overview of additional features of the implementation.

2 Semantics of the iRRAM

The iRRAM implements an imperative programming language for real numbers, as said in the previous section. We will give a rough idea of the theoretical background by sketching a corresponding operational semantics (without going into all the details). This greatly simplifies the explanation of the concepts of the iRRAM. The later sections will show how the semantics is realized by the implementation.

In the following, let $\mathcal{P}$ be a program for the iRRAM, or more precisely: a program $\mathcal{P}$ that is called via `iRRAM_exec`($\mathcal{P}$).

- For simplicity, we assume that input and output of $\mathcal{P}$ may each consist of two vectors of natural and real numbers, i.e. we have an *input and output space* $\mathbb{M} = \mathbb{N}^* \times \mathbb{R}^*$. Here the natural numbers might represent encoded values of other sets as long as these are denumerable.
- Further, we assume that $\mathbb{S} := \mathbb{N}^* \times \mathbb{R}^* \times \mathbb{M}$ is the set of possible *configurations* during the computation of $\mathcal{P}$. Appropriate encodings have to be used to represent the state of the program execution as well as the values of all discrete valued variables (including integers, floating point numbers, strings, pointers etc.) of $\mathcal{P}$ as natural numbers, which are given as the first component m_d of a configuration $s = (m_d, m_r, (o_d, o_r))$. The second component m_r represents the real-valued variables within the program. m_d and m_r together represent the data space. Finally, the third component consists of the discrete output o_d and the real output o_r produced during the computation. The input to $\mathcal{P}$ is not included in these configurations.
- On the configurations, the progress of the computation can be expressed via a *transition relation* $\overset{}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} \subseteq \mathbb{S}\times\mathbb{S}$ that depends on the program $\mathcal{P}$ and a given input vector $\mathcal{I} \in \mathbb{M}$, or equivalently as a (multi-valued!) transition function $\mathcal{T}_{\mathcal{I},\mathcal{P}} : \mathbb{S} \rightsquigarrow \mathbb{S}$.
  The finite iteration $\overset{n}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}}$ and the transitive closure $\overset{*}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}}$ of $\overset{}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}}$ are defined as usual, e.g. $s \overset{n}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} s'$ iff $s = s_0 \overset{}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} s_1 \overset{}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} \dots \overset{}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} s_n = s'$ for some states s_i.
  We assume that the output is append-only, i.e. it can neither be read nor rewritten. Using $\overset{}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}}$, this property can be expressed via:

$$(m_d, m_r, (o_d, o_r)) \overset{*}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} (m'_d, m'_r, (o'_d, o'_r))$$
$$\text{implies } o'_d = o_d \circ o''_d, o'_r = o_r \circ o''_r \text{ for some } o''_d, o''_r, \text{ and}$$

$$(m_d, m_r, (o_d, o_r)) \overset{*}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} (m'_d, m'_r, (o_d \circ o''_d, o_r \circ o''_r))$$
$$\text{if and only if } (m_d, m_r, (\varepsilon, \varepsilon)) \overset{*}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} (m'_d, m'_r, (o''_d, o''_r))$$

  Here ε denotes the empty string and $\circ$ denotes the concatenation of strings.
  Because of the enormous complexity of C++ and the encodings necessary to achieve the simple configuration set $\mathbb{S}$, we will *not* explicitly define $\overset{}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}}$ here, but it should

correspond to 'elementary' operations within the program, like entering or leaving loops, evaluation of (sub-)expressions, assignments, etc.
Basic operations of the iRRAM on the data type `REAL` like elementary arithmetic are also assumed to be executed within one single transition. These operations will be called *intrinsic* in the following. It is the appropriate choice of these intrinsics that determines the computational power as well as the implementability in the iRRAM. We will only use intrinsics that are computable multi-valued functions in the sense of TTE, which implies certain restrictions: For example, the iRRAM does not have a test for equality of real numbers.
The semantics of the evaluation of user-defined functions can be defined like the operational semantics of imperative languages with procedures, i.e. using the concept of a stack (that is hidden in the data space of the configurations).
In contrast, the application of a limit operator of the iRRAM should be considered similar to an intrinsic: A (user-defined) TTE-computable multi-valued function is evaluated in just one transition. In chapter 10 we will explain how this evaluation is actually done.

– We assume there is a distinct initial configuration $\mathcal{S}_0 = (\varepsilon, \varepsilon, (\varepsilon, \varepsilon))$ (which corresponds to the non-existence of global variables). Furthermore there are a set $\mathbb{S}^+$ of configurations, where the program terminates successfully, and a set $\mathbb{S}^-$, where the program fails with an error, for example due to division of a real by zero. So for $s \in \mathbb{S}^+ \cup \mathbb{S}^-$, there is no s' with $s \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} s'$.
A *computation* is either a finite sequence $\mathcal{S}_0 \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} s_1 \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} s_2 \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} \ldots s_n$ such that $s_n \in \mathbb{S}^+ \cup \mathbb{S}^-$ (called terminating or failing computations) or an infinite sequence of states $\mathcal{S}_0 \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} s_1 \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} s_2 \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} \ldots$ with $s_i \notin \mathbb{S}^+ \cup \mathbb{S}^-$ for all i.
Please note that due to the multi-valued nature of $\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}$ (and similar to nondeterministic machines), there may be many different computations for the same input $\mathcal{I}$, there may even exist computations of different types!
– Finally, the computed multi-valued function $\mathcal{S}_\mathcal{P} \colon \mathbb{M} \rightsquigarrow \mathbb{M}$ is defined by

$$\mathcal{S}_\mathcal{P}(\mathcal{I}) = \{\mathcal{O} \mid \mathcal{S}_0 \overset{*}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} (m_d, m_r, \mathcal{O}) \in \mathbb{S}^+\}$$

for any admissible $\mathcal{I}$. Here an input $\mathcal{I} \in \mathbb{M}$ is called admissible, iff $\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}$ has neither failing nor infinite computations.
So $\mathcal{S}_\mathcal{P}(\mathcal{I})$ is only defined if all possible computations are terminating, and then the value $\mathcal{S}_\mathcal{P}(\mathcal{I})$ consists of all possible results from these computations. We only consider terminating computations here to keep the model as simple as possible, although this is not required in implementation of the iRRAM.

3 Simulative Concept of the iRRAM

In the following we sketch the basic ideas of the iRRAM in a form suited to match the idealized semantics from the previous section.

In order to realize this semantics, the implementation of the iRRAM essentially simulates the real valued parts of $\mathcal{P}$ while preserving (almost) any computation on the

discrete parts. Internally, the simulation is done within the access methods to the real numbers. In consequence, there is no overhead to the discrete parts of the computations; a purely discrete program within the iRRAM would be as fast as outside of the iRRAM.

- The iRRAM uses a representation of real numbers that is based on a subset of the intervals with rational endpoints. So let $\mathbb{J} = \{(l,r) \mid l < r,\ l,r \in \mathbb{Q}\}$.
  A real number x is uniquely determined by an infinite sequence
  $$\mathcal{J} = ((l_0,r_0),(l_1,r_1),(l_2,r_2),\ldots) \in \mathbb{J}^\infty$$
  with $\lim l_i = \sup l_i = \inf r_i = \lim r_i (= x)$. In this case we write $\varrho(\mathcal{J}) := x$, so ϱ is a (partial) surjection from $\mathbb{J}^\infty$ onto $\mathbb{R}$. Here we only use that all the single intervals contain x and finally become arbitrarily small; nested intervals or a given convergence speed are not necessary. This *representation* ϱ of the real numbers can be extended in an obvious way to a surjection ϱ from $(\mathbb{J}^\infty)^*$ onto $\mathbb{R}^*$.
  Instead of the input space $\mathbb{M} = \mathbb{N}^* \times \mathbb{R}^*$, we use $\overline{\mathbb{M}} = \mathbb{N}^* \times (\mathbb{J}^\infty)^*$ in the simulation, i.e. each real number x is replaced by an (arbitrary) $\mathcal{J} \in \varrho^{-1}(x)$. We will use $\mathcal{I} \overset{I}{\sim} \overline{\mathcal{I}}$ to denote that $\mathcal{I} = (i_d, i_r)$, $\overline{\mathcal{I}} = (i_d, \overline{i_r})$, and $\varrho(\overline{i_r}) = i_r$.
  The simulation will obviously not be able to write a real number or an infinite sequence of intervals within finite time, so as an (intermediate) representation of the output we use $\overline{\mathbb{M}}^f = \mathbb{N}^* \times (\mathbb{J}^*)^*$. The relation $\overset{O}{\sim}$ between $\mathbb{M}$ and $\overline{\mathbb{M}}^f$ that corresponds to the simulation of output is more complex than $\overset{I}{\sim}$: For $\mathcal{O} = (o_d, x_1 \circ \ldots \circ x_n) \in \mathbb{M}$ and $\overline{\mathcal{O}} = (\overline{o}_d, (J_{1,1} \circ \ldots \circ J_{1,j_1}) \circ \ldots \circ (J_{\overline{n},1} \circ \ldots \circ J_{\overline{n},j_{\overline{n}}})) \in \overline{\mathbb{M}}^f$ we write $\mathcal{O} \overset{O}{\sim} \overline{\mathcal{O}}$ iff $o_d \in \mathbb{N}^*$ is a prefix of $\overline{o}_d \in \mathbb{N}^*$, $n \le \overline{n}$, and $x_k \in \bigcap_{j \le j_k} J_{k,j}$ for any $k \le n$. So all components from $\mathcal{O}$ must be present in $\overline{\mathcal{O}}$, either in an identical form for the discrete parts, or at least in a 'consistent' way for the real parts. It is important to notice that $\overline{\mathcal{O}}$ may have more components than $\mathcal{O}$.
- The configuration set of the simulation will be $\overline{\mathbb{S}} := \mathbb{N}^* \times \mathbb{J}^* \times \overline{\mathbb{M}}^f \times \mathbb{Z} \times \mathbb{N}^*$. Please note that $\overline{\mathbb{S}}$ is denumerable, in contrast to $\mathbb{S}$. So we are able to represent these configurations using ordinary data structures from C++.
  The two configuration sets are connected via a simulating relation $\overset{S}{\sim}$ with
  $$s = (m_d, x_1 \circ \ldots \circ x_n, \mathcal{O}) \quad \overset{S}{\sim} \quad \overline{s} = (\overline{m}_d, J_1 \circ \ldots \circ J_{\overline{n}}, \overline{\mathcal{O}}, \overline{p}, \overline{q})$$
  iff $m_d = \overline{m}_d$, $n = \overline{n}$, $x_i \in J_i$ for all $i \le n$, and if $\mathcal{O} \overset{O}{\sim} \overline{\mathcal{O}}$. So the simulating configuration $\overline{s}$ covers the same information m_d on the discrete part of the simulated configuration s (i.e. ordinary data structures and process state), but represents each real number by just one interval. The meaning of the *precision bound* $\overline{p}$ and the *multi-value cache* $\overline{q}$ will be explained later on.
  The initial state of the simulation will be $\overline{\mathcal{S}}_0 = (\varepsilon, \varepsilon, (\varepsilon,\varepsilon), \overline{p_0}, \varepsilon)$ (again, $\overline{p_0}$ will be explained later).
- On these simulating configurations, the progress of the computation is given by a transition relation $\overset{}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow}} \subseteq \overline{\mathbb{S}} \times \overline{\mathbb{S}}$ (with iterates $\overset{n}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow}}$ and closure $\overset{*}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow}}$).
  In contrast to the multi-valuedness of $\overset{}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}}$, this transition $\overset{}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow}}$ will be single-valued, i.e. for any given configuration $\overline{s}$, there is at most one $\overline{s}'$ with $\overline{s} \overset{}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow}} \overline{s}'$.

Again, the output $\overline{\mathcal{O}} = (\overline{o}_d, (J_{1,1} \circ \ldots \circ J_{1,j_1}) \circ \ldots \circ (J_{n,1} \circ \ldots \circ J_{n,j_n}))$ is append-only, i.e. a written natural number will be appended to $\overline{o_d}$, and a written interval will either be appended to one of the sequences $(J_{k,1} \circ \ldots \circ J_{k,j_k})$ or will start a new sequence. This is necessary to ensure that the output produced at any point during the simulation will remain valid forever.
In consequence the simulator, as implemented in C++, does not need to store any output. Although the configurations contain the output (to simplify the description), it will not lead to any noticeable overhead in the execution of the simulation.

– For inputs $\mathcal{I} \in \mathbb{M}$ and $\overline{\mathcal{I}} \in \overline{\mathbb{M}}$ with $\mathcal{I} \overset{I}{\sim} \overline{\mathcal{I}}$, the transition relations have the following two central properties:
correctness:
For any $\overline{s} \in \overline{\mathbb{S}}$ with $\overline{\mathcal{S}}_0 \overset{*}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\Longrightarrow}} \overline{s}$ there is an $s \in \mathbb{S}$ with $s \overset{S}{\sim} \overline{s}$ and $\mathcal{S}_0 \overset{*}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} s$
restricted completeness:
For any $s \overset{S}{\sim} \overline{s}$ and s' with $\mathcal{S}_0 \overset{*}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} s \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} s'$ and $\overline{\mathcal{S}}_0 \overset{*}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow}} \overline{s}$ there are $s'', \overline{s}''$ such that $s'' \overset{S}{\sim} \overline{s}''$, $s \overset{*}{\underset{\mathcal{I},\mathcal{P}}{\Longrightarrow}} s''$, and $\overline{s} \overset{*}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow}} \overline{s}''$.
It is important to note that we have a one step transition from s to s'', but it may take more time to reach $\overline{s}''$ from $\overline{s}$.
– Instead of the successful configurations $\mathbb{S}^+$, there is a set $\overline{\mathbb{S}}^\circ$ of *reiteration configurations* such that $s \overset{S}{\sim} \overline{s}$ and $s \in \mathbb{S}^+$ implies $\overline{s} \in \overline{\mathbb{S}}^\circ$.
For such a reiteration configuration, the transition relation $\overset{}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow}}$ is defined as

$$(\overline{m_d}, \overline{m_r}, \mathcal{O}, \overline{p}, \overline{q}) \underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow} (\varepsilon, \varepsilon, \mathcal{O}, \overline{p}', \overline{q})$$

with $\overline{p}' \ll \overline{p}$, i.e. the simulation is essentially restarted with an improved precision bound $\overline{p}'$. Only the multi-value cache $\overline{q}$ is saved during this transition. Of course, the yet produced output can not be deleted, but it has no influence on the continuation of the simulation.
The multi-value cache $\overline{q}$ guarantees that a restarted simulation will take the same computational path again (for details see section 9): If we have

$$\overline{\mathcal{S}}_0 \;=\; (\varepsilon, \varepsilon, (\varepsilon, \varepsilon), \overline{p_0}, \varepsilon) \quad \overset{*}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow}} \quad (\overline{m_d}, \overline{m_r}, (\overline{o_d}, \overline{o_r}), \overline{p}, \overline{q}) = \overline{s}$$

for a reiteration configuration $\overline{s}$, then the continued simulation on $\overline{s}$ will yield

$$\overline{s} \underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow} (\varepsilon, \varepsilon, (\overline{o_d}, \overline{o_r}), \overline{p}', \overline{q}) \quad \overset{*}{\underset{\overline{\mathcal{I}},\mathcal{P}}{\longrightarrow}} \quad (\overline{m_d}, \overline{m_r}', (\overline{o_d}, \overline{o_r}'), \overline{p}'', \overline{q}) = \overline{s}'$$

with $\overline{p}'' \leq \overline{p}'$ and unchanged multi-value cache $\overline{q}$. Furthermore, although $\overline{m_d}$ and $\overline{m_r}$ are deleted in the first transition after $\overline{s}$, $\overline{m_d}$ will be reconstructed afterwards, and $\overline{m_r}'$ will have the same length as $\overline{m_r}$ but will contain new (usually smaller) consistent intervals. The discrete output $\overline{o_d}$ will remain totally unchanged, and the real output $\overline{o_r}'$ will have the same number of interval sequences, but each of these will usually be extended in a consistent way.
In fact, there are many more reiteration configurations than just those simulating $\mathbb{S}^+$: As soon as the intervals in $\overline{m_r}$ are too big to allow a correct continuation of the

simulation, a reiteration is done. This will be the case e.g. if we have to compare two real values x_i and x_j, but the simulating intervals J_i and J_j have a nonempty intersection. In this case, the corresponding $\overline{s}$ will be a reiteration configuration, but the later $\overline{s}'$ (with smaller intervals) might allow to continue the simulation.
The same may hold for failing configurations $s \in \mathbb{S}^-$. If the failure is due to an error in the discrete part (like a division of a `double` by zero), a similar error will appear in the simulation, so a failure in the simulated computation may lead to a failure in the simulation. But if the failure comes from the real part, a reiteration may be executed. For example, a simulated division of a `REAL` by zero will always be a reiteration configuration, so it leads to an infinite sequence of reiterations, where none of them will be able get to get past this division operation.

– The reiterations lead to the **convergence** property of the simulation:
If $\mathcal{I}$ is admissible and $\mathcal{I} \overset{I}{\sim} \overline{\mathcal{I}}$, then there will be a terminating computation

$$\mathcal{S}_o \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} s_1 \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} \ldots \underset{\mathcal{I},\mathcal{P}}{\Longrightarrow} s_n = (m_d, m_r, (o_d, o_r)) \in \mathbb{S}^+$$

such that there is a (generally infinite) simulating computation

$$\overline{\mathcal{S}}_0 \underset{\overline{\mathcal{I}},\mathcal{P}}{\overset{*}{\longrightarrow}} \overline{s_n}^{(1)} \underset{\overline{\mathcal{I}},\mathcal{P}}{\overset{*}{\longrightarrow}} \overline{s_n}^{(2)} \ldots$$

with $s_n \overset{S}{\sim} \overline{s_n}^{(k)}$ for any k, where the output components $\overline{o_d}^{(k)}$ of $\overline{s_n}^{(k)}$ are identical to o_d, and the $\overline{o_r}^{(k)}$ converge to o_r.
So the output from the simulation will converge to one of the possible results from $\mathcal{S}_\mathcal{P}(\mathcal{I})$ by following exactly one terminating computation infinitely often. In consequence, $\mathcal{S}_\mathcal{P}$ is a computable multi-valued function in the sense of [BrHe94].

Please note that in most applications of the iRRAM, we will have discrete (i.e. non-real) output o_d only, and the real output o_r will remain empty. Usually, o_d will consist of approximations to real numbers with some finite precision. Here the simulation may stop as soon as a configuration from $\mathbb{S}^+$ is reached, as this discrete output will not be changed again. In this case, we get finite simulations and we also may use the iRRAM as a tool for ordinary approximative computations within arbitrary other programs!

4 Basic Multiple Precision Arithmetic

The backend of the iRRAM consists of an implementation of the intervals that are used to approximate the real numbers. Here multiple precision arithmetic (MP for short) is used, for which many freely available packages exist. Pioneering work in this area had been done by R.P. Brent [Bt75,Bt76,Bt78] with his MP package in `FORTRAN`. An older overview can be found in the file `BIGNUMS.TXT` [Rio94] available freely in the internet. At the moment, one of the most elaborate packages is GMP 3.1.1 (Gnu Multiple Precision, [Gr00]), using assembler routines at time critical parts.

In its first prototype from August 1996, the iRRAM was based on a small MP package called LR written by the author. Meanwhile, also GMP and the MPFR library [Zi00] can be used in different ways. As the interface to the MP backend is very simple, it would be easy to add further MP packages.

The routines for multiple precision arithmetic can be divided in two parts:

- A few low level but time critical operations that work on arbitrarily long integers (`gmp_n` in case of GMP) or on the arbitrarily long mantissa of fixed point numbers (`Dyadic_Base` in case of LR), essentially these are the 4 basic arithmetic operations (reduced to these sets) and a shifting operator.
- A higher level of about 20 routines for arbitrary MP numbers (from `gmp_f` for GMP or `Dyadic` for LR): arithmetic operations, comparison, type conversions, etc. Internally, the low level routines are applied whenever possible.

At compile time, the user can choose between four different variants of the iRRAM: GMP (using `gmp_f` and `gmp_n`), LR (using `Dyadic` and `Dyadic_base`), LRGMP (using `Dyadic` and `gmp_n`), and MPFR (using `mpfr` and `gmp_n`). This approach has the advantage that errors in the implementation can be found much easier. In fact, the use of LRGMP revealed three errors in the GMP 3.0 routines, and the interface to MPFR was very helpful in debugging MPFR.

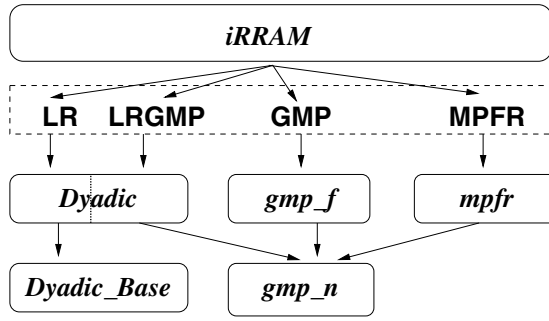


Fig. 2. The iRRAM and its backends.

One quite big conceptual difference between GMP/MPFR and LR is how the still finite precision is handled (i.e. how rounding errors are created): In GMP and MPFR, each variable c is allocated with a (changeable) amount s_c of memory. After an operation $c = a \circ b$ the variable c contains the best approximation to the exact result $a \circ b$ fitting into the allocated memory, so the single operations of GMP and MPFR work with relative precision of results and each operation introduces an error of order $c \cdot 2^{-s_c}$.

The concept of LR is different: After fixing a desired precision p, the operation $c = a \circ b$ yields a result c with $|c - a \circ b| \leq 2^p$, so the single operations work with an absolute precision independent from a, b, and '$\circ$'. In consequence, the user does not need to worry about memory allocation and can concentrate on the error propagation.

All the variants of the iRRAM implement a type `DYADIC` of floating point numbers with a variable sized mantissa and a 32 bit exponent. On the user level, the differences are hidden, the concept of precision is as in LR: The desired absolute precision can be changed dynamically using a procedure `setprec`(p). Overloading of the arithmetic operators simplifies the usage allowing e.g. assignments like `c=(a+b)/(a-b)`, where

each of the involved subexpressions delivers a result with error at most 2^p. Of course, control of the error propagation through the subexpressions must still be done manually.

In order to get access to special optimized functions of the single MP packages, the interfaces to the packages are slightly different. This has been used for example in order to get an implementation of the square root `sqrt` that is much faster than the high level version shown in section 10: E.g. the interface to the `LRGMP` backend of the iRRAM (which can be found in `LRGMP_interface.h`) contains the lines

```
#define MP_has_sqrt 1
#define MP_sqrt(z1,z,n) Dyadic_LRSQRT(&(z1),&(z),n)
```

In `REALLIB.cc`, where most of the higher level mathematical functions are implemented, there is a corresponding `#ifdef MP_has_sqrt` switching between the high level version from section 10 and a faster intrinsic version using `MP_sqrt`.

It should be mentioned that the iRRAM uses an own restricted form of memory management for the multiple precision values: The iRRAM manages a pool of variables that are already initialized, which significantly reduces the overhead for the very frequent creation and deletion of objects arising from overloaded operators in `C++`.

5 Simplified Interval Representation

In the previous chapter we presented a simple interface to multiple precision numbers. A similar concept with overloaded operators is used for the data type `REAL`. As explained in section 3, we only need to use single intervals to represent real numbers. So internally, each variable x of type `REAL` is implemented as a multiple precision number d together with information on the absolute error $e \geq |d - x|$, and in any stage of the computation of the iRRAM, we work with intervals $(d - e, d + e)$ knowing that the intended real x must be included in the interval.

The error information e could be stored in many different ways and especially in many different resolutions. We might use $\{2^{-n} \mid n \in \mathbb{N}\}$ as set of possible error informations, which could be called a 'precision of n bits'. As an other possibility, we might choose $\{z \cdot 2^p \mid z \in \mathbb{N}, p \in \mathbb{Z}\}$, so that we are able to describe errors very precise in the manner of a true interval arithmetic.

Both extremes have advantages and disadvantages: With the first approach, we would only need one additional integer to represent e (i.e. 4 byte) but we would lose at least one bit of information (concerning the mantissa of d) on any operation, demanding a very high initial precision. The second approach implies storing e as an own multiple precision number (or storing both values $d - e$ and $d + e$) allowing a very sharp control of the error propagation, but the memory consumption is doubled and operations are significantly slower.

The iRRAM implements an intermediate solution: The error is stored as a value from $Err := \{z \cdot 2^p \mid z \in \mathbb{N}, p \in \mathbb{Z} \text{ with } z < 2^{gb}, |p| < 2^{maxexp}\}$. The values of gb and $maxexp$ may depend on the CPU and the MP package in use. For usual 32 bit CPUs we have $gb = 30$ and $maxexp = 29$.

So internally, each variable x of type `REAL` is implemented as a (simplified) interval $(d \pm e)$, where d is a multiple precision number and e simply consists of two values

of type `long` to represent p and z. In consequence, each elementary operation on the intervals consists of one multiple precision operation (where the complexity increases with growing precision) and a few simple operations on integers (with fixed complexity), and we have a significant reduction in the growth of errors (compared to n-bit precision) during single operations by only a small overhead, especially in computing scalar products of vectors where all components are of similar error.

6 Controlling the Precision

Each single operation $c = a \circ b$ on `REAL` variables is transformed to the underlying intervals: If $a \sim (d_a \pm e_a)$ and $b \sim (d_b \pm e_b)$), the multiple precision number d_c of the result $c \sim (d_c \pm e_c)$ is computed from d_a and d_b with an absolute precision 2^p.

Here p essentially depends on the actual errors e_a and e_b of a and b: If e_a and/or e_b are quite large, it would be a waste of time to compute d_c very precisely. On the other hand, it will not always be advantageous to choose a very high precision, if e_a and e_b are very small: Already simple divisions with precise values (like `REAL(1)/REAL(3)`) would lead to unwanted and very high computational costs. In order to avoid this, p is not allowed to be smaller than a certain *precision bound* $\overline{p}$ already mentioned in section 3. Its value will change during the execution of a program and will be explained later in more detail.

Please note that we are working with errors of size 2^p and not 2^{-p}, so p should be larger and not smaller than $\overline{p}$! Usually, p and $\overline{p}$ will be negative.

As an example, we examine the resulting error propagation for the division a/b of reals: In order to exclude division by zero, suppose $|d_b| > 2 \cdot e_b > 0$, so $|d_b|/2 < |b|$. Suppose we compute d_c with $|d_c - d_a/d_b| \leq 2^p$ using the underlying multiple precision package, where p will be specified later.

Now consider $u_a, u_b, l_b \in Err$ approximating $|d_a|$ and $|d_b|$ as close as possible with $|d_a| < u_a$ and $l_b \leq |d_b| < u_b$. These values can be computed from d_a and d_b in constant time. With $|d_b| > 2 \cdot e_b$ we have $e_b < l_b$, so we get

$$
\begin{aligned}
|\frac{a}{b} - d_c| &\leq |\frac{a}{b} - \frac{d_a}{d_b}| + 2^p = |\frac{(a - d_a) \cdot d_b + d_a \cdot (d_b - b)}{b \cdot d_b}| + 2^p \\
&\leq \frac{e_a \cdot |d_b| + e_b \cdot |d_a|}{(|d_b| - e_b) \cdot |d_b|} + 2^p \leq \frac{e_a \cdot u_b + e_b \cdot u_a}{(l_b - e_b) \cdot l_b} + 2^p \leq z_c \cdot 2^{p_c} + 2^p
\end{aligned}
$$

where appropriate values z_c and p_c are easily computed using ordinary 32-bit integer arithmetic. As we only want error bounds, we may assume that $p_c \geq \overline{p}$.

Now simply choose $p = p_c$ leading to $|a/b - d_c| \leq (z_c{+}1) \cdot 2^{p_c} =: e_c$.

Normally, the resulting error e_c will be representable within Err. If not, it is usually sufficient to slightly increase e_c. Sometimes, e_c might be too large to have an upper bound in Err. In this case, we emit an 'overflow warning' and do a reiteration, that will be explained in detail in the next section.

7 Iterating a Computation

Essentially, all computations are done with approximations to the real numbers and each single operation is executed with finite precision (influenced by the precision bound $\overline{p}$).

Due to the accumulation of errors during a longer computation, it may happen that the error of an approximation gets too big to continue the computation (e.g. when trying to compare two numbers represented by two overlapping intervals). This is one of the points where the central concept of the iRRAM (taken from [BrHe95]) is used: *The whole computation* is repeated with a significantly better (i.e. smaller) precision bound $\overline{p}$! So instead of a single and fixed precision bound we have a sequence $\overline{p}_0 \gg \overline{p}_1 \gg \overline{p}_2 \ldots$, where a recomputation with $\overline{p}_{i+1}$ as precision bound instead of $\overline{p}_i$ will usually lead to better approximations.

The choice of these precision bounds has no influence of the correctness of the iRRAM simulation, as long as they get arbitrarily small. On the other hand, the time complexity may be quite different.

It is well known that the asymptotic complexity of sequences of iterated computations is of the same order as the complexity of the last iteration, if we have $\overline{p}_i = \overline{p}_0 - f^i$ for a fixed (e.g. rational) $f > 1$ and if all used operations and functions have a well behaved ('regular', see e.g. [Mu93]) complexity. Applications of such regular or 'smooth' complexity bounds have already been studied e.g. in [Bt76,FiSt74,CoAa89]

So in theory, the choice of $\overline{p_0}$ and f is not essential. Obviously, the starting precision as well as the increment in the iterated computations are critical points in practice. If we start with values for $\overline{p}_0$ and f that are too large, we lose computation time because we are too precise. On the other hand, small values lead to frequent reiterations.

The iRRAM uses $\overline{p}_i = \overline{p}_0 + g \cdot \frac{f^i - 1}{f - 1}$, where $\overline{p}_0 = g = -50$ and $f = 1.25$. This leads to the following precisions bounds for the first iterations:

$\overline{p}_0$	$\overline{p}_1$	$\overline{p}_2$	$\overline{p}_3$	...	$\overline{p}_{10}$	$\overline{p}_{15}$	$\overline{p}_{20}$	$\overline{p}_{25}$	...
$\overline{p}_0 = -50$	$\overline{p}_0 + g = -100$	-162	-240	...	-1703	-5502	-17096	-52477	...

$\overline{p}_0$, g, and f are not fixed at compile time, but can be changed at the start of any program to optimize its time complexity. The initial value $\overline{p}_0 = -50$ was chosen because the IEEE double precision floating point numbers have a mantissa of about 50 bits.

An important example for iterations is the division of reals: In the previous section, we used the assumption $|d_b| > 2 \cdot e_b > 0$ to exclude a division by zero. If this inequality does not hold, we may either have attempted a true division by zero or (usually) the error e_b in the approximation d_b is too large to exclude such a division. So this is a point where a reiteration with improved initial precision should be initiated.

This also implies that a division by zero initiates an infinite loop of iterations instead of an exception as in usual arithmetic. As division can not be extended to a total continuous function, this effect is unavoidable.

The same holds with the tests on the real numbers: If we compare two numbers x and y, and the corresponding intervals have a nonempty intersection, we reiterate the computation in the hope that we will get smaller intervals with empty intersection. So comparing numbers leads to an infinite loop if they are identical, which is a well-known property in TTE. On the other hand, if applied to different arguments, comparing will eventually lead to the correct result, maybe on the cost of a high precision.

The current version of the iRRAM implements the reiterations using a `longjmp`, i.e. an immediate return to an outer level of control. If the iRRAM is called in the default way as `iRRAM_exec(compute)`, this jump (usually, but see the chapter on limits) leaves the procedure `compute` and returns to `iRRAM_exec`. Here the precision bound is incremented as above, then `compute` is restarted from the beginning. Coming versions of the iRRAM might use the exception handling mechanisms of C++ instead of `longjmp`.

8 Input and Output

The iterative character of the computations described above poses problems for the interactive usage of the iRRAM: In each iteration each input from `stdin` and each output to `stdout` is repeated. Although this can be used to identify the iterations, any illusion of working with entire real numbers is destroyed. So it is necessary to use special IO-functions to realize the IO-behavior described in section 3.

Accordingly, the user should use `rscanf` for reading one input and `rprintf` for writing (a variable number of 'discrete') outputs, instead. These functions are built similar to the `C`-style I/O-functions `scanf` and `printf` and are able to use the usual format specifiers from `C` resp. `C++`.

In addition, for the output of real numbers, there is a function `rwrite(x,p,w)`, which prints a normalized decimal approximation to a real number with an error of at most 2^p and with a width of max(9,w) characters. If the real number is smaller than 2^p, the output consists simply of a single '0' padded with blanks.

As a variant, there exists `rwrite(x,w)`. Here the output '0' implies $x \leq 10^{-w}$, in any other case the output consists of a normalized decimal approximation to x with w characters, where only the last shown decimal might differ by 1 from the exact decimal value of x.

To print these precise results, the iRRAM may need reiterations. If this should be avoided, the function `rshow(x,w)` can be used to show x in a field of at most w characters (padded with blanks, if necessary). Again, the shown result is as correct as possible, but may have less then w significant digits. In addition, denormalized output in the form e.g. '.*E-0003' just indicates that the value of x is below 10^{-3}. So although `rshow` does never lead to a reiteration, the result might not be very helpful in case of a denormalized output,.

The implementation of `rscanf` simply uses buffered input: the input from the user is copied to a finite buffer (currently of fixed size 100 KB). On each reiteration a pointer in the buffer is reset to the beginning of the buffer, then inputs are taken from the buffer moving this pointer. Only when the pointer reaches the end of the buffered input, new characters are appended to the buffer by reading from the standard input.

`rprintf`, `rshow` and `rwrite` simply use a counter for the number of outputs form previous iterations to determine whether a call should actually lead to new output. This counter can be considered as a part of the multi-value cache, as its value must survive the reiterations.

Input and output of reals (i.e. of infinite objects, not just approximations like in `rwrite(x,w)`) are not implemented yet. As already said at the end of section 3,

this implies that our programs may stop as soon as they exit from being called via `iRRAM_exec`. Output of real numbers could of course be easily implemented by opening an own output stream for each written real value. In reiterations we would simply append new intervals to the corresponding streams. But as these streams are necessarily of infinite length, the simulation must be continued forever with improved precision bounds after reaching terminating configurations. This could be useful on machines with more than one processor or on computer clusters, where we might pipe the output from one iRRAM into another iRRAM in order to achieve parallelization. Until now, it has simply not been necessary to implement these input or output functions.

9 Multi-valued Functions

In traditional arithmetic for single or double precision numbers, there are many functions that cannot be adapted directly to the iRRAM: Essentially, the semantics of the iRRAM must be continuous, whereas these functions are not continuous. Sometimes, it is possible to implement similar functions by restricting the domains, like division with the exception of zero or the tests, where identical arguments lead to infinite loops.

Another possibility has been used in [BrHe94,BrHe95,Br96,Br99]: operations with uncertainty that usually give the correct result but also are allowed to give a 'slightly' wrong answer near points where the operation is not continuous in its arguments. So here an operation might deliver one of several different results for the same arguments. Instead of the usual single-valued functions we have to deal with relations $R \subseteq \mathbb{R}^2$ or set-valued functions $F : \mathbb{R} \to 2^{\mathbb{R}}$. (Both notions are equivalent, simply use $F(x) = \{y \mid (x, y) \in R\}$.) The underlying concept of a 'computable relation' had already been briefly defined in [We87].

Computing such a set-valued function F for an argument x means that the result of the computation of F on x must be one of the values in the set $F(x)$. An alternative approach (where a computation must compute *all* values of $F(x)$) has been intensively discussed in [Br99]. Although that approach has advantages from the theoretical side, any implementation would surely be quite ineffective.

Set-valued functions have been incorporated into the iRRAM since its very first prototype [Mu96]: In the iRRAM, the set-valued F appears like an ordinary (but *multi-valued*) function $f : \mathbb{R} \rightsquigarrow \mathbb{R}$, where the value $f(x)$ must be one of the elements of $F(x)$. The actual choice of the value is hidden and appears to be nondeterministic: The same real number x may lead to different results $f(x)$ at different points of a computation.

In the following, we list most of the intrinsic multi-valued functions that are implemented in the iRRAM. Of course, the user can easily construct new functions of this kind, either through the usual control structures of `C++` or using a special limit operator that will be explained in the next chapter.

`long size (const REAL& x)`

`size(x)` is one of the integer values $k = \lfloor \log_2 |x| \rfloor + 1$ or $k = \lceil \log_2 |x| \rceil + 1$. So the value of `size(x)` is not exactly fixed, but it is sure that $|$`x`$|$ lies in the open interval from 2^{k-2} to 2^k.

The iRRAM is allowed to return any of the possible values. In consequent calls with the same real value x, we might even get different values for `size(x)`, even

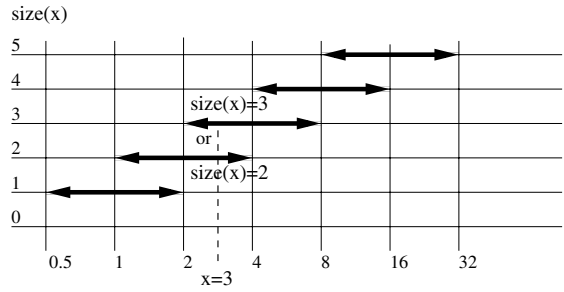


Fig. 3. The multi-valued function `size(x)`.

a test like `size(x)==size(x)` might return `false`. From the view of the programmer, this is a kind of nondeterminism in the program.

On the simulating level, where x is represented as an interval $(d \pm e)$, the choice of k is deterministic: We simply compute $k = \min\{n \in \mathbb{Z} \mid 2^n > |d| + e\}$, which immediately implies $|x| \leq |d| + e < 2^k$. If additionally $4e \leq |d|$, then $|x| \geq |d| - e > (|d| + e)/2 > 2^{k-2}$, so in this case we return k as the value of `size(x)`.

In case of $4e > |d|$, the precision of the interval might be insufficient, so we invoke a reiteration. This implies that `size(0)` leads to an infinite loop.

Because of the limited exponent of the MP numbers, it is sufficient to use `long` as result type for `size`.

`bool bound (const REAL& x,const long k)`

`bound`$(x, k) = \text{true}$ implies $|x| < 2^k$, while `bound`$(x, k) = \text{false}$ implies $|x| > 2^{k-2}$. So in the interval $2^{k-2} \leq |\mathtt{x}| \leq 2^k$, both truth values might be returned. Nevertheless, `bound(x,k)` can be used as a test on the size of $|\mathtt{x}|$.

If it is necessary to check that a number is small enough (e.g. given as the error of an approximating algorithm), `bound(x,k)` should be preferred to `size(x)<k` as there is no singularity for $x = 0$.

`bool positive (const REAL& x, const long k)`

`positive`$(x, k) = \text{true}$ implies $|x| > -2^k$, while `positive`$(x, k) = \text{false}$ implies $|x| < 2^k$. In the interval $(-2^k, 2^k)$, the result may be 'wrong'.

In contrast to the comparison operator '$x < 0$', the function `positive` has the advantage to be total, so it can be used even if it is unknown whether $x = 0$.

`DYADIC approx (const REAL& x, const long p)`

`approx(x,k)` results in a `DYADIC` d such that $|\mathtt{x} - d| \leq 2^{-k}$. The function can be used to enhance the set of `DYADIC` functions: If `REAL f(const REAL& x)` computes $f : \mathbb{R} \to \mathbb{R}$, then for any `DYADIC d,` we can get a `DYADIC` approximation to $f(d)$ with an error of at most 2^p simply with `approx(f(d),p)` (using an implicit conversion of `d` to type `REAL`).

Calling these function may imply reiterations, but (with exception of `size(0)`) they will eventually return a value, when the underlying intervals become small enough.

The functions can be used e.g. in the construction of loops: For any p and $x \geq 0$, the following (multi-valued!) function returns a real y such that $|y - \sqrt{x}| \leq 2^p$ using Heron's iteration algorithm for the square root.

```
REAL sqrt_approx(long p,REAL x)
{ REAL a=1,b=x;
  do { a=(a+b)/2; b=x/a; } while ( !bound(a-b,prec) );
  return a; }
```

In our implementation of the iRRAM, we must be careful as soon as intrinsic multi-valued functions are called: In different iterations, we get different intervals, so that the (deterministic) computation of the functions might lead to different results. For example, when computing `size`(x), an early iteration might deliver $\lceil \log_2 |x| \rceil + 1$ as result, based on some interval $(d \pm e)$. A later iteration might get another interval $(d' \pm e')$ with smaller $|d'| + e' < |d| + e$, so at the same stage of the computation we now would get $\lfloor \log_2 |x| \rfloor + 1$. Although both results are consistent with the definition of `size`, we might have trouble with the synchronization of the flow control in the iterations, as can easily be imagined from the following few lines of source code:

```
if ( size(x) < p  ) {  rprintf("Input A:");   rscanf(a); }
               else  {  rprintf("Input B:");   rscanf(b); }
```

If the results of multi-valued functions would change between iterations, a completely different behavior in different iterations could occur. The iRRAM solves this problem using its multi-value cache: It simply recalls known values of multi-valued functions from the cache before computing any new values (that are immediately appended to the cached values). In consequence, any flow of information from the continuous world of reals to the discrete world stays unchanged through the reiterations. This again implies that the flow of control in a computation will be repeated in reiterations up to the point where the previous iteration ended. As a side effect, we only need to maintain the sequence of results from multi-valued functions and a pointer to the last recalled value, and do not need any further information on the context of the operations to reproduce their results.

The current implementation simply defines an array of 100000 `long`s to store the results of `size` or of tests and an additional array of the same size to store the results of approximations using `approx`. In consequence, these operations should be applied as rarely as possible! In the following chapters, we discuss environments where caching the results is not necessary (or even causes errors). There the operations can be used freely without permanently occupying memory.

10 Computation of Limits

A unique feature of the iRRAM are several operators for the computation of limits of certain families of real numbers or real-valued functions. On one hand, this ability implies that the iRRAM e.g. is not able to check whether two given real numbers are equal (e.g. if one of them is such a limit). On the other hand, the limits operators are very powerful: Almost any important object in analysis is defined as some kind of a

limit, e.g. computing the square root can be done by computing the limit of the function `sqrt_approx` from the previous chapter.

This capability of computing limits extends the computational power of the iRRAM in a way that the scope of algebraic or symbolic computations is left, also the model of Blum, Shub, and Smale[BSS89] is no longer applicable. The correct theoretical model is Type 2 Theory of Effectivity (TTE) [Ko91,We95,We97], although this model works essentially on a Turing machine level. Example definitions of computations on topological structures like $\mathbb{R}$ on an abstract level (i.e. without using representations) can be found in [TZ99] (but without multi-valued functions and without an internal limit operator) or [Br99] (but with a slightly different approach to multi-valued functions).

Currently, the iRRAM implements three types operators for limits:

- for families $\{a_p \mid p \in \mathbb{Z}\}$ of functions $a_p : M_1 \dashrightarrow M_2$ converging uniformly and with known speed to a *single-valued limit* f with $f(x) = \lim_{p\to-\infty} a_p(x)$,
- for families $\{a_p \mid p \in \mathbb{Z}\}$ of functions similar to above, but where we have additional information on the limit f (essentially f must fulfill a *Lipschitz condition*),
- for families $\{a_p \mid p \in \mathbb{Z}\}$ of functions converging to a *multi-valued limit* $f(x)$.

Here M_1 and M_2 are the metric spaces of real numbers (Euclidean metric), complex numbers (using $d(a+ib, c+id) = \max\{|a-c|, |b-d|\}$), or real matrices (using $d((a_{ij}), (b_{ij})) = \max\{|a_{ij}-b_{ij}|\}$). The iRRAM is already prepared to deal with other metric spaces, although until now it was not necessary to implement such a case.

The limit operators need $\{a_p \mid p \in \mathbb{Z}\}$ as one of their parameters, usually as a function a of $p \in \mathbb{Z}$ and $x \in M_1$. This implies that there must be a program computing a, so the operators will only be applied to computable parameters, i.e. to families of (multi-valued) continuous functions.

10.1 Simple Limits

For limits of the first type we need a family $\{a_p\}$ e.g. of functions $a_p : \mathbb{R} \rightsquigarrow \mathbb{R}$ converging to a function $f : \mathbb{R} \dashrightarrow \mathbb{R}$ in the sense that $|a_p(x) - f(x)| \leq 2^p$ for any $x \in dom(f)$, $p \in \mathbb{Z}$ (and any possible value of the multi-valued $a_p(x)$). The corresponding limit operator has the signature

```
REAL limit (REAL a(long, const REAL&), const REAL& x);
```

It should be emphasized that the functions a_p will usually be multi-valued, but the limit f must be single-valued. In consequence, f will be a (partial) continuous function.

An example for $a_p(x)$ is `sqrt_approx(p,x)` already mentioned in the previous chapter; here the limit f is the square root defined on $\mathbb{R}_0^+$. In the iRRAM, we may now define the square root function by

```
REAL sqrt(const REAL& x) {return limit(sqrt_approx,x);}
```

This function `sqrt` can be used like any of the built-in elementary functions!

The idea of the implementation is quite simple: If we want to compute such a limit $\lim_{p\to-\infty} a_p(x)$ in an iteration, we simply compute and return $a_p(x)$ for some p, such that p tends to $-\infty$ with improving precision bounds $\overline{p_i}$ during the iterations.

Two problems must be faced: (1) the second argument `x` to `limit` is only known with errors, and (2) the flow of control will be different in different iterations!

Problem (1): The argument `x` in `limit(a,x)` is not known exactly during any stage of the computation. So we are only able to compute the values $a_p(x)$ with an error at least corresponding to the modulus of continuity of the function a_p using the given approximation to x. In practice, the best achievable result will usually be much worse, as the computation of $a_p(x)$ need not to be optimal with respect to error propagation. To solve this problem, the iRRAM tries to compute $a_p(x)$ for several values p that are chosen the same way as the precision bounds: In the i-th iteration of the iRRAM using precision bound $\overline{p}_i$, we consider the values $a_p(x)$ for $p = \overline{p}_0, \overline{p}_1, \ldots, \overline{p}_i$. Then we take that result giving the smallest error (which is the sum of the error from the computation of $a_p(x)$ and the bound 2^p for the difference between $a_p(x)$ and the limit $f(x)$).

So here we face the situation that a failure of a computation due to insufficient precision should *not* lead to a recomputation of the whole program. Instead of this, we have *local iterations* of the computations, i.e. the `longjmp` from within the computation of the limit must no longer lead to `iRRAM_exec`, but to the procedure implementing `limit`. In addition, `limit` has a `longjmp` to an outer level, if $a_p(x)$ can not be computed for any of the tested values of p due to the error in `x`. After that outer `longjmp`, the program will return to the computation of the limit with a better precision bound and hence a (usually) better approximation for the argument x. This ensures that eventually there will be successful tries of $a_p(x)$ even for arbitrarily small p. In consequence, the error of the approximations for the limit will get arbitrarily small in later iterations. This behavior of the limit operator (and of all the other intrinsic functions) is the origin for the convergence property mentioned in section 3.

Problem (2): For different iterations of the iRRAM, we now can not avoid different flows of control: The iRRAM has to compute a_p for several different values p depending on the precision bounds $\overline{p}_i$. But due to the construction, the different control flows are restricted to the computation of the limit, which is continuous in its argument x. So it is sufficient for the iRRAM to stop using its multi-value cache as long as it is in the process of computing a limit.

In consequence, the local changes of the flow of control will have no influence on the outer flow of control unless there are side effects. So neither input, output nor any other side effects (e.g. assignments to global variables) are allowed in the algorithm computing $a_p(x)$. Unfortunately, this can not be expressed within the syntax of `C++`.

Computing the limit in the way described above has an important disadvantage: The precision bounds $\overline{p_j}$ were chosen sparse in order to get an optimal behavior of the time complexity of the whole computation. But now this implies that the error of the result $f(x)$ might be very big compared to the error of the argument x: Sometimes it will not be possible to compute $a_{\overline{p_i}}(x)$ successfully using precision bound $\overline{p_i}$, regardless of the actual value of $\overline{p_i}$, simply because in that iteration x already has an error of this size. So it might be that, in any iteration, the best achievable result of the limit operator is only $a_{\overline{p_{i-1}}}(x)$ implying a very big loss of precision!

For example, numerical iterations $x_{i+1} = \Phi(x_i)$, where Φ is implemented using this limit operator, might only be usable for small values of i, as we will need at least iteration $\overline{p_i}$ of the iRRAM to get x_i with an error of $\overline{p_0}$.

Some heuristic improvements in the implementation should be mentioned: If in the iteration with precision bound $\overline{p_i}$ a limit operator is called, the values $a_p(x)$ are computed in the modified order $p = \overline{p_i}, \overline{p_0}, \overline{p_1}, \ldots, \overline{p_{i-1}}$ and with a precision bound that is changed to $\overline{p_{i+1}}$ during the computation of the limit. This change in the precision bound increases the probability that $a_{\overline{p_i}}$ can be computed successfully. If this is the case and if the resulting interval for $a_{\overline{p_i}}$ is not larger than $2^{\overline{p_{i-1}}}$, then this interval is taken as the result of the limit in this iteration (of course increased by $2^{\overline{p_i}}$). As a consequence, quite often it is not necessary to test more than one value.

10.2 Lipschitz Limits

If we have more information on the limit, we can do much better than above. Very often, we know that a (maybe lengthy) computation leads to a quite smooth result. One example is the trigonometric function sine, where we know that $|\sin(x) - \sin(y)| \leq |x - y|$. In this case, we know that it would be possible to approximate $\sin(x)$ by $\sin(d)$ with an error of only e, if x is known to be in the interval $(d \pm e)$. But as a longer computation is necessary to compute sine, any interval arithmetic starting with $(d \pm e)$ must yield an interval $(d' \pm e')$ with a much larger error $e' \gg e$. So the central idea is to use interval arithmetic not for $\sin(d \pm e)$, but for $\sin(d \pm \tilde{e})$ for a very small $\tilde{e}$ yielding an interval $(d'' \pm e'')$ with $\sin(d) \in (d'' \pm e'')$ and a still small e'', and then to use $(d'' \pm (e + e''))$ as a valid interval approximation for $\sin(x)$.

We may generalize this idea as follows: If x is known as an interval $(d \pm e)$ when computing the limit and if we know an upper bound 2^ℓ of a Lipschitz constant for the limit f in the interval $(d \pm e)$, we may simply choose $p \geq \overline{p_i}$ such that 2^p has about the same size as $2^\ell \cdot e$, and compute $a_p(d)$. Then

$$|a_p(d) - f(x)| \leq |a_p(d) - f(d)| + |f(d) - f(x)| \leq 2^p + 2^\ell \cdot e$$

i.e. $a_p(d)$ differs from the limit $f(x)$ by at most $2^p + 2^\ell \cdot e$. A similar concept had already been introduced in [Mu88] as a 'locally Lipschitz continuous function'.

Now the multiple precision number d can be treated as an exact real value, so we are able to compute $a_p(d)$ with an arbitrary small error: To do this, the iRRAM simply uses local iterations for the computation of $a_p(d)$ with precision bounds $\overline{p_{i+1}}, \overline{p_{i+2}} \ldots$ until one of these iterations gives a result $(d'' \pm e'')$ with an error e'' of at most 2^p. Together we get $(d'' \pm (e'' + 2^p + 2^\ell \cdot e))$ as a valid approximation for $f(x)$, so that the resulting error is as close as possible to the minimal possible error $2^\ell \cdot e$.

These iterations work similar to above: The `longjmp` within the computation of the $a_p(d)$ is redirected temporarily to the procedure `limit_lip` implementing the Lipschitz limit. However, there are two big differences to the implementation of the simple limit operator: For the simple limit we had a fixed precision bound within the different local iterations, but the index p of $a_p(x)$ was changed from iteration to iteration; now the Lipschitz limit is computed with a fixed index p of $a_p(d)$, but the precision bound changes. This also implies that the application of the Lipschitz limit will never lead to a reiteration of the whole computation.

Again, the local iterations do not need a multi-value cache as their result leads to a continuous function. As the implementation of local iterations starts with precision

bound $\overline{p_{i+1}}$ and only needs to get a result with error of size $2^{\overline{p_i}}$, it will happen very often that already the first iteration is successful.

The corresponding limit operator has the following signature, where the variable `lip` corresponds to the logarithm ℓ of the Lipschitz constant 2^ℓ from above:

```
REAL limit_lip (REAL a(long, const REAL&),
                          long lip, const REAL& x);
```

This operator has important advantages compared with the simple limit operator introduced before: (1) the computation of the limit is faster due to less inner iterations, (2) the growth of the errors is reduced to the minimal possible amount and in consequence (3) the overall computation time is also reduced very often, because smaller errors usually lead to fewer outer iterations. Most of the functions in the iRRAM use this concept, e.g. exp, sin and cos.

Here we will show how to implement the maximum function using the Lipschitz limit operator. To compute the maximum of two numbers using floating point arithmetic, usually a program similar to the following is used:

```
if ( x>y ) max=x; else max=y;
```

Obviously, this is not a suitable solution for the iRRAM, as `x=y` would lead to an infinite loop. So the right way is to interpret the maximum as a limit:

```
REAL max_approx (long k, const REAL& x, const REAL& y)
{ if ( positive(x-y,k) ) return x; else return y; };

REAL maximum  (const REAL& x, const REAL& y)
{ return limit_lip(max_approx,0,x,y); };
```

As a not yet implemented improvement an additional version of the Lipschitz limit operator is planned, where the Lipschitz constant can be given more precise, e.g. as a real number L or as a value $\ell_1 \cdot 2^{\ell_2}$ from Err instead of a simple integer power 2^ℓ.

10.3 Multi-valued Limits

In the previous examples, the results of intrinsic multi-valued functions were from discrete sets (boolean, integer, finite strings etc.) and it was possible to compute these results in finite time. Especially, it was always possible to store the actual outcome of an intrinsic multi-valued function in the cache in order to reuse it in a subsequent iteration of the iRRAM. This is no longer possible, if we are considering multi-valued functions with results e.g. from the set of real numbers.

Of course, simple functions of this kind can be constructed using the discrete multi-valued tests, e.g. $\mathtt{modulo}(x, y) := x - ky$ where $k \in \mathbb{Z}$ with $(k-1)y \leq x \leq (k+1)y$. This function computes one of the possible values z, $|z| \leq y \neq 0$, such that $\frac{x-z}{y}$ is an integer. It is used to scale the arguments of periodic trigonometric functions.

```
REAL modulo (const REAL& x, const REAL& y)
{ return x-round(x/y)*y; };
```

Here round(x) is an intrinsic integer-valued (and multi-valued) function with $| x - \mathtt{round}(x) | < 1$. When applying the modulo function, the iRRAM would automatically store the result of round, which is sufficient to reconstruct the value of modulo in possible later reiterations.

A more elaborate example concerns the square root of complex numbers z: The square root z is one of the two numbers x or $-x$, where $x^2 = z$, which immediately leads to multi-valuedness. If we restrict ourself to $z \in \mathbb{R}_0^+$, it is easy to choose one of the values: usually the positive root is taken; but for arbitrary $z \in \mathbb{C}$, a *continuous* choice of one of the roots is not possible.

If we restrict ourself to $z \neq 0$, a multi-valued implementation of the complex square root can be implemented using the simple multi-valued tests: If $z = a + ib \neq 0$, at least one of the real numbers a and b must also be different from 0. Then one of the complex roots can be computed using the ordinary real square root, but the computation may depend on the non-zero value that is determined in a multi-valued way. The only small difficulty is to find this non-zero value without having a test for equality of real numbers:

```
COMPLEX csqrt(const COMPLEX& z)
{ long choice;
  REAL r= abs (z),          a= real(z),          b= imag(z),
       c= sqrt( (r+a)/2 ), d= sqrt( (r-a)/2 );

  if ( !bound( b, size(r)-3) )
       { if ( b > 0 )  choice=1;  else  choice=2; }
  else { if ( a > 0 )  choice=3;  else  choice=4; }

  switch ( choice )
  { case 1: return COMPLEX (  c      ,  d     );
    case 2: return COMPLEX (  c      , -d     );
    case 3: return COMPLEX (  -c     , -b/2/c );
    case 4: return COMPLEX (  b/2/d  ,  d     );  }
}
```

Each value of the variable choice corresponds to one of the following four open half planes in $\mathbb{C}$: $\{z \mid imag(z) > 0\}$, $\{z \mid imag(z) < 0\}$, $\{z \mid real(z) > 0\}$, and $\{z \mid real(z) < 0\}$. So for any $z \neq 0$, the initial tests in the program will be left after setting an appropriate value for choice.

It can easily be verified that the returned value is indeed one of the two roots of z. Figure 4 shows results of this function for the square roots on circles with radius 1, 2, and 3. The results for $b < 0$ and $a < 0$ are adjacent, so only three of the four values for choice can be distinguished. Changing the result for $a > 0$ to the more 'natural' COMPLEX (c,b/2/c) would have the effect that all results are adjacent (leading to a less interesting figure).

This solution for the square root is not fully satisfying: We use size to find a lower bound for the nonzero value from a or b. So for $z = 0$, the result is not defined and an infinite loop occurs, and for arguments near to 0, the complexity is large even for a computation with low precision .

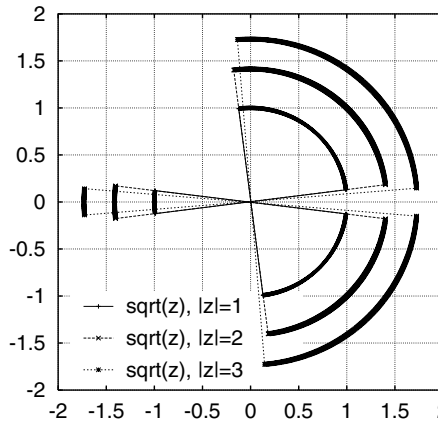


Fig. 4. Values of the complex square root in the iRRAM, using `csqrt`.

To overcome both problems, the complex square root can be seen as a limit: For arguments z with $|z| > \varepsilon$, let $f(\varepsilon, z)$ be one of the roots of z, otherwise let $f(\varepsilon, z) = 0$ as an approximation. Then simply let $\varepsilon \to 0$.

When trying to implement this within the iRRAM, we must face the problem that for different ε or for different approximations of z, the values $f(\varepsilon, z)$ might switch between the two roots of z. But when computing the limit, we may not switch between the roots, otherwise in different iterations changed flows of control could occur. So the actual choice of the root must be passed from iteration to iteration, which reminds of ordinary iterative numerical procedures.

To implement this, the argument list for the operator computing multi-valued limits must be extended:

```
COMPLEX limit_mv
       ( COMPLEX f(long p, long* choice, const COMPLEX& z),
         const COMPLEX& z );
```

The parameter `choice` can be interpreted as follows: It selects a subset of the possible values of the limit that may be computed by `f`. The (initial) value of 0 has the interpretation that all the possible values are still allowed. If `f` changes the value of `choice`, then this must correspond to a reduction of the set of allowed values. At the end of the computation of `f`, the result of `f` must be an approximation (with an error of at most 2^p) to all of the values that still correspond to `choice`.

To compute the limit, we proceed almost as in the case of the usual limit operator, i.e. different values of `p` are checked whether the computation of `f(p,choice,z)` leads to an approximation of the limit. The very first try starts with `choice=0`, i.e. `f` may approximate any of the possible values. For the other tests (in the same or later iterations), the value of `choice` is passed from one computation of `f` to the next using the multi-value cache. So a result of a computation of `f(p,choice,z)` at some stage in the iRRAM must be a refinement of the previous computations at this stage.

As said above, this multi-valued limit operator corresponds to iterative numerical procedures, but all necessary information on earlier iterations must be contained in `choice`. Although this simple version of the operator should be sufficient in theory (where `choice` could carry arbitrary denumerable information), it might be too weak for many examples in practice (where `choice` only carries 32 bits). On the other hand, only a few essential points of the iterations might be important, so choosing type `long` for `choice` can still be far reaching. However, it is planned to implement generic iteration operators as a generalization of this concept.

In the case of the complex square root, the value of `choice` will need at most one change: If z is so big compared to the intended precision that 0 is no longer a valid approximation to the roots of z, `choice` is changed to indicate in which of the half planes z is located. This is sufficient to choose one of the roots of z, and this choice will not be changed again. A corresponding implementation being able to compute a square root for any complex z including 0 looks as follows:

```
COMPLEX csqrt1(long p, long* choice, const COMPLEX& z)
{ REAL r= abs (z),          a= real(z),          b= imag(z),
       c= sqrt( (r+a)/2 ), d= sqrt( (r-a)/2 );

  if ( *choice == 0 && !bound(r, 2*p) )
      if ( ! bound(b,p-2) )
           {  if ( b > 0 )  *choice=1;  else  *choice=2;  }
      else {  if ( a > 0 )  *choice=3;  else  *choice=4;  }

  switch ( *choice )
  { case 0: return COMPLEX (  0      , 0      );
    case 1: return COMPLEX (  c      ,  d     );
    case 2: return COMPLEX (  c      , -d     );
    case 3: return COMPLEX (  c      , b/2/c  );
    case 4: return COMPLEX (  b/2/d  ,  d     );  }
}

COMPLEX  sqrt(const COMPLEX& z) { return limit_mv(csqrt1,z); }
```

11 Optimizing: Hints and Assertions

In the following we briefly discuss three methods to optimize the efficiency of programs for the iRRAM:

(1) If a part of a numerical algorithm is known to have a modulus of continuity significantly larger than in the rest of the program, e.g. if we try to compute a function with a large first derivative, it may improve the overall complexity significantly if we temporarily use a higher precision which may lead to less reiterations. In this case, the sensitive part of the program can be marked with a pair of functions `stiff_begin()` and `stiff_end()`, where the precision bound is improved by the first function and reset by the second function.

This technique had initially been used e.g. in the AGM methods for π and $\log(x)$. In these special cases it turned out to be actually much better to temporarily switch the

influence of the precision bound on the actual computations from absolute to relative precision.

The use of these functions is simply a *hint* to the iRRAM how to proceed most efficiently. Using these functions might change the flow of control because of the possibility of changing results from multi-valued functions. However, their use can not influence the correctness of a program, only its time complexity.

The following methods behave different, their use is not only a hint but an *assertion* that a part of a program has certain special properties (that can not be verified by the iRRAM). The computation will be optimized correspondingly.

(2) The multi-value cache is an expensive simulation tool, as each cached result needs to be stored until the end of the program. This is why the use of multi-valued intrinsics should be reduced to the minimum.

On the other hand, there are many computations leading to a single-valued continuous results although they use multi-valued decisions. As an example, consider $\sin(x)$ again. It is well known that using a Taylor series in the computation of $\sin(x)$ for large x is a big waste of time. It is better and quite simple to use the periodicity of this trigonometric function for a range reduction and to compute a y of size $\pi/4$ or less such that either $\sin(x) = \pm\sin(y)$ or $\sin(x) = \pm\cos(y)$. To do this, we have to apply multi-valued tests, as we have to distinguish between 4 distinct cases that later fit together to a result that is single-valued and continuous in x. In consequence, it would not be necessary to cache these tests.

If such a part of a program is marked with the functions `continuous_begin()` and `continuous_end()`, the caching is temporarily stopped. The programmer must take care himself that it is impossible to enter or leave the section without calling the corresponding function (except maybe for reiterations), as this could easily destroy the consistency of the multi-value cache.

Another important application is found in linear algebra: If we invert a nonsingular matrix M, the inverse M^{-1} is determined uniquely and the mapping from M to M^{-1} is continuous. Here the classical algorithms based on Gaussian elimination internally rearrange the matrices due to the size of pivot elements which requires multi-valued tests that need not to be cached.

(3) If we know that a sequence of operations has a significantly better modulus of continuity than we achieve through the normal interval arithmetic, we may be able to reduce the growth of the error bounds using a technique similar to the Lipschitz limit operator: Instead of calling a function f with known Lipschitz constant 2^l directly, we may use the form `lipschitz`(f, l, x). Here the precision bound is increased during the computation, and instead of $f(x)$ we compute $f(d)$ for the center d of the interval $(d \pm e)$ representing x. The resulting interval is corrected according to the Lipschitz constant and the size e of the interval. In consequence, the error propagation between x and $f(x)$ is almost reduced to the minimum.

12 Overview: Classes and Functions

In the following we give a brief overview of the classes, operators, and functions, that are implemented in the iRRAM. Unless explicitly stated, these functions are intrinsic;

their implementations may use private properties of the class REAL and have access to the underlying intervals that are hidden to the ordinary user. We will not repeat the limit operators here, as they have been explained previously in detail.

Basic classes

The class DYADIC is essentially a wrapper for the underlying multiple precision arithmetic. The most important class REAL is derived from a basic (mostly hidden) class METRIC_OBJECT that provides a rudimentary garbage collection necessary for the reiterations and that is the basic class for the limit operators. Further derived classes are COMPLEX, REALMATRIX, SPARSEREALMATRIX, where the latter aims at very large but mostly empty matrices. All those classes have a destructor ~, a default constructor, and a copy constructor = for assignments.

Real arithmetic

REALs can be constructed from `int`, `long`, `char*`, `double`, DYADIC, and from REAL itself. In consequence, there are corresponding implicit type conversions, e.g. `x=REAL("3.1415")` can be written as `x="3.1415"`.

The overloaded arithmetic operators '`x+y`', '`x-y`', '`-y`', '`x*y`', and '`x/y`' can be used in a naive way, but a division by `REAL(0)` leads to an infinite loop.

The usual binary tests '`x<y`', '`x<=y`','`x>y`', and '`x>=y`' exist, but there is no test on equality or inequality of reals. These tests will lead to infinite loops if `x` and `y` are equal. So '`x<y`' and '`x<=y`' really are the same function.

The multi-valued functions '`size(x)`', '`positive(x,p)`', '`bound(x,p)`', '`approx(x,p)`', and '`round(x)`' have been explained previously. As `round` delivers a result of type `long` (which is very restricted in size), there is a corresponding function `round2(x)` with a result of type REAL.

Quite often, multiplication with a (positive or negative) power of 2 is necessary, which is implemented as an intrinsic function `scale(x,k)`.

Finally there is a library of mathematical functions that are not intrinsic (i.e. they are defined essentially 'high level' but usually with use of the limit operators). These function are still very efficient, as the necessary overhead is small. There are algebraic functions like `modulo(x,y)`, `maximum(x,y)`, `minimum(x,y)`, `sqrt(x)`, `root(x,n)`, `power(x,n)`, and `abs(x)`; transcendentals include `exp(x)`, `log(x)`, and there is a full set of trigonometric functions and their inverses, from `sin(x)` to `acosech(x)`. In addition, the special numbers `pi()` $= \pi = 3.1415...$, `ln2()` $= \log(2) = 0.6931...$ and `euler()` $= e = 2.1718...$ are defined (necessarily as functions).

Matrix arithmetic

The classes REALMATRIX and SPARSEREALMATRIX deal with matrices of real components. As they are derived from the class METRIC_OBJECT, limit operators can be applied. The corresponding algorithms are usually not intrinsic, most of them are written as applications of the type REAL. There is a variety of constructors, like `(SPARSE)REALMATRIX(r,c)` for a (sparse) matrix of `r` rows and `c` columns, or like `(SPARSE)REALMATRIX(m)` to duplicate a matrix `m`. To access matrix elements, the parentheses `()` have been overloaded, allowing assignments like `m(1,2)="3.1414"`.

Basic matrix arithmetic 'm1+m2', 'm1-m2', 'm1*m2', 'm1/m2' is implemented, where the latter computes a solution of the linear system $M_2 \cdot X = M_1$ by Gaussian elimination (if M_2 is not singular). Multiplication `m*x` and division `m/x` by scalar values `x` are defined. In addition, `eye(n)`, `zeroes(r,c)` or `ones(r,c)` deliver the unit matrix, or a matrices filled with zeroes or ones.
The matrix exponential $\exp(m) := \sum m^k/k!$ is an example for the implementation of a matrix transcendental as a limit.

Complex arithmetic
`COMPLEX` numbers are a further example for the possibility of user-defined data types. Only a few functions have been implemented yet: basic arithmetic, `abs(z)`, `real(z)`, `imag(z)`, as well as the complex square root that has been the central example for the multi-valued limits in section 10.

Input and Output
The special role of I/O and the functions `rscanf`, `rprintf`, `rshow` and `rwrite` have been discussed in section 8. Additionally, there is `x.check()` printing approximations to `x` *without* respect to the iterations. It can be used for debugging.

Special Functions and Operators
The special functions `continuous_begin()`, `continuous_end()` (modifying the caching strategy), as well as `stiff_begin()`, `stiff_end()`, and the operator `lipschitz(f,l,x)` (modifying the error propagation) have been discussed in the section 11.

13 Extending MP Packages

Although the iRRAM implements *exact* arithmetic implying that programs should run forever, it also can be used as an extensible library of functions with variable precision: If the programmer defines an own version of `main`, he is able to call the iRRAM via `iRRAM_exec(compute)` several times within his program, even with several different functions instead of `compute()`.

A technical problem lies within the parameters: As the argument of `iRRAM_exec` is of type `void()`, `compute` may neither have arguments nor return a result. So the only possibility to pass data between `main` and `compute` is to use global variables, usually of type `DYADIC`. Using this, we are able to define new functions extending the multiple precision backend. For example, the iRRAM sources contain some extensions to GMP, where e.g. the cosine (still missing in GMP!) is implemented as follows:

```
DYADIC glob_arg,glob_result; long glob_precision;

void compute_cos()
{ glob_result=approx(cos(REAL(glob_arg)),glob_precision); }

DYADIC cos(DYADIC x, long p)
{ glob_arg=x;                glob_precision=p;
  iRRAM_exec(compute_cos);   return glob_result; }
```

Here we use three global variables: for the argument, the result and the precision of the computation. Please remember: we are implementing a finite precision version of cosine using our exact real version, so we have to specify the precision of the result!

Current multiple precision packages like [Zi00] intend to smoothly extend the hardware floating point arithmetic and still try to follow the principles of IEEE standards for floating point arithmetic. This implies e.g. that faults like non-monotonic behavior of the implementation of monotonic functions must be avoided. One similar aspect are exact rounding modes, i.e. it should be possible to specify that the result of an MP operation is either the next, the greatest smaller or the smallest larger MP number compared with the exact result.

From the view of exact real arithmetic, exact rounding modes are similar to testing whether two reals are equal: Nontrivial single-valued real functions with a discrete set of possible values are necessarily discontinuous and hence not computable! Furthermore, any correct implementation of a theoretically monotonic function using exact arithmetic will be monotonic, and any 'irregular' non-monotonic behavior of the conversion to discrete sets can be completely explained using multi-valued functions, so exact rounding is not an issue in exact arithmetic.

It would even be possible to create MP routines with exact rounding for functions f using the iRRAM as backend, as long as the exact values $f(d)$ for MP numbers d are not exactly representable as MP numbers again. E.g. for $\cos(d)$ with $d \neq 0$, we could simply iterate the procedure above with different values for `glob_precision` until we get enough *different* bits to allow the exact rounding. It should almost always be sufficient to choose an initial bounding precision just smaller than necessary for the expected value to decide the rounding without any further iterations. Rare exceptions needing several iterations could surely be tolerated.

14 Exploring the Borders

In the following we present a few examples for the time complexity of the iRRAM. All computations have been performed using an AMD Athlon 800 MHz (512 KB cache, slot A) on an ASUS mainboard K7M equipped with 256 MB 100 MHz SDRAM. The backend was LRGMP using GMP 3.1, operating system was Linux with kernel 2.4.0-test4. Programs had been compiled with `gcc`, version 2.95.2.

– **Selected values and functions**
We computed n decimals of $\sqrt{1/3}$, π, $\log(1/3)$, and $\sin(1/3)$:

Number n of Decimals	10	100	1000	10000	100000
Time for conversion	8 μs	37 μs	39 μs	14 ms	1.5 s
Time for π	-	-	-	140 ms	9.1 s
Time for inversion	3.4 μs	6.0 μs	80 μs	2.9 ms	140 ms
Time for $\sqrt{1/3}$	10 μs	20 μs	113 μs	3.9 ms	327 ms
Time for $\log(1/3)$	145 μs	620 μs	3.9 ms	160 ms	13.1 s
Time for $\sin(1/3)$	97 μs	230 μs	4.1 ms	202 ms	22.6 s

As the internal format of the multiple precision backend is in binary form, we have to split the computation time in two parts. During the computations, a part of the time was taken for the conversion from the internal binary format to the decimal form; this conversion is not optimized yet and has complexity of order $\mathcal{O}(n^2)$.

The other given times do not include this conversion time, so it has to be added to the given values in order to get the full time for a computation with the desired precision including the output.

π is computed using a quartically convergent algorithm due to the Borwein brothers [Ba88,Bo87]. Due to an internal optimization that avoids unnecessary recomputations of π, the timings for small n could not be measured.

For the inversion, the time necessary to compute $1/\pi$ from given π was measured.

The square root $\sqrt{\ }$ is taken almost directly from the MP package, as mentioned in section 4. In case of LRGMP it is implemented with a variant of Heron's quadratic convergent method, but computed with variable precision as in [Bt76].

The implementation of the logarithm uses an AGM method based on ideas from [Sch90] and needs precomputed values for π and $\log(2)$. The given times are without these precomputations, so they are valid for all but the first evaluation of $\log$.

The sine function uses a Taylor series approach after an appropriate range reduction that also uses a precomputed π. The given times are without this precomputation.

– **Iterated function systems**

We computed iterates of the *logistic function* $x_i{=}f(x_{i-1}) = 3.75{\cdot}x_{i-1}{\cdot}(1{-}x_{i-1})$, starting with $x_0 = 0.5$, and printed the first 6 decimals of each tenth x_i:

Index i	1000	5000	10000	50000	100000
Value x_i	+.791747	+.814694	+.824205	+.283081	+.666947
Max. Precision Bound	-2166	-10888	-21407	-102635	-200601
Execution Time	60 ms	1.56 s	8 s	385 s	2054 s

Using `double` instead of `REAL`, beginning from x_{80} the 6th digit is incorrect, x_{90} has only one correct digit left, and finally from x_{100}, all digits are wrong:

	`REAL`	`double`
x_{70}	+.4521952998	0.4521952586
x_{80}	+.8561779966	0.8561759906
x_{90}	+.7399137486	0.7400517104
x_{100}	+.8882939922	0.9017659679
x_{110}	+.7156795292	0.2201217854

This example has been taken from [Ku96], where a similar table was shown up to x_{450} using the software package C-XSC [Kl93]. Unfortunately, this package is quite restricted in the mantissa length, making it is impossible to compute x_i for any $i > 500$.

– **Matrix arithmetic**

We computed approximations (with maximal error 2^{-50}) of the inverse of the (bad conditioned) Hilbert matrix H_n of size $n{\times}n$ using the built-in matrix arithmetic (i.e. with Gaussian elimination) and compared this to the same computation applied to the well conditioned matrix $H_n + \mathbb{1}_n$ of size $n{\times}n$:

	Dimension n	50	100	150	200	250	500
H_n	Max. Prec. Bound	-1037	-2745	-4372	-5502	-6915	?
	Used Memory	1.46MB	9.2MB	40 MB	81MB	152 MB	> 1GB
	Execution Time	3.2 s	79 s	457 s	1200 s	3052 s	?
$H_n+\mathbb{1}_n$	Max. Prec. Bound	-100	-100	-100	-100	-100	-162
	Used Memory	0.5 MB	2.1 MB	4.7 MB	8.4 MB	13 MB	52MB
	Execution Time	0.7 s	5.4 s	19 s	45s	91 s	1237 s

Obviously, the condition of the matrix has big influence on the necessary precision bound, which explains the big differences in execution time and memory consumption between the two examples.

Similar computations were done using `octave` (a freely available high-level interactive language for numerical computations), where optimized routines working with the limited precision of the hardware are used.

`octave` is already unable to invert the Hilbert matrix of size 12 (giving a warning `matrix singular to machine precision`). On the other hand, the inversion of the well-conditioned matrix $H_n + \mathbb{1}_n$ with $n = 500$ takes only 18.8 s, so the iRRAM is about a factor of 65 slower. This factor is different for other types of CPUs, e.g. for an AMD K6-200 with its weaker floating point performance we got a factor of only 35. Please have in mind that here we compare the software iRRAM with the hardware FPU (on the same CPU)!

15 Future Work

The current state of the iRRAM allows reliable and efficient exact arithmetic. But of course, many enhancements and optimizations can be imagined. In the following we list just a few of them:

- Until now, the iRRAM has only been tested using the GNU `g++` compiler and with Linux on i386-type CPUs. A port to other popular systems should be done.
- There should be a better garbage collection that clears all temporary objects whenever a reiteration happens. At the moment, a restricted garbage collector is used that is able to recover memory occupied by objects derived from `METRIC_OBJECT`. However, memory allocated on the stack using `alloca` is freed automatically, as the reiterations are implemented using `longjmp`.
- `COMPLEX` versions of the transcendental and trigonometric functions are missing.
- The algorithms for `exp` and for the trigonometric functions should be further improved using AGM methods, see e.g. [Bt76].
- New data types for (full range) integers or rationals should be added. Perhaps a hierarchy of number types would be possible: `INTEGER` - `DYADIC` - `RATIONAL` - `ALGEBRAIC` - `SYMBOLIC` - `REAL` - `COMPLEX`.
- The trigonometric functions have been optimized for precision bounds in the area of a few thousand bits. On the low precision end, a lot of work is still to be done.
- Planned generalizations of the limit operators have been mentioned in section 10.

References

Ba88. D.H. Bailey, The computation of π to 29,360,000 Decimal Digits Using Borweins Quartically Convergent Algorithm *Mathematics of Computation* Vol. **50** Number 181 (1988) 238-296

Bo87. J.M. Borwein and P.B. Borwein, Pi and the AGM, A study in analytic number theory, Wiley, New York, 1987

BoCa90. H. Boehm and R. Cartwright, Exact Real Arithmetic: Formulating real numbers as functions. In T. D., editor, *Research Topics in Functional Programming,* 43-64 (Addison-Wesley, 1990)

BSS89. L. Blum, M. Shub, and S. Smale, On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, *Bulletin of the AMS* **21**, 1, July 1989

Bt75. R.P. Brent, The complexity of multiple precision arithmetic, Proc. Seminar on Complexity of Computational Problem Solving, Queensland U. Press, Brisbane, Australia (1975) 126-165

Bt76. R.P. Brent, Fast multiple precision evaluation of elementary functions, *J. ACM* **23** (1976) 242-251

Bt78. R.P. Brent, A Fortran multiple precision package, *ACM Trans. Math. Software* **4** (1978), pp 57-70

Br96. V. Brattka, Recursive characterization of computable real-valued functions and relations, Theoret. Comput. Sci. **162** (1996),47-77

BrHe95. V. Brattka and P. Hertling, Feasible Real Random Access Machines, *Informatik Berichte 193 - 12/1995, FernUniversität Hagen*,

BrHe94. V. Brattka and P. Hertling, Continuity and Computability of Relations, *Informatik Berichte 164 - 9/1994, FernUniversität Hagen*,

Br99. V. Brattka, Recursive and Computable Operations over Topological Structures, Thesis, *Informatik Berichte 255 - 7/1999, FernUniversität Hagen*

CoAa89. S.A. Cook and S.O. Aanderaa, On the minimum computation time of functions, *Trans. Amer. Math. Soc.* **142** (1969) 291-314

EdPo97. A. Edalat and P. Potts, A new representation for exact real numbers, *Proc. of Mathematical Foundations of Programming Semantics 13*, Electronic notes in Theoretical Computer Science 6, Elsevier Science B.V., 1997,
URL: `www.elsevier.nl/locate/entcs/volume6.html`

FiSt74. M.J. Fischer and L.J. Stockmeyer, Fast on-line integer multiplication, *J. Comput. System Scis.* **9** (1974) 317-331

Gr00. T. Granlund, GMP 3.1.1, `http://www.swox.com/gmp/`

GL00. P. Gowland and D. Lester, The Correctness of an Implementation of Exact Arithmetic, *4th Conference on Real Numbers and Computers, 2000, Dagstuhl, 125-140*

HN00. S. Heinrich, E. Novak et al., The Inverse of the Star-Discrepancy depends linearly on the Dimension, *Acta Arithmetica, to appear*

Kl93. R. Klatte, U. Kulisch et al., C-XSC , a C++ Class Library for Extended Scientific Computing (Springer, Berlin 1993)

Ko91. K. Ko, Complexity Theory of Real Functions, (Birkhäuser, Boston 1991)

Ku96. U. Kulisch, Memorandum über Computer, Arithmetik und Numerik (Universität Karlsruhe, Institut für angewandte Mathematik)

Mm96. V. Ménissier-Morain, Arbitrary precision real arithmetic: design and algorithms, *J. Symbolic Computation, 1996,* **11**

Mu88. N.Th. Müller, Untersuchungen zur Komplexität reeller Funktionen, *Dissertation* (Fern-Universität Hagen, 1988)

Mu93. N.Th. Müller, Polynomial Time Computation of Taylor Series, *Proc. 22 JAIIO - PANEL '93, Part 2, Buenos Aires, 1993,* 259-281
(also available at `http://www.informatik.uni-trier.de/~mueller`)

Mu96. N.Th. Müller, Towards a real Real RAM: a Prototype using `C++`, (preliminary version), *Second Workshop on Constructivity and Complexity in Analysis, Forschungsbericht Mathematik-Informatik, Universität Trier 96-44*, Seiten 59-66 (1996)
(also available at `http://www.informatik.uni-trier.de/~mueller`)

Mu97. N.Th. Müller, Towards a real RealRAM: a Prototype using `C++`, *Proc. 6th International Conference on Numerical Analysis*, *Plovdiv, 1997*

Mu98. N.Th. Müller, Implementing limits in an interactive RealRAM, *3rd Conference on Real Numbers and Computers, 1998, Paris,* 13-26

Rio94. M. Riordan, `ftp://ripem.msu.edu/pub/bignum/BIGNUMS.TXT`

Sch90. A. Schönhage, Numerik analytischer Funktionen und Komplexität, *Jber. d. Dt. Math.-Verein.* 92 (1990) 1-20

TZ99. J.V. Tucker, J.I. Zucker, Computation by 'While' programs on topological partial algebras, *Theoretical Computer Science* 219 (1999) 379-420

We87. K. Weihrauch, Computability (volume 9 of: *EATCS Monographs on Theoretical Computer Science*), (Springer, Berlin, 1987)

We95. K. Weihrauch, A Simple Introduction to Computable Analysis, *Informatik Berichte 171 - 2/1995, FernUniversität Hagen*

We97. K. Weihrauch, A Foundation for Computable Analysis, *Proc. DMTCS´96*, (Springer, Singapore, 1997) 66-89

Zi00. P. Zimmermann, MPFR: A Library for Multiprecision Floating-Point Arithmetic with Exact Rounding, *4th Conference on Real Numbers and Computers, 2000, Dagstuhl, 89-90*, see also `http://www.loria.fr/projets/mpfr/`

The Uniformity Conjecture

Daniel Richardson

Mathematics, Bath University, Bath BA2 7AY, UK
dsr@maths.bath.ac.uk

Abstract. This paper discusses the relationship between the syntactic length of expressions built up from the integers using field operations, radicals and exponentials and logarithms, and the smallness of non zero complex numbers defined by such expressions. The Uniformity Conjecture claims that if the expressions are written in an expanded form in which all the arguments of the exponential function have absolute value bounded by 1, then a small multiple of the syntactic length gives a bound for the number of decimal places needed to distinguish the defined number from zero. The consequences of this conjecture are compared with some known results about closeness of approximation from Liouville, Baker, Waldschmidt, Thue-Siegel-Roth. A few of many practical computational consequences are stated. Also the problem of searching for a possible counterexample to the Uniformity Conjecture is discussed and some preliminary results are given.

1 Introduction

The nested radical and exponential-logarithmic expressions are, roughly speaking, those which can be constructed from expressions for the integers using the operators $\{+, -, *, /, \sqrt[n]{}, \exp, \log\}$.

In the section below, the family of nested radical exponential-logarithmic expressions is described and the field of closed form numbers is defined. An expanded form is defined for the expressions, and the Uniformity Conjecture is stated. This claims that for expressions in expanded form, a small multiple of the syntactic length bounds the number of decimal places needed to distinguish the defined number from zero, if it is non zero.

In the following section, some consequences of the Uniformity Conjecture are compared with known, fairly simple, approximation estimates.

In the next section, a few of the many practical computational consequences of the Uniformity Conjecture are stated.

In the final section, the problem of searching for a counterexample to the Uniformity Conjecture is discussed, and some preliminary results are given.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 253–272, 2001.

2 Closed Form Numbers and the Uniformity Conjecture

2.1 Expressions

We assume, to begin with, some canonical representation for the integers. Then the set of nested radical exponential and logarithmic expressions is the smallest set of expressions so that:

1. All the representations of the integers are in the set.
2. If A and B are in the set so are $A + B$, $A - B$, $A * B$, and A/B.
3. If A is in the set, so are (A), $-A$, $\exp(A)$ and $\log(A)$
4. If A is in the set and n is a canonical representation of a natural number bigger than 1, then $A^{1/n}$ is in the set.

Each nested radical exponential and logarithmic expression E is either undefined, or is interpreted as a real or complex number $V(E)$, as follows.

1. If E is a representation of an integer, $V(E)$ is that integer.
2. The operators are given the usual precedence in the absence of brackets.
3. If A and B are defined, then $V(A+B), V(A-B), V(A*B)$ and $V(-A)$ are defined with the usual interpretation of the operators. If A is defined, and B is defined and $V(B)$ is not zero, then $V(A/B)$ is defined, with the usual interpretation, as $V(A)$ divided by $V(B)$. $V((A))$ is the same as $V(A)$.
4. If A is defined, then $\exp(A)$ is defined with meaning e^A.
5. If A is defined, and $V(A) \neq 0$, then $\log(A)$ is defined, as the branch of the logarithm base e so that $-\pi < ImaginaryPart(\log(A)) \leq \pi$.
6. If A is defined and and $V(A) \neq 0$, then $A^{1/n}$ is defined and equal to $exp(\log(A)/n)$.

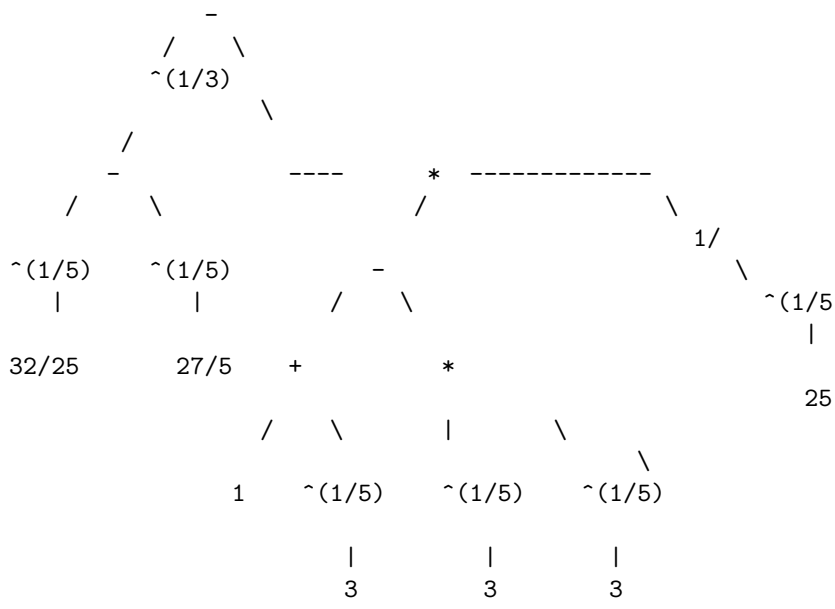
The operator V is called evaluation.

The complex numbers defined in this way are called *closed form* ([Chow]). We define the *real closed form numbers* to be the closed form numbers which happen to have zero imaginary part. Thus the real closed form numbers are closed under trigonometric functions as well as exponentiation and logarithms.

A field with good closure properties including the closed form numbers is the field of elementary numbers. These are numbers of the form $q(\alpha)$, where q is in $\boldsymbol{Q}[x_1, \ldots, x_n]$, and $\alpha \in \boldsymbol{C}^n$ is a non singular solution of a system of equations $(p_1, \ldots, p_n) = 0$ and each p_i is in $Z[x_1, \ldots, x_n, e^{x_1}, \ldots, e^{x_n}]$. It has been shown that this is an effective field if the Schanuel conjecture is true. See [Richardson], [Richardson2].

The nested radical exponential and logarithmic expressions which do not use either the exponential or the logarithm will be called radical expressions. The numbers defined by radicals are the closed form numbers represented by such expressions.

The nested radical and exponential-logarithmic expressions can also be written as expression trees. Such trees have representations of integers at the leaves, and have interior nodes labelled with operator symbols. So, for example,



We will regard the expression trees and the expressions as equivalent from now on, since the translation from one form to the other is straightforward.

Please notice that although we have expressions here for n th roots, we do not have expressions for n th powers. If we want A^2, for example, we have to write it as $A * A$.

2.2 Length of an Expression

Our set of nested radical exponential and logarithmic expressions depends on a choice of canonical representation for the integers. Assume that we have chosen some base b for representation of the integers. We define the length of a non negative integer in this representation to be the number of digits base b which are used.

We define the *length* of a nested radical exponential and logarithmic expression by the following:

$$
\begin{aligned}
&length(A + B) = length(A) + length(B) + 1\\
&length(A - B) = length(A) + length(B) + 1\\
&length(-A) = length(A) + 1\\
&length(A * B) = length(A) + length(B) + 1\\
&length(A/B) = length(A) + length(B) + 1\\
&length((A)) = length(A) + 2\\
&length(A^{1/n}) = length(A) + length(n) + 1
\end{aligned}
$$

$length(e^A) = length(A) + 1$
$length(\log(A)) = length(A) + 1.$

So, for example, in decimal notation, $4 - 3 * (10)^{1/8}$ would have length 10, since it has 5 digits and 3 operator symbols and two (unnecessary) brackets.

2.3 Expressions with Variables

All of the above may be generalised by allowing a certain set $x_1, \ldots, x_k$ of variables to appear on the leaves of the expression trees, as well as representations of integers. Of course, in this case, the expressions represent partially defined complex valued functions of k complex variables. The same notion of length can be applied to such expressions, provided that we define the length of the variables in some way. We might take $length(x_i) = i$ for each i.

2.4 Gap Functions

Definition 1. *A gap function for the closed form numbers is a function* $g : Exp \to \boldsymbol{R}_+$, *where Exp is the set of nested radical exponential and logarithmic expressions, so that if* x *is a closed form number represented by an expression* A, *and* $x \neq 0$ *then*
$|x| > 10^{-g(A)}$

A gap function tells us the amount of decimal precision which is needed to distinguish a non zero number from zero. Of course gap functions exist. We hope that there is a computable gap function, and even an easily computable gap function.

We have a natural scale, i.e. length, on expressions; and we also have a natural scale, i.e. the logarithm of the absolute value, on the complex numbers which are non zero. The central question of this article is: How does the evaluation operator, V, behave with respect to these two scales?

2.5 Uniformity Conjecture

Using iterated exponentiation, it is possible to define very large numbers. Since we have division, it is also possible to define very small numbers with expressions involving iterated exponentiation, followed by division. There does not seem to be any other way to get very large numbers, or very small non zero numbers. (Note that although we allow *nth* roots, we do not have *nth* powers. So we can not easily write a short expression for the result of a sequence of repeated squarings, for example.)

We consider an expression E to be a subexpression of itself.

We will say that an expression E is in expanded form if for any exponential subexpression e^A of E, we have $|V(A)| \leq 1$.

It appears to be true that it is not possible to define any very large numbers, or any very small non zero numbers using small expressions in expanded form.

The original motivation for this paper is an attempt to understand this somewhat vague idea.

Uniformity Conjecture: *If E is an expression in expanded form, and $V(E) \neq 0$, then $|V(E)|$ is bigger than $1/N(k)$, where k is the length of E, and $N(k)$ is the number of syntactically correct expanded form expressions of length $\leq k$.*

Expressions of length less than k can be coded as sequences of symbols of length k by padding with an initial sequence of right brackets. Therefore, if we have S symbols for operators and digits and brackets in our alphabet, the number of syntactically correct expressions of length $\leq k$ is bounded by S^k. In case we use decimal notation, for example, S would be 19 (counting the two brackets). In binary notation, S would be 11.

Roughly speaking, the conjecture says that the amount of base S precision which is needed to discriminate the value of an expanded form expression from zero is proportional to the length of the expression.

We will say that a number occurs at level k if it has an expanded form expression representing it with length no more than k.

A weaker form of the uniformity conjecture may be obtained by replacing $N(k)$ by $N(C * k)$ for some universal constant C. A reason why at least this weaker form of the conjecture seems plausible is that if a number α occurs at level j, then the neighbourhood of 0 at level k may be translated to a subset of the neighbourhood of α at level $k + j + 1$, since if α is represented by A, and a number ϵ is represented by E, then $\alpha + \epsilon$ is represented by $A + E$; and similarly a neighbourhood of α at one level translates to a subset of a neighbourhood of 0 at a higher level, and therefore we would expect all the neighbourhoods of closed form numbers, graded by level, to look alike, except for a change of scale.

The expressions I know which have small evaluations are not even close to being counterexamples. $4/3 - 10^{1/8}$ is zero to three decimal places; $7 \log 2 - 3 \log 5$ is zero to 2 decimal places, for example. Even if we add π as a constant to our language, I still do not know a counterexample in the enlarged language. The famous $3 * \log(640320)/\sqrt{163} - \pi$ is only zero to 15 decimal places, whereas its syntactic length is 15 plus the length of π. We can replace π by $\log(-1)/(-1)^{1/2}$, in which case the length of π would be 8.

It has been pointed out that the Uniformity conjecture is related to a family of conjectures, called Witness conjectures, originating with Joris van der Hoeven. See [van der Hoeven 95], [van der Hoeven 97], [van der Hoeven 2000]. For the family of constants we are considering, the witness conjectures may be stated as follows. For each positive rational number N, denote by Ξ_N the set of all nested radical exponential and logarithmic expressions A such that if E is any subexpression of A, then $N^{-1} \leq |V(E)| \leq N$. The Witness conjectures say that if A is in Ξ_N, then there is a function $\omega_N(n)$ so that $|V(A)| \leq e^{-\omega_N(n)}$, where n is the length of A. The strong witness conjecture is that we can take $\omega_N(n) = Kn$, where K depends on N, and the weak witness conjecture is that we can take $\omega_N(n) = K^n$, where K depends on N. Of course the Uniformity Conjecture is much more specific than either of the Witness Conjectures, since van der Hoeven

does not attempt to estimate K. An especially interesting feature of the recent preprint is an attempt to extend the witness conjectures to a much wider class of constants. Although the Uniformity Conjecture and the Witness Conjectures were stated independently, it seems that the Uniformity Conjecture should be regarded as an extreme form of the Witness Conjecture. In the earlier paper by Van Der Hoeven referred to, I find the comment that some form of Witness Conjecture might imply the Schanuel Conjecture. I think this is possible, but do not see a proof. There is some discussion of this at the end of this paper.

Note that we can replace any expression which defines a value by an equivalent one in expanded form. But this is unlikely to be a good way to decide zero equivalence over the set of all nested radical exponential and logarithmic expressions. Joris van der Hoeven ([van der Hoeven 95]) suggests that expressions not in expanded form could be put into some sort of asymptotic form, as if the large numbers were exp-log functions which tended to infinity. I imagine that this is a good idea. For the rest of this article, however, we will just ignore expressions not in expanded form.

3 Mahler, Liouville, Baker, Waldschmidt Estimates

3.1 Norm Estimate

Theorem 1. *Suppose α is an algebraic number with defining polynomial*
$a_d x^d + \cdots + a_0 = a_d(x - \alpha_1) \ldots (x - \alpha_d)$
in $\boldsymbol{Z}[x]$. Then

$$\log \alpha \geq -d \; Maximum(\log |a_d| , \log |\alpha_1| , \ldots , \log |\alpha_d|).$$

Proof.
$|a_d \alpha_1 \ldots \alpha_d| = |a_0|$, and a_0 is a non zero integer, since the defining polynomial is chosen with minimal degree. So
$|a_d \alpha_1 \ldots \alpha_d| \geq 1$
$\log |a_d| + \log |\alpha_1| + \ldots \log |\alpha_d| \geq 0$
and the theorem follows from this.
□

3.2 Mahler Measure

Let $p(x)$ be a polynomial in one variable with complex coefficients. Suppose

$$p(x) = a_d x^d + \cdots + a_0 = a_d(x - z_1) \ldots (x - z_d)$$

where $z_1 \ldots z_d$ are the roots of $p(x)$. Define the Mahler measure of $p(x)$ by

$$M(p) = |a_d| \prod_{j=1}^{d} max(1, |z_j|)$$

We have, in general, $M(x^d p(1/x)) = M(p)$ and $M(p * q) = M(p) * M(q)$. Also, for any positive integer k, $M(p(x^k)) = M(p(x))$.

In case α is an algebraic number, we define $M(\alpha) = M(p)$, where $p(x)$ is a minimal defining polynomial for α in $\boldsymbol{Z}[x]$.

Lemma 1. *If α and β are algebraic, with degrees d_1 and d_2 respectively, then*

1. $1/M(\alpha) \leq |\alpha| \leq M(\alpha)$
2. $M(\alpha * \beta) \leq M(\alpha)^{d_2} M(\beta)^{d_1}$
3. $M(\alpha + \beta) \leq 2^{d_1 d_2} M(\alpha)^{d_2} M(\beta)^{d_1}$
4. *If k is a positive integer,* $M(\alpha^{1/k}) \leq M(\alpha)$
5. *If k is a positive integer,* $M(\alpha^k) \leq M(\alpha)^k$
6. $M(1/\alpha) = M(\alpha)$

Proof. All of this is standard, using the fact that the defining polynomials have integral coefficients. See [Mignotte], pp 148 ff.

Define the logarithmic Mahler measure to be the logarithm base 2 of the Mahler measure.

3.3 Liouville Estimate

Define the logarithmic height, $h(p)$, of a polynomial p with integral coefficients to be the logarithm of the maximum of the absolute values of the coefficients. Define the logarithmic height of an algebraic number to be the logarithmic height of the minimal defining polynomial in $\boldsymbol{Z}[x]$. We have:

Theorem 2. *Liouville Estimate. Let $\xi_1, \ldots, \xi_m$ be algebraic numbers, of degrees $d_1, \ldots, d_m$ and logarithmic heights $h_1, \ldots, h_m$ respectively. Let d be the degree of the extension $\boldsymbol{Q}(\xi_1, \ldots, \xi_m)$ over $\boldsymbol{Q}$. Let P be a polynomial in $\boldsymbol{Z}[x_1, \ldots, x_m]$, of degree N_i in x_i. If $P(\xi_1, \ldots, \xi_m) \neq 0$, then*

$$-d(h(P) + \sum_{i=1}^{m} (N_i h_i / d_i) + 2N_i) \leq \log |P(\xi_1, \ldots, \xi_m)| .$$

For proof, see [Lang].

3.4 Examples

Consider a linear form in radicals.

$\lambda = b_1 (a_1)^{1/n_1} + b_2 (a_2)^{1/n_2} + \cdots + b_k (a_k)^{1/n_k} + b_{k+1}$, with $b_i \in \boldsymbol{Z}$, and n_i, a_i positive natural numbers for all i. Assume $\lambda \neq 0$. Then we get the following contrasting estimates. (Note that logarithms here are base 2.)

1. Norm estimate and Mahler measure. We have $deg(\lambda) \leq \prod n_i$, and the denominator of λ is 1. So we get

$$\log|\lambda| \geq -(\prod n_i)(\log(|b_{k+1}| + \sum_{i \leq k} |b_i|\,|a_i|^{1/n_i})).$$

2. Liouville Estimate.

$$\log|\lambda| \geq -(\prod n_i)(\log Max_i(|b_i|) + \sum_i \log(|a_i|)/n_i + 2k).$$

3. Uniformity Conjecture.

$$\log|\lambda| \geq -C(\sum \log|n_i| + \sum \log|a_i| + \sum \log|b_i| + 3k),$$

where $C \leq \log(19)$.

The most striking feature of the above is that in all the earlier estimates the precision needed increases as the product of the radical degrees; but according to the uniformity conjecture all that is needed is the sum of the logarithms. Thus even in this simple algebraic case, the uniformity conjecture would imply quite strong new results.

3.5 Baker-Waldschmidt

See [De Weger].

Theorem 3. *(Baker-Waldschmidt) Suppose* $p_1, \ldots, p_n \in \boldsymbol{Z}$ *with* $2 \leq p_1 < \cdots < p_n$ *and that* $(\boldsymbol{Q}(p_1^{1/2}, \ldots, p_n^{1/2}) : \boldsymbol{Q}) = 2^n$. *Let* $b_1 \ldots, b_n \in \boldsymbol{Z}$, *with* $B = Max_{1 \leq i \leq n} |b_i|$. *Suppose* $\lambda = b_1 \log p_1 + \cdots + b_n \log p_n \neq 0$. *Let* $V_i = Max(1, \log p_i)$, *for* $i = 1, \ldots, n$. *Let* $\omega = V_1 \ldots V_n$. *Let*

$$C_1 = 2^{9n+26} n^{n+4} \omega \log(eV_{n-1})$$

$$C_2 = C_1 \log(eV_n).$$

Then

$$|\lambda| > e^{-(C_1 \log B + C_2)}.$$

This should be compared with the uniformity conjecture, which predicts:

$$|\lambda| > 19^{-(\sum length(p_i) + length(b_i) + 2n - 1)}$$

In this case, the most striking difference is that the coefficient C_1 in the Baker-Waldschmidt result increases as the product of the lengths of the p_i, whereas in the prediction from the uniformity conjecture, only the sum of the lengths appears.

Patrice Philippon relates a similar lower bound on linear forms in logarithms with the abc conjecture. See [PP], [PP2].

3.6 Thue-Siegel-Roth

See [HS], [Sprindzhuk].

Theorem 4. *(Thue-Siegel-Roth) For any algebraic number α and for any $\delta > 0$*

$$|p/q - \alpha| > q^{-(2+\delta)}$$

for all but finitely many (p/q) in $\boldsymbol{Q}$.

The uniformity conjecture would predict that if A is an expression representing α then:

$|p/q - \alpha| > 1/N(k)$, where k is the length of $p/q - A$, ie $k = length(p) + length(q) + length(A) + 2)$

provided $\alpha \neq p/q$. In this case the inequality directly implied by the uniformity conjecture is weaker than that of Thue-Siegel-Roth, but does not allow exceptions. So even in this case, the uniformity conjecture would imply new results. On the other hand, the Thue-Siegel-Roth theorem implies a bound on the number of counterexamples to the uniformity conjecture of the form $\alpha - p/q$, for fixed algebraic closed form α.

Suppose we represent integers base 10, and therefore count length base 10.

Proposition 1. *Suppose α is an algebraic closed form number represented by an expression A. Then there are at most only finitely many p/q in $\boldsymbol{Q}$ so that*
$|\alpha - p/q| < 1/N(k)$ *with* $k = length(p) + length(q) + 2 + length(A)$.

Proof. Suppose, for the sake of a contradiction, that α is a fixed algebraic closed form number and that there are infinitely many counterexamples to the uniformity conjecture of the form

$$\alpha - p/q$$

Let (p_i/q_i) be a sequence of rationals, with $q_i \to \infty$, and $k_i = length(p_i) + length(q_i) + length(\alpha) + 2$ and

$$|\alpha - p_i/q_i| < 1/N(k_i).$$

We have $N(k) \geq 10^{k(1+\epsilon)}$ for some $\epsilon > 0$.
Also $p_i/q_i \to \alpha$ and $\alpha \neq 0$, and $length(q_i) \to \infty$, so $length(p_i)/length(q_i) \to 1$.
Thus
$|\alpha - p_i/q_i| < 10^{-(1+\epsilon/2)2(length(q_i)+1)}$ for sufficiently large i.
Thus for some $\epsilon > 0$ and for infinitely many p/q in $\boldsymbol{Q}$
$|\alpha - p/q| < q^{-(2+\epsilon)}$, contradicting Thue-Siegel-Roth.

Thus although there may be counterexamples to the uniformity conjecture of the form $a^{1/n} - p/q$, we should not expect very many of them.

4 Computational Consequences

4.1 Lazy Exact Computation

We hope eventually to extend the possibility of exact computation beyond the field of rational numbers in such a way that each expression for a real number can easily be approximated to any desired precision. One important step in this direction would be to understand how to compute with the nested radical and exponential-logarithmic expressions.

Even without the exponential and the logarithm, basic questions about nested radical expressions may seem quite difficult to decide. In particular, equality is not well understood, as illustrated by the following examples, from [Davenport, p93].

$$\sqrt{5+2\sqrt{6}}+\sqrt{5-2\sqrt{6}} = 2\sqrt{3}$$

$$\sqrt{16-2\sqrt{29}+2\sqrt{55-10\sqrt{29}}} = \sqrt{22+2\sqrt{5}} - \sqrt{11+2\sqrt{29}}+\sqrt{5}$$

$$\sqrt[3]{\sqrt[5]{32/5}-\sqrt[5]{27/5}} = (1+\sqrt[5]{3}-\sqrt[5]{3}^2)/\sqrt[5]{25}$$

$$\sqrt{112+70\sqrt{2}+(46+34\sqrt{2})\sqrt{5}} = 5+4\sqrt{2}+(3+\sqrt{2})\sqrt{5}$$

Open Problem 1. *What is the computational difficulty of deciding $V(A) = 0$, where A is a radical expression, i.e. a radical exponential logarithmic expression with no exponential or logarithmic terms?*

See [Strsebonski], [Pohst2], [Loos], [Johnson], [Hur], [Langley] for some methods currently used or proposed to solve such problems.

In particular, it would be interesting to know whether or not the zero equivalence problem for radical expressions can be solved in a number of steps which is bounded by some polynomial in the length of the expression. In other words, is this basic problem *tractable* in the sense of Blum-Smale-Shub? See [BSS].

We would also like to know:

Open Problem 2. *What is the computational complexity of finding $V(A)$ to n decimal places, given nested radical exponential and logarithmic expression A ?*

See proposition 3 of this article for a very crude upper bound.

Note that for this question we want the complexity in terms of the two input parameters, $length(A)$ and n. We know that for fixed A, we can approximate to n decimal places in a number of steps bounded by $O(\log(n)m(n))$, where $m(n)$ is the complexity of multiplication of n digit numbers. See [Borwein].

In any case, it is clear that the Uniformity conjecture implies easy decidability of the zero equivalence problem for closed nested radical exponential and logarithmic expressions.

4.2 Inverse Symbolic Calculation

The uniformity conjecture implies that the decimal approximation of precision $\log_{10}(19) * 2n$ is a code for the closed form expression, with integers in decimal notation, and length n, with that value. This implies the existence of a potentially quite useful inverse symbolic calculator, that is a method of deriving a closed form expression from a decimal approximation.

4.3 Unwound Straight Line Programs.

Consider expression trees with integers or variables on the leaves and internal nodes labelled with operators for addition, subtraction and multiplication. We will call these unwound straight line programs. This is a compact way to represent multivariate polynomials. One of the difficult problems in this area is how efficiently to recognise the zero polynomial. (See [Zippel] for a discussion of this problem.) This could be done by substitution of algebraically independent values. For example, if $p_1, \ldots, p_n$ are the first n prime numbers, we know that $e^{1/\sqrt{(p_1)}}, \ldots, e^{1/\sqrt{(p_n)}}$ are algebraically independent. If we accept the Schanuel conjecture, we know that $\log(p_1), \ldots, \log(p_n)$ are algebraically independent. Thus, assuming the Schanuel conjecture, if the variables are $(x_1, \ldots, x_n)$, we could substitute $(\log(p_1), \ldots, \log(p_n))$, where $(p_1, \ldots, p_n)$ are the first n primes. If k is the length of the resulting nested exponential and logarithmic expression, we could recognise whether or not the original polynomial was zero by approximation with decimal precision $\log_{10}(19)k$.

Suppose expression tree T represents polynomial P_T in $\boldsymbol{Z}[x_1, \ldots, x_n]$. Define $h(T)$ to be the maximum length of the integers on the frontier of T. Define $s(T)$ to be the number of nodes in T. Define A_T to be the nested radical and exponential-logarithmic expression obtained by substituting $\log(p_i)$ for x_i, for $i = 1 \ldots n$, where p_i is the i th prime. Note that A_T is in expanded form. The length of p_i is no more than i. Thus the length of A_T is bounded by

$s(T) + s(T)h(T) + s(T)n$

So far we have:

Proposition 2. *(Assuming the Uniformity Conjecture and the Schanuel Conjecture.) P_T is the zero polynomial if and only if $|V(A_T)| < 1/N(k)$ where $k = s(T) + s(T)h(T) + s(T)n$.*

Also,

Proposition 3. *(Assuming the Uniformity Conjecture and the Schanuel Conjecture.) The computational bit complexity of deciding whether or not P_T is the zero polynomial is bounded by*

$O(s(T)\log(k)m(s(T)k))$

where $k = s(T) + s(T)h(T) + s(T)n$

and $m(k)$ is the complexity of multiplication of two k digit numbers.

Proof. We need $O(k)$ decimal places for the output of A_T so that we can apply the previous proposition.

The value output by a tree A_T, or by any of its subtrees is bounded in absolute value by $10^{O(k)}$. This means that at most $O(k)$ decimal places of precision can be lost at each computation step. So at most $s(T)O(k)$ decimal places of precision are lost in the whole computation.

Thus we need to do the whole computation with $O(s(T)k)$ decimal places.

We can get j decimal places for each $\log p_i$ in $O(m(j)\log j)$ steps, where $m(j)$ is the complexity of multiplication of j digit numbers. So we can get the necessary $O(s(T)k)$ decimal places for these n logarithms in $O(s(T)\log(k)m(s(T)k))$ steps, since $n < s(T) < k$. Addition, subtraction and multiplication to j decimal places can be done in $O(m(j))$ steps. So the rest of the computation can be done in $O(s(T)\log(k)m(s(T)k)$ steps. Thus the whole computation can be done in a number of steps bounded by $O(s(T)\log(k)m(s(T)k))$.

Of course this could be done without the Schanuel conjecture, using another substitution.

5 Search for a Counterexample to the Uniformity Conjecture

5.1 Good Rational Approximations

If α is real we can look for good rational approximations to α by searching for large coefficients in the continued fraction expansion of α. In this way we might find counterexamples or near counterexamples to the Uniformity conjecture of the form $\alpha - p/q$.

5.2 Near Integer Relations

If a is a vector of n real numbers, an integer relation for a is a non zero integer vector c so that $c^T a = 0$. A near integer relation is a non zero integer vector c so that $c^T a$ has "small" absolute value.

We can make the notion of "small" precise if a consists of closed form numbers. Define $length(a)$ to be the sum of the lengths of the components, considering these as closed form numbers. We will say that $c^T a$ is small if

$$\left|c^T a\right| < 10^{-(length(c)+length(a)+2n-1)}$$

These near integer relations include all possible counterexamples to the uniformity conjecture of the form $c^T a$.

For fixed a, we can search for such near integer relations using the LLL algorithm.

Let the Euclidean length of a vector $v \in \boldsymbol{R}^n$ be, as usual $(\sum v_i^2)^{1/2}$.

Suppose $B = \{v^{(1)}, \ldots, v^{(n)}\}$ is a set of n vectors in $\boldsymbol{R}^n$ which are linearly independent over $\boldsymbol{R}$. By the lattice spanned by B we mean the set

$$L = \sum r_i v^{(i)} : r_i \in \boldsymbol{Z}$$

This lattice has dimension n and rank n. The determinant of the lattice, $det(L)$ is the determinant of any matrix of real numbers whose rows span the lattice. This is independent of basis.

A lattice reduction algorithm, such as LLL, finds a reduced basis for a given lattice, the first vector of which has Euclidean length at most $2^{(n-1)/2}m(L)$, where $m(L)$ is the Euclidean length of the shortest non zero vector in the lattice. The first vector in a reduced basis also has Euclidean length bounded by $2^{(n-1)/4}det(L)^{1/n}$. For definition of reduced basis and proofs of basic properties, see [HJLS], [HST].

This can be used to find near integer relations in the following way. Suppose we are given $a = (a_1, \ldots, a_n)$, and let $(a'_1, \ldots, a'_n)$ be a vector of their rational approximations. Let M be a large rational constant. Let A be the n by n matrix which is obtained from the n by n identity matrix by replacing the last column by the transpose of $(Ma'_1, \ldots, Ma'_n)$. Consider the lattice L spanned by the n dimensional vectors $v^{(i)}, 1 \leq i \leq n$, which are the rows of the matrix A.

If V is a short vector in a reduced basis for L

$$V = (w_1, \ldots, w_{n-1}, M\sum_{i \leq n} w_i a'_i)$$

then

$$\left|\sum w_i a'_i\right|$$

is small and thus $(w_1, \ldots, w_n)$ is a good candidate for a near integer relation for $(a_1, \ldots, a_n)$.

The problem now is to pick the precision of the approximation and the size of the multiplier M in such a way the first vector in a reduced basis can be used to indicate the non existence of small $c^T a$ with some conditions on c. We will fix $length(c)$ and also put an upper bound on the length of each component of c.

Assume that $(a_1, \ldots, a_n)$ is ordered so that $|a_n|$ is maximal among $|a_i|$. The determinant of the lattice is Ma_n.

Suppose we are looking for small $c^T a$ with $Max(length(c_i) = m$, and thus $length(c) \leq nm$. Let

$$M = 10^{length(c)+m+length(a)+2n-1}.$$

Approximate a with precision more than $(n+1)m + length(a) + 2n - 1$. Find the reduced basis. If b_1 is the first vector in the reduced basis, we expect

$$|b_1| \leq 2^{(n-1)/4}(Ma_n)^{1/n}.$$

However a small $c^T a$ with $length(c_i) \leq m$ for all i would imply

$c^T a < 10^{-(length(c)+length(a)+2n-1)}$, and thus each component of the shortest vector in the lattice would have length no more than m, and thus

$$|b_1| \leq 2^{(n-1)/2} n^{1/2} 10^m.$$

Therefore, if this condition does not happen, we can eliminate the possibility of small $c^T a$ of this kind. This can be used to eliminate some parts of the search space for counterexamples.

5.3 Various Results

Assorted Near Counterexamples. $12^{(1/17)} - 4015/3496$. This is 0 to 10 decimal places. Quite small, but not small enough. I found this by looking for large numbers in the continued fraction expansions of n th roots.

$17^{(1/25)} - 28/25$ is zero to 6 decimal places. Also found with continued fractions.

$3 * \log(640320)/\sqrt{163} - \pi$. This is famous for being small, but it is only 0 to 15 decimal places.

$\pi - 355/113$, zero to 6 decimal places, could be found by looking for large numbers in the continued fraction expansion of π, and is also well known.

$199\log(3) + 142\log(7) - 183$ is zero to 8 decimal places. This was found with LLL.

In [De Weger] LLL is used to find small linear forms in logarithms, and many results are tabulated. No counterexample to the uniformity conjecture is found.

Some Examples Found by Students. Students in my computer algebra course examined $a^{1/n} - p_k/q_k \neq 0$ for natural numbers a between 2 and 100, and natural numbers n between 2 and 100, with p_k/q_k the kth convergent of the continued fraction approximation for values of k up to 80, and found no counterexamples to the uniformity conjecture. Some small numbers they found were:

$2^{1/8} - 494/453$, which is 0 to 6 decimal places ; $3^{1/14} - 53/49$, which is 0 to 6 decimal places ; $3^{1/6} - 6/5$, which is 0 to 4 decimal places ; $7^{1/8} - 311830/244501$, which is 0 to 14 decimal places ; $11^{1/5} - 21/13$, which is 0 to 5 decimal places ; $11^{1/6} - 6772/4541$, which is 0 to 10 decimal places ; $14^{1/95} - 73/71$, which is 0 to 7 decimal places ; $21^{1/20} - 7243/6958$, which is 0 to 11 decimal places ; $27^{1/8} - 77/51$, which is 0 to 6 decimal places ; $31^{1/97} - 115/111$, which is 0 to 8 decimal places ; $38^{1/50} - 92/83$, which is 0 to 8 decimal places ; $61^{1/11} - 93/64$, which is 0 to 6 decimal places ; $65^{1/11} - 19/13$, which is 0 to 6 decimal places ; $89^{1/6} - 374/177$, which is 0 to 9 decimal places ; $109^{1/5} - 23/9$, which is 0 to 6 decimal places ; $544^{1/6} - 20/7$, which is 0 to 6 decimal places ; $823^{1/6} - 1849/604$, which is 0 to 13 decimal places.

The following were found with LLL:

$143\log(3) - 183\log(7) + 199$ is zero to 8 decimal places.

$4\log(3) + 13^{1/2} - 8$ is zero to 6 decimal places.

Thanks to John Hillier, Simon Attfield, Mark Wilcox, Mark Waddingham, Steven Olney, Jonathan Brown, Jake Holdridge and Emma Jones for these examples.

Linear Forms in Radicals: An Example. Consider the linear form

$$\lambda = c_1\sqrt{2} + c_2\sqrt{3} + c_3.$$

Suppose there were a counterexample to the uniformity conjecture with the $length(\lambda) = k + 8$, where k is the sum of the lengths of c_i, which are assumed to be integral. Let m be the maximum length of the coefficients c_i. In the counterexample, λ must be small, and so

$2m \leq k \leq 3m$

Let $j = k + 8 + m$, and take LLL multiplier $M = 10^j$, as described in the previous section. Then if b_1 is the first vector in a reduced basis, we must have

$$|b_1| \leq 2\sqrt{(}3)10^m.$$

An upper bound for m is $(j - 8)/3$. The LLL algorithm was run for $j = 1, ...50$. The results all violate this constraint, and thus show that there is no counterexample of this form with coefficients using up to 12 digits. This is quite an easy computation, and it is clear that other regions of the search space could be investigated in this way.

5.4 A Confession

The search for counterexamples has so far concentrated on linear forms in a few simple terms. This is because we have reasonably good tools for such a search. I have been trying to eliminate the possibility that the unfound items are under the street lamp since I don't know how to search anywhere else.

6 Extension to the Elementary Numbers

An *elementary number* is a number of the form $q(\alpha)$ where $q \in \boldsymbol{Z}[x_1, \ldots, x_n]$, and α is a point in $\boldsymbol{C}^n$ which is a non singular solution of a system of equations $S(x) = 0$, and $S = (p_1, \ldots, p_n)$ with each p_i a polynomial with integral coefficients in $x_1, e^{x_1}, \ldots, x_n, e^{x_n}$.

Such a number will be defined by:

$$\eta = q(\alpha), S(\alpha) = 0, \alpha \in B,$$

where B is a ball in $\boldsymbol{C}^n$ which contains the point α and which does not contain any other solution of $S(x) = 0$. (The assumption that α is a non singular solution means that the Jacobian of S, evaluated at α is non zero.)

We will say that such a definition is in expanded form if B is the unit ball around the origin in $\boldsymbol{C}^n$.

Define the length of a power product $x_1^{j_1} \ldots x_n^{j_n} e^{k_1 x_1} \ldots e^{k_n x_n}$ to be the sum of the lengths of the degrees, $length(j_1) + \cdots + length(j_n) + length(k_1) + \cdots + length(k_n)$. Define the length of a monomial to be the length of the coefficient plus the length of the power product. Define the length of a sum of k monomials to be the sum of the lengths of the monomials. This gives a definition of length for all polynomials in $\boldsymbol{Z}[x_1, e^{x_1}, \ldots, x_n, e^{x_n}]$

Suppose elementary number η is defined, in expanded form, by:

$$\eta = q(\alpha), S(\alpha) = 0, B.$$

where B is the unit ball around the origin. Define the size of this definition to be the product of the sum of the lengths of the polynomials including q and the product of the total degrees of the polynomials in S.

$$(length(p_1) + \cdots + length(p_n) + length(q))(degree(p_1) * \cdots * degree(p_n)),$$

where $S = (p_1 \ldots, p_n)$.

Uniformity Conjecture for Elementary Numbers. If η is a non zero elementary number and has an expanded form definition of size n, then $|\eta| > 10^{-n}$.

7 Relation to the Schanuel Conjecture

The Schanuel Conjecture. *If $\alpha_1, \ldots, \alpha_m$ are complex numbers which are linearly independent over $\boldsymbol{Q}$, then $(\alpha_1, \ldots, \alpha_m, e^{\alpha_1}, \ldots, e^{\alpha_m})$ has transcendence rank at least m.*

It has been shown that if the Schanuel conjecture is true, then the zero equivalence problem for closed form numbers is decidable. Thus the Schanuel conjecture implies the existence of a computable gap function. See [Richardson], [Richardson2].

It seems that it might be possible to prove part of the Schanuel conjecture, via Gelfond's method, from the uniformity conjecture. For a good exposition of Gelfond's method, see [Lang]. Roughly speaking, Gelfond's method would construct a function $F(t)$ with a large number of roots from the existence of a counterexample to the Schanuel conjecture. A closed form number $F(w)$ would then be found which has a fairly small length but is non zero. The maximum modulus principle (an analytic function defined on an open set including a ball has modulus inside the ball bounded by maximum modulus on the boundary of the ball) is then applied to show that the modulus of $F(w)$ is so small that it contradicts the uniformity conjecture.

It would be nice to show the following.

Conjecture. *If K is a field of closed form numbers with transcendence rank d, and the uniformity conjecture is true for numbers in K, and $\alpha_1 \ldots, \alpha_m$ are in K, and are linearly independent over $\boldsymbol{Q}$, and $m > d$ then $e^{\alpha_1}, \ldots, e^{\alpha_m}$ are not all in K.*

I am not able to prove this or anything like it. Here is a small result which shows how the Gelfond method can be used.

We will say that $(x_1, \ldots, x_d, y)$ is a proper set of generators of a subfield F of $\boldsymbol{C}$ if $x_1, \ldots, x_d$ are algebraically independent and y is algebraic and integral over $\boldsymbol{Z}[x_1, \ldots, x_d]$, and F is $\boldsymbol{Z}(x_1, \ldots, x_d)[y]$. Any number in such a field has a canonical form as a polynomial in $\boldsymbol{Z}(x_1, \ldots, x_d)[y]$ with y-degree less than the degree of y over $\boldsymbol{Z}[x_1, \ldots, x_d]$. If $A_1, \ldots, A_j$ are numbers in K and we wish to solve an equation

$$a_1 A_1 + \cdots + a_j A_j = 0$$

for integers $a_1, \ldots, a_j$, we can do this by putting $A_1, \ldots, A_j$ into canonical form in K and replacing this equation by a system of equations, one for each power product which appears in the canonical forms. The number of such equations is bounded in terms of the degree in $(x_1, \ldots, x_d)$ of $A_1, \ldots, A_j$.

If E is an expression in canonical form for an element of field F, with proper set of generators $(x_1, \ldots, x_d, y)$, let $h(E)$ be the maximum of the lengths of the integral coefficients in E; and let $deg(E)$ be the maximum of the degrees of any x_i in E. The number of integral coefficients in E is then bounded by $O(deg(E)^d)$. Using this, we can bound $h(E_1 * E_2)$ in terms of $h(E_1)$ and $h(E_2)$ and $deg(E_1)$ and $deg(E_2)$.

Lemma 2. *Suppose E_1, E_2, and E_3 are canonical form elements of field F, with proper set of generators. Then*

1. *If $E_3 = E_1 * E_2$ then $deg(E_3) \leq O(deg(E_1) + deg(E_2))$ and $h(E_3) \leq O(h(E_1) + h(E_2) + deg(E_1) + deg(E_2))$*
2. *If $E_2 = E_1^j$, then $deg(E_2) \leq O(j(h(E_1) + deg(E_1)))$ and $h(E_2) \leq O(j(h(E_1) + deg(E_1)))$*

The constant multipliers implied in the big O notation do depend on the defining polynomial for y over $\boldsymbol{Z}[x_1, \ldots, x_d]$, but can be chosen independent of E_1, E_2, E_3, j. For a proof of this standard result, see [Lang], Ch V, Lemma 1.

Proposition 4. *(Assuming Uniformity Conjecture.) Let K be a field of closed form numbers with transcendence rank d, and proper set of generators $(x_1, \ldots, x_d, y)$. Suppose $c_1, \ldots, c_{d_1}$ are complex numbers which are linearly independent over $\boldsymbol{Q}$, and $\alpha_1, \ldots \alpha_{d_2}$ are complex numbers which are linearly independent over $\boldsymbol{Q}$. Then not all of*

$e^{c_i \alpha_j}$, for $1 \leq i \leq d_1, 1 \leq j \leq d_2$

can be in K, provided $(d_1 + d_2)(d + 1) \leq d_1 d_2$

Proof. For the sake of a contradiction, suppose that $(d_1 + d_2)(d + 1) \leq d_1 d_2$ and also $e^{c_i \alpha_j}$ in K for all i, j. In all the following, d, d_1, d_2 are fixed.

Let n be a large whole number. We will eventually let it go to infinity.

Define S_n to be the set of numbers of the form $\sum_{1 \leq i \leq d_1} n_i c_i$ where each n_i is a natural number with $|n_i| \leq n$. Since the c_i are linearly independent, the number of distinct numbers in S_n is $(n + 1)^{d_1}$.

For $j = 1, \ldots, d_2$, define $f_j(t) = e^{\alpha_j t}$. These functions are multiplicatively independent. Each f_j maps S_n into K. Form

$$F(t) = \sum a_{(i)} x_1^{i_{d_2+1}} \ldots x_d^{i_{d_2+d}} f_1^{i_1} \ldots f_{d_2}^{i_{d_2}}.$$

where $(i) = (i_1, \ldots i_{d_2+d})$, and $1 \leq i_1, \ldots, i_{d_2} \leq r$ and $1 \leq i_{d_2+1}, \ldots, i_{d_2+d} \leq rn$, and $r = cn^{d_1/d_2}$, for some fixed number c.

We now have d, d_1, d_2 fixed, n going to infinity, and r defined in terms of n, depending on fixed c, which will be chosen later, but will be independent of r and n.

We need to solve for integer coefficients $a_{(i)}$, not all zero, so that $F(z) = 0$ for all z in S_n.

The number of unknowns is the number of $a_{(i)}$, that is $r^{d_2}(rn)^d$.

For each z in S_n, we put $F(z)$ in canonical form, with respect to the proper set of generators $(x_1, \ldots, x_d, y)$. We need a bound on the degrees in $(x_1, \ldots, x_n)$ which can occur in this canonical form. For z in S_n, $f_j(z)^{i_j}$ is a product of $O(rn)$ numbers from a fixed finite set, $\{e^{c_i \alpha_j} : i \leq d_1, j \leq d_2\}$, of numbers in K. Therefore, according to the Lemma preceeding, the degrees in $(x_1, \ldots, x_d)$ of $F(z)$, for z in S_n are bounded by $O(rn)$. When we put each equation $F(z) = 0$ into canonical form, we get a system of equations, one for each distinct power product in the canonical form. The number of equations which must be solved is thus bounded by $O(n^{d_1}(rn)^d)$. The integral coefficients in these equations have length bounded by $O(rn)$, according to the Lemma preceeding. Now pick the fixed number c so that the number of unknowns is more than twice the number of equations, and thus so that there is a solution in integers $a_{(i)}$ with length bounded by $O(rn)$.

We know that $F(t)$ is not identically zero. Let s be the first number bigger than n so that for some w in S_{s+1} we have $F(w) \neq 0$. There must be such an s since otherwise $F(t)$ has too many roots, in fact a dense set of roots. (We expect $s = n$.) Consider $F(w)$. We can show that if n is picked sufficiently large, $F(w)$ will be a counterexample to the uniformity conjecture.

Put $F(w)$ in canonical form in $\boldsymbol{Z}(x_1, \ldots, x_d)[y]$. The length of $F(w)$ is bounded by $O((rs)^{d+1})$.

Define

$$G(w,t) = F(t) \prod_{z \in S_s} (w - z) / \prod_{z \in S_s} (t - z)$$

$F(z) = 0$ for all z in S_s.

$G(w,t)$ is analytic, and $F(w) = G(w,w)$. By the maximum modulus principle, the modulus of $F(w)$ is bounded by the modulus of $G(w,t)$ for t on a circle which includes w in its interior.

Consider a circle of radius $R = s^{1+\epsilon}$ for some small positive ϵ. For s sufficiently large, this circle includes all of S_{s+1}.

Assume $|t| = R$. Then

$|(w-z)/(t-z)| < s^{-\epsilon}$ for z in S_s and thus the product of all these numbers is bounded by $s^{-\epsilon(s+1)^{d_1}}$ for sufficiently large s. It follows that

$$\log \left| \prod_{z \in S_s} (w-z) / \prod_{z \in S_s} (t-z) \right| < -K_1 (s^{d_1} \log s)$$

for some constant $K_1 > 0$. Also,

$$\log |F(t)| \leq O(s^{1+\epsilon} r) \leq O(s^{1+\epsilon+d_1/d_2}).$$

since $r = cn^{d_1/d_2}$. This is dominated by $K_1 s^{d_1} \log s$.

We get that $\log |F(w)|$ is bounded above by $-K_2(s^{d_1} \log s)$ for some constant $K_2 > 0$.

On the other hand, the number of monomials in the canonical form of $F(w)$ is bounded by $O(rs)^d$, and the integral coefficients have length bounded by $O(rs)$, and thus the length of $F(w)$ (in canonical form) is bounded by $O((rs)^{d+1})$, and this is bounded by $O(s^{(d_1+d_2)(d+1)/d_2})$.

This either contradicts the uniformity conjecture, or our assumption that $(d_1 + d_2)(d+1) \leq d_1 d_2$.

□

Here is a sample consequence.

Corollary 1. *(Assuming the Uniformity Conjecture.) All the numbers e^{e^n} can not be in $\boldsymbol{Q}(e)$ for $n = 0, 1, \ldots, 6$.*

Proof. Take $d = 1$ and $d_1 = 4, d_2 = 4$ in the proposition, and $c_i = e^{i-1} = \alpha_i$.

References

[Baker] A. Baker, *Transcendental number theory*, CUP, 1975

[Borwein] J.M Borwein and P.B. Borwein, On the Complexity of Familiar Functions and Numbers, SIAM Review, Vol 30, No 4, Dec 1988, pp 589-601

[BSS] Blum, Cucker, Shub, Smale, *Real Complexity and Computation*, Springer

[Chow] T.Y. Chow, What is a Closed-Form Number?, American Mathematical Monthly, Vol 106, No 5, 1999, pp 440-448

[Davenport] J. Davenport, Y. Siret, and E. Tournier, Computer Algebra, Academic Press, 1993.

[De Weger] B.M.M. De Weger, Solving exponential Diophantine equations using lattice basis reduction algorithms, J. Number Theory 26 (1987) 325-367.

[HJLS] J. Håstad, B. Just, J.C. Lagarias, and C.P.Schnorr, Polynomial time algorithms for finding integer relations among real numbers, SIAM J. Comput. 18 (1989). pp 859-881.

[HS] M. Hindry and J. Silverman, Diophantine Geometry, An Introduction, Springer Graduate Texts in Mathematics, 2000

[HST] E. Hlawka, J. Schoissengeier, and R. Taschner, Geometric and Analytic Number Theory. Springer Verlag, 1991

[Hur] Namhyun Hur and James H. Davenport, An exact Real Arithmetic with Equality Determination, draft, (Bath U)

[Johnson] J.R. Johnson, Real Algebraic Number Computation using Interval Arithmetic, ISSAC '92, pp 195-205

[Koiran] P. Koiran, A weak version of the Blum, Shub and Smale Model, FOCS '93, pp 486-495; also in JCSS 54 (1), pp 177-189, 1997

[Langley] S. Langley and D. Richardson, Bounds on Algebraic Expressions, preprint

[Lang] S. Lang, Introduction to Transcendental Numbers, Addison Wesley, 1966

[Loos] R. Loos, Computing in Algebraic Extensions , Computing, 4, (Suppl) pp 173-187, 1982

[Sombra] Martin Sombra, Estimaciones para el Teorema de Ceros de Hilbert, Tesis, Departamento de Matematica, Facultad de Ciencias Exactas y Naturales, 1998.

[Philippon] P. Philippon, Sur des hauteurs alternatives, I, Math Ann. 289 (1991), pp 255-283; II Ann. Inst. Fourier 44 (1994), pp 1043-1065; III, J. Math. Pures Appl. 74 (1995), pp 345-365

[PP] P. Philippon, Quelques remarques sur des questions d'approximation diophantine, Bull Austral. Math. Soc., Vol 59(2), (1999), pp 323-334.

[PP2] P. Philippon, Addendum a quelques remarques sur des questions d'approximation diophantine, Bull Austral. Math. Soc., vol 61(1), (2000), pp 167-169.

[Pohst1] M. Pohst, *Algorithmic Algebraic Number Theory*

[Pohst2] M. Pohst, On Validated Computing in Algebraic Number Fields, J. Symbolic Computation 24, pp 657-666, 1997

[Macintyre] A. Macintyre and A. Wilkie, On the decidability of the real exponential field, in *Kreiseliana, About and Around Georg Kreisel*, A.K. Peters, 1996, pp 441-467.

[Mignotte] M. Mignotte, *Mathematics for Computer Algebra*, Springer Verlag, 1991

[Richardson] D. Richardson, How to Recognise Zero, J. Symbolic Computation (1997), 24, pp 627-645

[Richardson2] D. Richardson, Multiplicative Independence of Algebraic Numbers and Expressions, Mega2000 conference Bath June 2000, to appear in Journal of Pure and Applied Algebra.

[Sprindzhuk] V.G. Sprindzhuk, Achievements and Problems in Diophantine Approximation Theory, Russian Mathematical Surveys 35 (4), 1980, pp 1-80

[Strsebonski] A. W. Strsebonski, Computing in the Field of Complex Algebraic Numbers, J. Symbolic Computation 24, pp 647-656, 1997

[van der Hoeven 95] Joris van der Hoeven, Automatic Numerical Expansions, in J.C. Bajard, D. Michelucci, J.M. Moreau, and J.M. Muller, editors, Proc. of the conference "Real numbers and computers", Saint-Etienne, France, Pages 261-274, 1995

[van der Hoeven 97] Joris van der Hoeven, Automatic Asymptotics, Ph.D. thesis, Ecole Polytechnique, 1997.

[van der Hoeven 2000] Joris van der Hoeven, Zero-testing, witness conjectures and differential diophantine approximation, Preprint

[Zippel] R. Zippel, *Effective Polynomial Computation*, Kluwer Academic Publishers, 1993

Admissible Representations of Limit Spaces

Matthias Schröder

FernUniversität Hagen, D-58084 Hagen, Germany
Matthias.Schroeder@fernuni-hagen.de

Abstract. We give a definition of admissible representations for (weak) limit spaces which allows to handle also non topological spaces in the framework of TTE (Type–2 Theory of Effectivity). Limit spaces and weak limit spaces spaces are generalizations of topological spaces.
We prove that admissible representations $\delta_{\mathfrak{X}}, \delta_{\mathfrak{Y}}$ of weak limit spaces $\mathfrak{X}, \mathfrak{Y}$ have the desirable property that every partial function f between them is continuously realizable with respect to $\delta_{\mathfrak{X}}, \delta_{\mathfrak{Y}}$, iff f is sequentially continuous. Furthermore, we characterize the class of the spaces having an admissible representation. The category of these spaces (equipped with the total sequential continuous functions as morphisms) turns out to be bicartesian–closed. It contains all countably–based T_0–spaces.
Thus, a reasonable computability theory is possible on important non countably–based topological spaces as well as on non topological spaces.

1 Introduction

Type–2 Theory of Effectivity (cf. [Wei00,KW85,Wei87,Wei95,Wei96]) supplies a computational framework for uncountable sets X which are equipped with a natural notion of approximation (e.g. with a topology). The main idea of TTE is to represent the points of a set X by sequences of symbols of a finite or countable infinite alphabet Σ. On these infinite words the actual computation is executed. The corresponding partial surjection[1] $\delta :\subseteq \Sigma^\omega \to X$ mapping every name $p \in dom(\delta)$ to the encoded element $\delta(p)$ is called a *representation* of X.

Computability for functions $f :\subseteq X_1 \times \ldots \times X_k \to X_{k+1}$ is introduced by defining computability for functions on Σ^ω and by relativizing this computability notion by representations. The former is done via computable monotone word–functions $g : (\Sigma^*)^k \to \Sigma^*$ (see Section 2), or, equivalently, via *Type–2 machines*. Let $\delta_i :\subseteq \Sigma^\omega \to X_i$ be a representation of the set X_i for every $i \in \{1, \ldots, k\}$. Then f is said to be *computable* with respect to these representations, iff there is a computable function $\Gamma :\subseteq (\Sigma^\omega)^k \to \Sigma^\omega$ which maps every name of every argument to a name of the corresponding result, i.e.

$$\delta_{k+1}\big(\Gamma(p_1, \ldots, p_k)\big) = f\big(\delta_1(p_1), \ldots, \delta_k(p_k)\big) \tag{1.1}$$

holds for every $(p_1, \ldots, p_k) \in dom\big(f \circ (\delta_1 \times \ldots \times \delta_k)\big)$. By using the binary signed digit representation of $\mathbb{R}$, a computational model on the real numbers is

[1] $\Sigma^\omega := \{p | p : \mathbb{N} \to \Sigma\}$ denotes the set of sequences over Σ. $f :\subseteq A \to B$ indicates that f is a partial function whose domain $dom(f)$ is not necessarily equal to A.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 273–295, 2001.

obtained which is essentially equivalent to the ones considered by other authors like A. Grzegorczyk in [Grz57], Ker-I Ko in [Ko91], M. Pour–El and J. Richards in [PER89] or V. Stoltenberg–Hansen and J.V. Tucker in [SHT99].

The natural question arises which representations of a space lead to a reasonable computability theory on that space. Here the notion of *continuous realizability* plays an important role. It reflects the fact that every computable function $F :\subseteq (\Sigma^\omega)^k \to \Sigma^\omega$ is continuous w.r.t. the Cantor topology on Σ^ω. A function $f :\subseteq X_1 \times \ldots \times X_k \to X_{k+1}$ is said to be *continuously realizable* w.r.t. $\delta_1, \ldots, \delta_{k+1}$, iff there is a *continuous* function $F :\subseteq (\Sigma^\omega)^k \to \Sigma^\omega$ which *realizes* f, i.e. satisfies Equation (1.1). Since computable realizability implies continuous realizability, continuous realizability can be considered as a "topological" generalization of computability. Many representations turn out to be unsuitable, because they lead to an unsatisfactory class of continuously realizable functions. An example is the ordinary decimal representation of $\mathbb{R}$ which does not admit continuous realizability of real addition. In this paper, we will only investigate continuous realizability.

Now we assume that the sets X_i are equipped with a natural notion of approximation. By this we mean a *limit relation* $\to_{X_i} \subseteq X_i^{\mathbb{N}} \times X_i$ assigning to some sequences $(y_n)_n$ some points x considered to be the *limits* of $(y_n)_n$. Limit relations induce a natural notion of continuity: a function $f :\subseteq X \to Y$ is said to be *continuous* with respect to limit relations $\to_X$ and $\to_Y$, iff f preserves convergent sequences (cf. Section 2.1, [Bir36,Dud64]). It seems to be reasonable to demand of the used representations $\delta_1, \ldots, \delta_{k+1}$ to guarantee continuous realizability of all continuous functions $f :\subseteq X_1 \times \ldots \times X_k \to X_{k+1}$. This property is satisfied by *admissible* representations (see Section 3). Kreitz and Weihrauch showed that all topological countably–based T_0–spaces have representations with this property (cf. [Wei00,KW85]). We prove that a strictly larger class of spaces have *admissible* representations.

In Section 2 we introduce the *weak limit spaces* and define continuity of functions between them. Weak limit spaces are reasonable generalizations of limit spaces (cf. Section 2.3) and thus of topological spaces. In Section 3 we define the admissible representations of weak limit spaces and prove the Generalized Main Theorem stating that continuity is equivalent to continuous realizability with respect to admissible representations. Section 4 is devoted to the characterization of the category AWL of those weak limit spaces that are equipped with an admissible representation. In Section 5 we investigate closure properties of AWL and prove that AWL is bicartesian–closed. Finally, we compare in Section 6 the category AWL with the category $\mathsf{PQ_L}$ defined by M. Menni and A. Simpson in [MS00] and prove that both are closely related.

2 Preliminaries

In the following, let Σ be a finite or countably infinite alphabet with at least two elements. For $a \in \Sigma$, $u, v \in \Sigma^*$, $W \subseteq \Sigma^*$, $p, p' \in \Sigma^\omega$ and $n \in \mathbb{N}$ we denote

by

a^n	the word consisting of n symbols a;
a^ω	the sequence q with $q(i) = a$ for every $i \in \mathbb{N}$;
$p^{<n}$	the prefix of length n of p, i.e. the word $p(0)p(1)\ldots p(n-1)$;
$\sqsubseteq$	the prefix–relation on $\Sigma^* \cup \Sigma^\omega$, i.e. $u \sqsubseteq v :\Longleftrightarrow (\exists w \in \Sigma^*)\, uw = v$, $u \sqsubseteq p :\Longleftrightarrow (\exists j \in \mathbb{N})\, u = p^{<j}$, $p \not\sqsubseteq u$ and $p \sqsubseteq p' :\Longleftrightarrow p = p'$;
up	the sequence with prefix u followed by p;
$u\Sigma^\omega$	the set $\{q \in \Sigma^\omega \mid u \sqsubseteq q\}$;
$W\Sigma^\omega$	the set $\{q \in \Sigma^\omega \mid (\exists w \in W) w \sqsubseteq q\}$;
$(\exists^\infty n)$	for infinitely many numbers n;
$(\forall^\infty n)$	for almost all numbers n;
$\overline{\mathbb{N}}$	the set $\mathbb{N} \cup \{\infty\}$.

A function $g : (\Sigma^*)^k \to \Sigma^*$ is called *monotone*, iff $(u_1 \sqsubseteq v_1) \wedge \ldots \wedge (u_k \sqsubseteq v_k)$ implies $g(u_1, \ldots, u_k) \sqsubseteq g(v_1, \ldots, v_k)$ for all words $u_1, v_1, \ldots, u_k, v_k \in \Sigma^*$.
If $g : (\Sigma^*)^k \to \Sigma^*$ is monotone, then we denote by g^ω the partial function $g^\omega :\subseteq (\Sigma^\omega)^k \to \Sigma^\omega$ defined by

$$dom(g^\omega) := \left\{(p_1, \ldots, p_k) \in (\Sigma^\omega)^k \;\middle|\; \{g(p_1^{<n}, \ldots, p_k^{<n}) \mid n \in \mathbb{N}\} \text{ is infinite}\right\}$$

and

$$g^\omega(p_1, \ldots, p_k) := \text{ that sequence } q \in \Sigma^\omega \text{ with } (\forall n \in \mathbb{N})\, g(p_1^{<n}, \ldots, p_k^{<n}) \sqsubseteq q$$

for all $(p_1, \ldots, p_k) \in dom(g^\omega)$.
We call a function $\Gamma :\subseteq (\Sigma^\omega)^k \to \Sigma^\omega$ *computable*, iff there is a computable monotone function $g : (\Sigma^*)^k \to \Sigma^*$ with $g^\omega = \Gamma$, where computability in the case $\Sigma = \mathbb{Z}$ is defined via a canonical effective numbering $\nu_{\mathbb{Z}^*} : \mathbb{N} \to \mathbb{Z}^*$ of $\mathbb{Z}^*$. This notion of computability is equivalent to the one in [Wei00].

2.1 Limit Relations and Continuous Functions

Let X be a set. We denote a sequence $y : \mathbb{N} \to X$ by $(y_n)_n$ or $(y_n)_{n\in\mathbb{N}}$ and a generalized sequence $y : \overline{\mathbb{N}} \to X$ by $(y_n)_{n\in\overline{\mathbb{N}}}$, where $\overline{\mathbb{N}} := \mathbb{N} \cup \{\infty\}$. We call any relation $\to_X \subseteq X^{\mathbb{N}} \times X$ a *limit relation* (or *convergence relation*) on X and say that a sequence $(y_n)_n \in X^{\mathbb{N}}$ *converges to* $x \in X$ with respect to $\to_X$, iff $(y_n)_n \to_X x$ holds. Furthermore, we call a function $x : \overline{\mathbb{N}} \to X$ a *convergent sequence* of the *space* $(X, \to_X)$, iff $(x_n)_n$ *converges to* x_∞ with respect to $\to_X$.
For every $i \in \{1, \ldots, k\}$ let Y_i be a set and $\to_{Y_i}$ be a limit relation on Y_i. As in [Dud64], we define a function $f :\subseteq Y_1 \times \ldots \times Y_k \to X$ to be *continuous with respect to* $\to_{Y_1}, \ldots, \to_{Y_k}$ *and* $\to_X$, iff f preserves convergent sequences, i.e. $\big(f(\bar{y}_n)\big)_n$ converges to $f(\bar{y}_\infty)$ with respect to $\to_X$ whenever $\{\bar{y}_m \mid m \in \overline{\mathbb{N}}\} \subseteq dom(f)$ holds and for every i the i–th projection sequence $\big(pr_i(\bar{y}_n)\big)_n$ converges to $pr_i(\bar{y}_\infty)$ with respect to $\to_{Y_i}$. Sometimes this notion of continuity is called *sequential continuity*. Obviously, the composition of functions preserves continuity.

2.2 Convergence and Continuity in Topological Spaces

Let $\mathfrak{X} = (X, \tau_{\mathfrak{X}})$ and $\mathfrak{Y} = (Y, \tau_{\mathfrak{Y}})$ be topological spaces (cf. [Eng89,Fra65]). The topology $\tau_{\mathfrak{X}}$ induces a limit relation on X denoted by $\to_{\mathfrak{X}}$ or $\to_{\tau_{\mathfrak{X}}}$. It is defined by saying that a sequence $(y_n)_n$ converge to a point $x \in X$, iff $(y_n)_n$ is *eventually in* O for all open sets $O \in \tau_{\mathfrak{X}}$ containing x. (We say that $(y_n)_n$ is *eventually in* a subset M, iff there is some $n_0 \in \mathbb{N}$ such that $y_n \in M$ for all $n \geq n_0$.)
A partial function $f :\subseteq X \to Y$ is called *topologically continuous* with respect to $\tau_{\mathfrak{X}}$ and $\tau_{\mathfrak{Y}}$, iff for every open set $V \in \tau_{\mathfrak{Y}}$ there is an open set $U \in \tau_{\mathfrak{X}}$ with $f^{-1}(V) = U \cap dom(f)$. Topological continuity implies sequential continuity (i.e. continuity w.r.t. the limit relations induced by the topologies), whereas the converse is only true in special cases (cf. [Eng89,Fra65]), for example if $\mathfrak{X}$ is a countably–based space, i.e. a topological space with a countable base. In the following, continuity will always mean sequential continuity and not topological continuity.

On the set Σ^ω we use the Cantor topology $\tau_{\Sigma^\omega} := \{W\Sigma^\omega \mid W \subseteq \Sigma^*\}$ and on the $\overline{\mathbb{N}}$ the topology $\tau_{\overline{\mathbb{N}}} := \left\{O \subseteq \overline{\mathbb{N}} \,\middle|\, \infty \in O \Longrightarrow (\exists n_0 \in \mathbb{N})(\forall n \geq n_0)\, n \in O\right\}$. Both topologies have countable bases, hence topological continuity and sequential continuity coincide for functions whose domain is a subset of Σ^ω or $\overline{\mathbb{N}}$. We denote the limit relations of these topologies by $\to_{\Sigma^\omega}$ and $\to_{\overline{\mathbb{N}}}$, respectively, and will always assume them to be the used limit relations on Σ^ω and $\overline{\mathbb{N}}$.

2.3 Limit Spaces and Weak Limit Spaces

Let $\to_{\mathfrak{X}}$ be a limit relation on X. Then the pair $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ is called a *limit space*, iff the following three axioms are satisfied (cf. [MS00]) (where $x \in X$ and $(y_n)_n \in X^{\mathbb{N}}$):

(L1) $(x)_n \to_{\mathfrak{X}} x$;
(L2) $(y_n)_n \to_{\mathfrak{X}} x$ implies $(y_{\varphi(n)})_n \to_{\mathfrak{X}} x$ for every strictly increasing function $\varphi : \mathbb{N} \to \mathbb{N}$;
(L3) if $(y_n)_n \not\to_{\mathfrak{X}} x$, then there is a strictly increasing function $\varphi : \mathbb{N} \to \mathbb{N}$ such that $(y_{\varphi\psi(n)})_n \not\to_{\mathfrak{X}} x$ for all strictly increasing function $\psi : \mathbb{N} \to \mathbb{N}$.

Limit spaces whose converging sequences have only one limit are called $\mathcal{L}^*$*–spaces* (cf. [Ury26,Kis60,Dud64,Kur66,Eng89]). It is well–known and easy to prove that for every topological space $\mathcal{Z} = (Z, \tau_{\mathcal{Z}})$ the pair $(Z, \to_{\tau_{\mathcal{Z}}})$ is a limit space. Therefore we treat any topological space $(Z, \tau_{\mathcal{Z}})$ as a limit space.

Now we introduce the weak limit spaces. We define the pair $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ to be a *weak limit space*, iff it satisfies Axioms (L1), (L4) and (L5):

(L4) if $(x_n)_{n\in\overline{\mathbb{N}}}$ is a convergent sequence of $\mathfrak{X}$, then the function $x : \overline{\mathbb{N}} \to X$ is continuous with respect to $\to_{\overline{\mathbb{N}}}$ and $\to_{\mathfrak{X}}$;
(L5) $(y_{n+1})_n \to_{\mathfrak{X}} x$ implies $(y_n)_n \to_{\mathfrak{X}} x$.

Obviously, weak limit spaces satisfy Axiom (L2). The weak limit spaces whose converging sequences have only one limit turn out to be exactly the $\mathcal{L}^+$–spaces defined in [Dud64]. Weak limit spaces are motivated by their characterization in Lemma 10 in conjunction with Theorem 13. Axioms (L4) and (L5) are equivalent to the two technical properties stated in Lemma 1.

Lemma 1.
Let $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ be a weak limit space, and let $\mathtt{a} \in \Sigma$.

(1) For every convergent sequence $(y_n)_{n\in\overline{\mathbb{N}}}$ of $\mathfrak{X}$ the function $\phi : \Sigma^\omega \to X$ defined by $\phi(p) := y_{\min(\{\infty\} \cup \{n \in \mathbb{N} \,|\, p(n) \neq \mathtt{a}\})}$ is continuous.

(2) Let $\delta :\subseteq \Sigma^\omega \to X$ be continuous, and let $p \in dom(\delta)$.
Then for every sequence $(z_n)_n$ that does not converge to $\delta(p)$ there is some $k \in \mathbb{N}$ such that $z_n \notin \delta(p^{<k}\Sigma^\omega)$ holds for infinitely many n.

Proof:

(1) The function $k : \Sigma^\omega \to \overline{\mathbb{N}}$ defined by $k(p) := \min(\{\infty\} \cup \{n \in \mathbb{N} \,|\, p(n) \neq \mathtt{a}\})$ is continuous with respect to $\to_{\Sigma^\omega}$ and $\to_{\overline{\mathbb{N}}}$. By Axiom (L4) this implies that $\phi = y \circ k$ is continuous with respect to $\to_{\Sigma^\omega}$ and $\to_{\mathfrak{X}}$.

(2) Assume $(\forall k \in \mathbb{N})(\exists l \in \mathbb{N})(\forall n \geq l)\, z_n \in \delta(p^{<k}\Sigma^\omega)$. Then there is a strictly increasing sequence $l_0, l_1, l_2 \ldots$ such that $\{z_n \,|\, n \geq l_k\} \subseteq \delta(p^{<k}\Sigma^\omega)$ for all $k \in \mathbb{N}$. For every $k \in \mathbb{N}$ and every $n \in \{l_k, \ldots, l_{k+1} - 1\}$ there is some $q_n \in p^{<k}\Sigma^\omega$ with $\delta(q_n) = z_n$. Since $\lim_{n\to\infty} q_{n+l_0} = p$ and δ is continuous, the sequence $(z_{n+l_0})_n$ converges to $\delta(p)$. Axiom (L5) implies $(z_n)_n \to_{\mathfrak{X}} \delta(p)$, a contradiction.

□

Every limit space and thus every topological space is a weak limit space, whereas not every weak limit space satisfies Axiom (L3). We only prove:

Lemma 2. *Every limit space $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ is a weak limit space.*

Proof:

Axiom (L4): Let $(x_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence of $\mathfrak{X}$ and let $(k_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence of $(\overline{\mathbb{N}}, \tau_{\overline{\mathbb{N}}})$. We show that every subsequence of $(x_{k_n})_n$ has a subsequence converging to x_{k_∞}. Let $\varphi : \mathbb{N} \to \mathbb{N}$ be strictly increasing.
If $k_{\varphi(n)} = k_\infty$ for infinitely many $n \in \mathbb{N}$, then there is a strictly increasing function $\psi : \mathbb{N} \to \mathbb{N}$ with $range(\psi) = \{n \in \mathbb{N} \,|\, k_{\varphi(n)} = k_\infty\}$. By Axiom (L1), $(x_{k_{\varphi\psi(n)}})_n$ converges to x_{k_∞}.
Otherwise we have $k_{\varphi(n)} \neq k_\infty$ for almost all $n \in \mathbb{N}$ and therefore $k_\infty = \infty$. Since $(k_{\varphi(n)})_n$ converges to ∞, for all $m \in \mathbb{N}$ there is some $n_m \in \mathbb{N}$ such that $m \leq k_{\varphi(n)} < \infty$ holds for all $n \geq n_m$. Hence we can define $\psi : \mathbb{N} \cup \{-1\} \to \mathbb{N} \cup \{-1\}$ recursively by $\psi(-1) := -1$ and

$$\psi(i) := \min\left\{n > \psi(i-1) \,\middle|\, k_{\varphi\psi(i-1)} < k_{\varphi(n)} < \infty\right\}.$$

Then ψ and $i \mapsto k_{\varphi\psi(i)}$ are strictly increasing. Thus $(x_{k_{\varphi\psi(i)}})_i$ converges to x_∞ by Axiom (L2).
From Axiom (L3) we conclude that $(x_{k_i})_i$ converges to x_{k_∞}. Thus the function $k \mapsto x_k$ is continuous with respect to $\to_{\overline{\mathbb{N}}}$ and $\to_{\mathfrak{X}}$.

Axiom (L5): Let $(y_n)_n$ be a sequence in X such that $(y_{n+1})_n$ converges to some $x \in X$. Define $\psi : \mathbb{N} \to \mathbb{N}$ by $\psi(n) := n + 1$. For every strictly increasing function $\varphi : \mathbb{N} \to \mathbb{N}$ the sequence $(y_{\varphi\psi(n)})_n$ is a subsequence of $(y_{n+1})_n$. Thus $(y_{\varphi\psi(n)})_n$ converges to x by Axiom (L2). From Axiom (L3) we deduce $(y_n)_n \to_{\mathfrak{X}} x$.

□

We mention a nice characterization of the weak limit spaces showing that they form a very natural class. Let $\mathcal{Z} = (Z, \tau_{\mathcal{Z}})$ be a topological space and let $f : Z \to X$ be a surjection onto the set X. Consider the smallest (finest) limit relation $\to_X$ on X such that f is sequentially continuous with respect to $\to_{\mathcal{Z}}$ and $\to_X$. Then the pair $(X, \to_X)$ is a weak limit space (cp. Proposition 14(1)). Conversely, every weak limit space is equal to such a "quotient" of an appropriate metric space (cp. Proposition 9). This characterization is a generalization of Franklin's theorem stating that the sequential topological spaces are exactly the topological quotients of metric spaces (cf. [Fra65]).

2.4 T_0-Property

We say that a weak limit space $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ satisfies the T_0–property, iff for all $x, y \in X$ with $x \neq y$ we have $(x)_n \not\to_{\mathfrak{X}} y$ or $(y)_n \not\to_{\mathfrak{X}} x$. This definition is consistent with the usual definition of a topological T_0–space.

2.5 Sequentially Open and Sequentially Closed Sets

For spaces satisfying the Axioms (L1) and (L2) there is a natural notion of "open" and "closed" sets (cf. [Bir36,Dud64], see also [Fra65,Eng89,MS00]).
Let $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ and $\mathfrak{Y} = (Y, \to_{\mathfrak{Y}})$ be weak limit spaces, hence (L1) and (L2) are satisfied. We call a subset O of X *sequentially open*, iff every sequence $(y_n)_n$ that converges to an element of O is eventually in O. The complements of sequentially open sets are called *sequentially closed*. Sequentially closed sets are characterized by the property that they contain all limits of their converging sequences. The family $seq(\to_{\mathfrak{X}})$ of all sequentially open sets of $\mathfrak{X}$ turns out to be a topology on X. We called it the *associated topology* of $\mathfrak{X}$. The limit relation $\to_{seq(\to_{\mathfrak{X}})}$ induced by this topology contains the original limit relation $\to_{\mathfrak{X}}$, i.e. $(y_n)_n \to_{\mathfrak{X}} x$ implies $(y_n)_n \to_{seq(\to_{\mathfrak{X}})} x$. Every total function $f : X \to Y$ which is continuous with respect to $\to_{\mathfrak{X}}$ and $\to_{\mathfrak{Y}}$ is also topologically continuous with respect to the associated topologies $seq(\to_{\mathfrak{X}})$ and $seq(\to_{\mathfrak{Y}})$. If there is any topology on X inducing the limit relation $\to_{\mathfrak{X}}$ of $\mathfrak{X}$, then also $seq(\to_{\mathfrak{X}})$ induces $\to_{\mathfrak{X}}$. The topology of a countably–based topological space $\mathcal{Z}$ is equal to the topology $seq(\to_{\mathcal{Z}})$ of the sequentially open sets of $\mathcal{Z}$ (cf. [Eng89,Fra65]).

3 Admissible Representations

In this section we define the admissible representations and compare this notion of admissibility with the classical one in [Wei00,KW85,Wei87,Wei95,Wei96]. We prove that continuous realizability of a function f with respect to admissible representations is equivalent to sequential continuity of f. Furthermore, we investigate the final topology of an admissible representation.

3.1 The Definition of Admissible Representations

Let $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ be a weak limit space, and let $\delta_{\mathfrak{X}} :\subseteq \Sigma^\omega \to X$ be a representation of $\mathfrak{X}$. For every $p \in dom(\delta_{\mathfrak{X}})$ and every $n \in \mathbb{N}$, each element $y_n \in \delta_{\mathfrak{X}}(p^{<n}\Sigma^\omega)$ can be considered as an "approximation" of $\delta_{\mathfrak{X}}(p)$. If $\delta_{\mathfrak{X}}$ is continuous with respect to $\to_{\mathfrak{X}}$, then this is true with respect to the notion of approximation given by $\to_{\mathfrak{X}}$, because in this case such a sequence $(y_n)_n$ converges to $\delta_{\mathfrak{X}}(p)$ in the space $\mathfrak{X}$. Thus, it is quite natural to demand continuity of $\delta_{\mathfrak{X}}$.

On the other hand, our aim is to have a representation $\delta_{\mathfrak{X}}$ such that at least every continuous function $\phi :\subseteq \Sigma^\omega \to X$ is continuously realizable w.r.t. the identical representation id_{Σ^ω} of Σ^ω and $\delta_{\mathfrak{X}}$. This requirement implies that ϕ is *continuously reducible* to $\delta_{\mathfrak{X}}$ (denoted by $\phi \leq_{\mathrm{t}} \delta_{\mathfrak{X}}$) which means the existence of a continuous function $G :\subseteq \Sigma^\omega \to \Sigma^\omega$ such that $\phi(p) = \delta_{\mathfrak{X}}\big(G(p)\big)$ holds for all $p \in dom(\phi)$.

These considerations justify the following definition.

Definition 3 (Admissible Representations).
Let $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ be a weak limit space, and let $\delta :\subseteq \Sigma^\omega \to X$ be a representation of X.

(1) δ is said to be an *admissible representation of* the space $\mathfrak{X}$, iff δ is continuous and every continuous function $\phi :\subseteq \Sigma^\omega \to X$ is continuously reducible to δ.
(2) Let $\to$ be any limit relation on X such that $(X, \to)$ is a weak limit space. Then δ is said to be $\to$*–admissible* or *admissible with respect to* $\to$, iff δ is an admissible representation of the weak limit space $(X, \to)$.

The property of an admissible representation δ that every continuous function $\phi :\subseteq \Sigma^\omega \to X$ is continuously reducible to δ is called the *universal property* of δ.

Two representations $\delta_1, \delta_2 :\subseteq \Sigma^\omega \to X$ of X are called *continuously equivalent* ($\delta_1 \equiv_{\mathrm{t}} \delta_2$ for short), iff they are continuously reducible to each other, i.e. iff $\delta_1 \leq_{\mathrm{t}} \delta_2 \leq_{\mathrm{t}} \delta_1$ holds. One can easily prove that if $\delta :\subseteq \Sigma^\omega \to X$ is an admissible representation of $\mathfrak{X}$, then for any further representation $\gamma :\subseteq \Sigma^\omega \to X$ of X we have

$$\gamma \text{ is admissible with respect to } \to_{\mathfrak{X}}, \text{ iff } \gamma \equiv_{\mathrm{t}} \delta. \tag{3.1}$$

3.2 Comparison with the Classical Definition of Admissibility

In [Wei00,Wei95,Wei96], a representation $\gamma :\subseteq \Sigma^\omega \to X$ of a countably–based T_0–space $\mathfrak{X} = (X, \tau)$ is defined to be admissible, iff γ is continuously equivalent to a *standard representation* $\delta_{(X,\sigma,\nu)}$ of $\mathfrak{X}$ which is derived from a notation $\nu :\subseteq \Sigma^* \to \sigma$ of a subbase σ of the space $\mathfrak{X}$. As an important property, it is shown that the standard representation $\delta_{(X,\sigma,\nu)}$ is continuous with respect to $\to_{\Sigma^\omega}$ and $\to_\tau$ and satisfies the universal property. Thus $\delta_{(X,\sigma,\nu)}$ is admissible in the sense of Definition 3. From Equivalence (3.1) it follows that every representation of the countably–based T_0–space $\mathfrak{X}$ is admissible in the classical sense, if and only if it is an admissible one of $(X, \to_\tau)$ in the sense of Definition 3.

The following example shows a limit space which has an admissible representation, but is not topological, i.e. no topology induces its limit relation.

Example 4.
Let $X := \{1, 2, 3\}$. We define the limit relation $\to_{\mathfrak{X}}$ on X by

$$\begin{aligned}
(y_n)_n \to_{\mathfrak{X}} 1 &:\iff \{n \in \mathbb{N} \mid y_n = 2\} \text{ is finite} \\
(y_n)_n \to_{\mathfrak{X}} 2 &:\iff \{n \in \mathbb{N} \mid y_n = 3\} \text{ is finite} \\
(y_n)_n \to_{\mathfrak{X}} 3 &:\iff \{n \in \mathbb{N} \mid y_n = 1\} \text{ is finite}
\end{aligned}$$

for all sequences $(y_n)_n \in X^{\mathbb{N}}$. One can easily prove that $\mathfrak{X} := (X, \to_{\mathfrak{X}})$ satisfies the Axioms (L1), (L2) and (L3), i.e. $\mathfrak{X}$ is a limit space. The only sequentially closed sets of $\mathfrak{X}$ are $\emptyset$ and X. Thus the topology $seq(\to_{\mathfrak{X}})$ is the indiscrete topology on X. Since in an indiscrete topological space every sequence converges to every point, the topology $seq(\to_{\mathfrak{X}})$ does not induce the limit relation $\to_{\mathfrak{X}}$. This implies that there is no topology on X inducing $\to_{\mathfrak{X}}$ (cf. Section 2.5).

An admissible representation $\delta :\subseteq \Sigma^\omega \to X$ over the alphabet $\Sigma := \{\mathtt{1}, \mathtt{2}, \mathtt{3}\}$ can be defined by

$$\delta(p) := \begin{cases} 1 & \text{if } (\exists m \in \mathbb{N})\, p(m) = \mathtt{1} \text{ and } (\forall m \in \mathbb{N})\, p(m) \neq \mathtt{3} \\ 2 & \text{if } (\exists m \in \mathbb{N})\, p(m) = \mathtt{2} \text{ and } (\forall m \in \mathbb{N})\, p(m) \neq \mathtt{1} \\ 3 & \text{if } (\exists m \in \mathbb{N})\, p(m) = \mathtt{3} \text{ and } (\forall m \in \mathbb{N})\, p(m) \neq \mathtt{2} \\ \uparrow & \text{else} \end{cases}$$

for all $p \in \Sigma^\omega$.
Let $(p_n)_{n \in \overline{\mathbb{N}}}$ be a convergent sequence in $dom(\delta)$. W.l.o.g. we can assume $\delta(p_\infty) = 1$. Thus there are $n_0, m \in \mathbb{N}$ with $p_\infty(m) = \mathtt{1}$ and $p_n(m) = \mathtt{1}$ for all $n \geq n_0$. This implies $\{n \in \mathbb{N} \mid \delta(p_n) = 2\}$ is finite, i.e. $\big(\delta(p_n)\big)_n$ converges to $\delta(p_\infty)$. Hence δ is continuous.

Let $\phi :\subseteq \Sigma^\omega \to X$ be continuous. Define $\beta_1 := \{1, 3\}$, $\beta_2 := \{2, 1\}$ and $\beta_3 := \{3, 2\}$. Let $p \in dom(\phi)$. By continuity of ϕ, there is some $m \in \mathbb{N}$ such that $\phi(p^{<m}\Sigma^\omega) \subseteq \beta_{\phi(p)}$. Therefore it is possible to define $i_p \in \mathbb{N}$, $j_p \in \{1, 2, 3\}$ and

$k_{p,1}, k_{p,2}, k_{p,3} \in \overline{\mathbb{N}}$ by

$$\begin{aligned} i_p &:= \min\left\{i \in \mathbb{N} \,\middle|\, (\exists j \in \{1,2,3\})\, \phi(p^{<i}\Sigma^\omega) \subseteq \beta_j\right\} \\ j_p &:= \min\left\{j \in \{1,2,3\} \,\middle|\, \phi(p^{<i_p}\Sigma^\omega) \subseteq \beta_j\right\} \\ k_{p,l} &:= \min\left\{\infty\right\} \cup \left\{k \geq i_p \,\middle|\, \phi(p^{<k}\Sigma^\omega) = \{l\}\right\}. \end{aligned}$$

We define $G :\subseteq \Sigma^\omega \to \Sigma^\omega$ by

$$G(p) := \begin{cases} \mathtt{1}^{k_{p,3}}\mathtt{3}^\omega & \text{if } j_p = 1 \text{ and } k_{p,3} < \infty \\ \mathtt{1}^\omega & \text{if } j_p = 1 \text{ and } k_{p,3} = \infty \\ \mathtt{2}^{k_{p,1}}\mathtt{1}^\omega & \text{if } j_p = 2 \text{ and } k_{p,1} < \infty \\ \mathtt{2}^\omega & \text{if } j_p = 2 \text{ and } k_{p,1} = \infty \\ \mathtt{3}^{k_{p,2}}\mathtt{2}^\omega & \text{if } j_p = 3 \text{ and } k_{p,2} < \infty \\ \mathtt{3}^\omega & \text{if } j_p = 3 \text{ and } k_{p,2} = \infty \\ \uparrow & \text{else} \end{cases}$$

Then G is continuous, because the functions $p \mapsto i_p$, $p \mapsto j_p$ and $p \mapsto k_{p,l}$ are continuous. Obviously, we have $\phi(p) = \delta\big(G(p)\big)$ for all $p \in dom(\phi)$. Hence δ has the universal property.

□

3.3 Continuous Realizability versus Sequential Continuity

We will now show that continuous realizability with respect to admissible representations is equivalent to sequential continuity (cf. Section 2). Thus our notion of admissibility has the property promised in Section 1. Since for partial functions between countably–based topological spaces topological continuity is equivalent to sequential continuity, Theorem 6 generalizes the Main Theorem in [Wei00,Wei95,Wei96] to weak limit spaces.
First we prove that every admissible representation is "convergent–sequence–covering".

Lemma 5. *Let $\delta :\subseteq \Sigma^\omega \to X$ be an admissible representation of a weak limit space $\mathfrak{X} = (X, \to_{\mathfrak{X}})$. Then for every convergent sequence $(x_n)_{n\in\overline{\mathbb{N}}}$ there is a convergent sequence $(q_n)_{n\in\overline{\mathbb{N}}}$ in $dom(\delta)$ with $\big(\delta(q_n)\big)_{n\in\overline{\mathbb{N}}} = (x_n)_{n\in\overline{\mathbb{N}}}$.*

Proof: Choose two symbols $\mathtt{a} \neq \mathtt{b}$ from the alphabet Σ. By Lemma 1, the function $\phi : \Sigma^\omega \to X$ defined by

$$\phi(p) := x_{\min(\{\infty\} \cup \{n \in \mathbb{N} \,|\, p(n) \neq \mathtt{a}\})}$$

is continuous. The universality of δ yields a continuous function $G : \Sigma^\omega \to \Sigma^\omega$ with $\phi = \delta \circ G$. Define $q_\infty := G(\mathtt{a}^\omega)$ and $q_n := G(\mathtt{a}^n\mathtt{b}^\omega)$ for every $n \in \mathbb{N}$. Then $(q_n)_n$ converges to q_∞, because G is continuous. Obviously, we have $\delta(q_n) = x_n$ for all $n \in \overline{\mathbb{N}}$.

□

Theorem 6 is an easy consequence of the universal property and of Lemma 5.

Theorem 6 (Generalized Main Theorem).
Let $\mathfrak{X}_i = (X_i, \to_{\mathfrak{X}_i})$ be a weak limit space and $\delta_i :\subseteq \Sigma^\omega \to X_i$ be an admissible representation of $\mathfrak{X}_i$ for $i \in \{1, \ldots, k+1\}$. Then a function $f :\subseteq X_1 \times \ldots \times X_k \to X_{k+1}$ is continuously realizable with respect to $\delta_1, \ldots, \delta_{k+1}$, iff f is continuous.

Proof: For the sake of simplicity, we restrict us to the case $k = 1$.

"$\Longleftarrow$": If f is continuous, then by continuity of δ_1 the function $\phi := f \circ \delta_1$ is continuous as well. By universality of δ_2, there is a continuous function $G :\subseteq \Sigma^\omega \to \Sigma^\omega$ with $\phi = \delta_2 \circ G$. Obviously, G realizes f with respect to δ_1 and δ_2.

"$\Longrightarrow$": Let $(x_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence in $dom(f)$. By Lemma 5, there is convergent sequence $(q_n)_{n\in\overline{\mathbb{N}}}$ with $\big(\delta_1(q_n)\big)_{n\in\overline{\mathbb{N}}} = (x_n)_{n\in\overline{\mathbb{N}}}$. Since f is continuously realizable with respect to δ_1 and δ_2, there is a continuous function $G :\subseteq \Sigma^\omega \to \Sigma^\omega$ with $f \circ \delta_1 = \delta_2 \circ G$. For all $n \in \overline{\mathbb{N}}$ we have

$$f(x_n) = f\big(\delta_1(q_n)\big) = \delta_2\big(G(q_n)\big)\,.$$

By continuity of $\delta_2 \circ G$, the sequence $\big(\delta_2(G(q_n))\big)_n$ converges to $\delta_2(G(q_\infty))$. Thus the sequence $\big(f(x_n)\big)_n$ converges to $f(x_\infty)$.

□

Since relative computability implies continuous realizability, we obtain:

Corollary 7. *Computability with respect to admissible representations implies sequential continuity.*

3.4 The Final Topology of Admissible Representations

Now we prove that the final topology of an admissible representation of a weak limit space $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ is equal to the family $seq(\to_{\mathfrak{X}})$ of all sequentially open subsets of $\mathfrak{X}$ (cf. Section 2.5).

Proposition 8. *Let $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ be a weak limit space, and let $\delta :\subseteq \Sigma^\omega \to X$ be an admissible representation of $\mathfrak{X}$. Then the final topology*

$$\tau_\delta := \big\{O \subseteq X \;\big|\; (\exists U \in \tau_{\Sigma^\omega})\, \delta^{-1}(O) = U \cap dom(\delta)\big\}$$

of δ is equal to the family $seq(\to_{\mathfrak{X}})$ of all sequentially open subsets of $\mathfrak{X}$.

Proof:

"$\tau_\delta \subseteq seq(\to_{\mathfrak{X}})$": Let $O \in \tau_\delta$. Then there is a set $W \subseteq \Sigma^*$ with $\delta^{-1}(O) = W\Sigma^\omega \cap dom(\delta)$. Let $(x_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence of $\mathfrak{X}$ with $x_\infty \in O$. By Lemma 5, there is convergent sequence $(q_n)_{n\in\overline{\mathbb{N}}}$ with $\big(\delta(q_n)\big)_{n\in\overline{\mathbb{N}}} = (x_n)_{n\in\overline{\mathbb{N}}}$. Since $\delta(q_\infty) \in O$, there exists some $i \in \mathbb{N}$ with $q_\infty^{<i} \in W$. There is some $n_0 \in \mathbb{N}$ with $q_n^{<i} = q_\infty^{<i}$ for all $n \geq n_0$, because $\lim\limits_{n\to\infty} q_n = q_\infty$. We obtain

$$x_n = \delta(q_n) \in \delta(q_n^{<i}\Sigma^\omega) = \delta(q_\infty^{<i}\Sigma^\omega) \subseteq \delta(W\Sigma^\omega) = O$$

for all $n \geq n_0$. Thus $(x_n)_n$ is eventually in O. Hence O is sequentially open.

"$seq(\to_{\mathfrak{X}}) \subseteq \tau_\delta$": Let O be sequentially open in $\mathfrak{X}$. Let $p \in \delta^{-1}(O)$.
Assume that for every $n \in \mathbb{N}$ there is some $q_n \in p^{<n}\Sigma^\omega$ with $\delta(q_n) \in X \setminus O$.
Since $(q_n)_n$ converges to p and δ is continuous, $\big(\delta(q_n)\big)_n$ converges to $\delta(p)$.
Therefore the sequence $\big(\delta(q_n)\big)_n$ is eventually in O, a contradiction.
Hence there is some $n_p \in \mathbb{N}$ with $\delta(p^{<n_p}\Sigma^\omega) \subseteq O$. Obviously, the set

$$W := \big\{ q^{<n_q} \,\big|\, q \in \delta^{-1}(O) \big\}$$

satisfies $\delta^{-1}(O) = W\Sigma^\omega \cap dom(\delta)$. We obtain $O \in \tau_\delta$.

□

The following "quotient–property" of admissible representations is a direct consequence of Lemma 5.

Proposition 9. *Let $\delta :\subseteq \Sigma^\omega \to X$ be an admissible representation of a weak limit space $\mathfrak{X} = (X, \to_{\mathfrak{X}})$. Then $\to_{\mathfrak{X}}$ is the smallest limit relation $\to$ on X such that δ is continuous with respect to $\to_{\Sigma^\omega}$ and $\to$.*

4 Characterization of the Weak Limit Spaces with Admissible Representations

In this section, we characterize the class AWL of those weak limit spaces that have admissible representations. For this purpose we define the *limit bases* of a weak limit space. Then we prove that a weak limit space has an admissible representation, if and only if it has a countable limit base and satisfies the T_0–property.

4.1 Limit Bases

Let $\to_{\mathfrak{X}}$ be a limit relation on the set X. Let $\mathcal{B}$ be a family of subsets of X. Then we call $\mathcal{B}$ a *limit base of* the pair $\mathfrak{X} = (X, \to_{\mathfrak{X}})$, iff for all $x \in X$ and all sequences $(y_n)_n$ and $(z_n)_n$ there is an element $B \in \mathcal{B}$ such that

- $x \in B$ (LB1)
- $(\forall^\infty n \in \mathbb{N})\, y_n \in B$ if $(y_n)_n$ converges to x and (LB2)
- $(\exists^\infty n \in \mathbb{N})\, z_n \notin B$ if $(z_n)_n$ does not converge to x. (LB3)

Every superset of a limit base is a limit base of the same space. It is easy to see that every subbase of a topological space (X, τ) is a limit base of the corresponding limit space $(X, \to_\tau)$. The family $\big\{\{1,3\},\{2,1\},\{3,2\}\big\}$ can be shown to be a limit base of the space considered in Example 4.

Weak limit spaces are characterized by the existence of a limit base.

Lemma 10. *Let $\to_{\mathfrak{X}}$ be a limit relation on the set X, i.e. $\to_{\mathfrak{X}} \subseteq X^{\mathbb{N}} \times X$. Then $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ is weak limit space, iff $\mathfrak{X}$ has a limit base.*

Proof:

"$\Longrightarrow$": Let $(y_n)_n$ be a sequence converging to a point $y_\infty \in X$, and let $(z_n)_n$ be a sequence that does not converge to y_∞. For every $k \in \mathbb{N}$ define $B_k := \{y_\infty\} \cup \{y_i \mid i \geq k\}$.

Assume that for all $k \in \mathbb{N}$ the set B_k contains z_n for almost all $n \in \mathbb{N}$. Then there is a strictly increasing sequence $l_0, l_1, \ldots \in \mathbb{N}$ with $\{z_n \mid n \geq l_k\} \subseteq B_k$. For every $k \in \mathbb{N}$ and every $n \in \{l_k, \ldots, l_{k+1} - 1\}$ there is some $\xi(n) \geq k$ with $y_{\xi(n)} = z_n$. Since $\big(\xi(n + l_0)\big)_n$ converges to ∞ and the function $\iota \mapsto y_\iota$ is continuous by Axiom (L4), the sequence $(z_{n+l_0})_n = (y_{\xi(n+l_0)})_n$ converges to y_∞. Axiom (L5) implies that $(z_n)_n$ converges to y_∞, a contradiction.

Hence there is some $k \in \mathbb{N}$ such that B_k does not contain z_n for infinitely many $n \in \mathbb{N}$. Thus 2^X is a limit base of $\mathfrak{X}$.

"$\Longleftarrow$": Let $\mathcal{B}$ be a limit base of $\mathfrak{X}$.

Axiom (L1): Assume that the constant sequence $(z_n)_n := (x)_n$ does not converge to x. Then there is a set $B \in \mathcal{B}$ with $x \in B$ such that B does not contain z_n for infinitely many $n \in \mathbb{N}$, a contradiction.

Axiom (L5): Let $(y_n)_n$ be a sequence such that $(y_n)_{n\geq 1}$ converges to some $x \in X$. Assume that $(y_n)_n$ does not converge to x. Then there are $B \in \mathcal{B}$ and $n_0 \in \mathbb{N}$ such that $\{y_{n+1} \mid n \geq n_0\} \subseteq B$ and B does not contain y_n for infinitely many $n \in \mathbb{N}$. Of course this is impossible.

Axiom (L4): Let $(x_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence of $\mathfrak{X}$. Let $(m_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence in $\overline{\mathbb{N}}$.

Case 1: Let $m_\infty = \infty$.

Assume that $(x_{m_n})_n$ does not converge to x_{m_∞}. Then there are $B \in \mathcal{B}$ and $n_1 \in \mathbb{N}$ such that $\{x_\infty, x_n \mid n \geq n_1\} \subseteq B$ and $x_{m_n} \notin B$ for infinitely many n. But since $(m_n)_n$ converges to ∞, there is some $n_2 \in \mathbb{N}$ such that $m_n \geq n_1$ for all $n \geq n_2$ and thus $\{x_{m_n} \mid n \geq n_2\} \subseteq B$, a contradiction.

Case 2: Let $m_\infty \neq \infty$. Then there is some $n_0 \in \mathbb{N}$ with $m_n = m_\infty$ for all $n \geq n_0$. By Axiom (L1), $(x_{m_n})_{n\geq n_0}$ converges to x_{m_∞}. This implies $(x_{m_n})_n \to_{\mathfrak{X}} x_{m_\infty}$ by Axiom (L5).

□

4.2 Countable Limit Bases versus Admissible Representations

Now we can characterize the weak limit spaces which have an admissible representation.

First we show that the existence of an admissible representation implies the existence of a countable limit base.

Proposition 11. *Let $\delta :\subseteq \Sigma^\omega \to X$ be an admissible representation of a weak limit space $\mathfrak{X} = (X, \to_{\mathfrak{X}})$. Then the family $\big\{\delta(w\Sigma^\omega) \,\big|\, w \in \Sigma^*\big\}$ is a countable limit base of $\mathfrak{X}$.*

Proof: Let $(x_n)_n$ be a sequence converging to an element $x_\infty \in X$, and let $(z_n)_n$ be a sequence that does not converge to x_∞. By Lemma 5, there is convergent sequence $(q_n)_{n\in\overline{\mathbb{N}}}$ with $\big(\delta(q_n)\big)_{n\in\overline{\mathbb{N}}} = (x_n)_{n\in\overline{\mathbb{N}}}$. By Lemma 1, there is some $i \in \mathbb{N}$ such that $\delta(q_\infty^{<i}\Sigma^\omega)$ does not contain z_n for infinitely many $n \in \mathbb{N}$. There is some n_0 with $q_n \in q_\infty^{<i}\Sigma^\omega$ for all $n \geq n_0$. Obviously, the word $w := q_\infty^{<i}$ satisfies

$$\{x_\infty\} \cup \{x_n \,|\, n \geq n_0\} \subseteq \delta(w\Sigma^\omega) \quad \text{and} \quad (\exists^\infty n \in \mathbb{N})\, z_n \notin \delta(w\Sigma^\omega)\,.$$

□

Now we construct an admissible representation $\delta_{\mathfrak{X},\beta}$ of a weak limit space $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ with the T_0–property starting from a numbering[2] $\beta : \mathbb{N} \to \mathcal{B}$ of a countable limit base $\mathcal{B}$ of $\mathfrak{X}$. We define $\delta_{\mathfrak{X},\beta} :\subseteq \mathbb{Z}^\omega \to X$ by

$$\delta_{\mathfrak{X},\beta}(p) = x \;:\Longleftrightarrow\; \begin{cases} \mathit{range}(p) \cap \mathbb{N} \subseteq \{i \in \mathbb{N} \,|\, x \in \beta_i\} \text{ and} \\ \big(\forall (z_n)_n \in X^{\mathbb{N}}\big)\big((z_n)_n \not\to_{\mathfrak{X}} x \implies \\ \qquad (\exists i \in \mathit{range}(p))(\exists^\infty n)\, z_n \notin \beta_i\big) \end{cases} \tag{4.1}$$

for all $p \in \mathbb{Z}^\omega$ and $x \in X$, where $\mathit{range}(p) = \{p(j) \,|\, j \in \mathbb{N}\}$. Thus, a $\delta_{\mathfrak{X},\beta}$–name p of a point x lists "sufficiently many" limit base elements containing x. Note that in general $\delta_{\mathfrak{X},\beta}$ would not become an admissible representation of $\mathfrak{X}$, if we demanded of all names of x to list *all* limit base elements containing x, because limit base elements are not necessarily sequentially open.

Theorem 12 (Construction of an Admissible Representation).
Let $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ be a weak limit space satisfying the T_0–property, and let $\beta : \mathbb{N} \to \mathcal{B}$ be a numbering of a limit base $\mathcal{B}$ of $\mathfrak{X}$.
Then the function $\delta_{\mathfrak{X},\beta}$ from (4.1) *is well–defined and is an admissible representation of the space $\mathfrak{X}$.*

Proof:

Unambiguity: Let $x \neq y$ be points of X and let $p \in \mathbb{Z}^\omega$. By the T_0–property, the constant sequence $(y)_n$ does not converge to x or $(x)_n$ does not converges to y, w.l.o.g. we assume $(y)_n \not\to_{\mathfrak{X}} x$. If (p, x) satisfies the right hand side of (4.1), then there is some $j \in \mathbb{N}$ such that $(y)_n$ is not eventually in $\beta_{p(j)}$. Thus the sequence p cannot be a $\delta_{\mathfrak{X},\beta}$–name for y, because $y \notin \beta_{p(j)}$.

Surjectivity: Let $x \in X$. Then every $p \in \mathbb{Z}^\omega$ with $\mathit{range}(p) = \{n \in \mathbb{N} \,|\, x \in \beta_n\}$ is a $\delta_{\mathfrak{X},\beta}$–name of x, because $\{\beta_0, \beta_1, \dots\}$ is a limit base of $\mathfrak{X}$.

Continuity: Let $(q_n)_n$ be a sequence in $\mathit{dom}(\delta_{\mathfrak{X},\beta})$ converging to some $p \in \mathit{dom}(\delta_{\mathfrak{X},\beta})$. For $n \in \mathbb{N}$ define $z_n := \delta_{\mathfrak{X},\beta}(q_n)$.
Assume that $(z_n)_n$ does not converge to $\delta_{\mathfrak{X},\beta}(p)$. Then there is some $j \in \mathbb{N}$ such that $p(j) \in \mathbb{N}$ and $z_n \notin \beta_{p(j)}$ for infinitely many $n \in \mathbb{N}$. Since $(q_n)_n$ converges to p, there is some n_0 such that $q_n(j) = p(j)$ for all $n \geq n_0$ implying $\{z_n \,|\, n \geq n_0\} \subseteq \beta_{p(j)}$, a contradiction.
Hence, $\delta_{\mathfrak{X},\beta}$ is continuous.

[2] i.e. β is surjective

Universality: Let $\phi :\subseteq \mathbb{Z}^\omega \to X$ be continuous.

Let $p \in dom(\phi)$, $x := \phi(p)$ and $(z_l)_l$ be a sequence that does not converge to x. We show that there is a pair $(i,n) \in \mathbb{N}^2$ such that

$$\phi(p^{<n}\mathbb{Z}^\omega) \subseteq \beta_i \quad \text{and} \quad (\exists^\infty l \in \mathbb{N})\, z_l \notin \beta_i\,. \tag{4.2}$$

Assume that no pair (i,n) satisfies (4.2). Then for every $m \in \mathbb{N}$ there is some

$$y_m \in \phi(p^{<m}\Sigma^\omega) \text{ such that } (\exists^\infty l)\, z_l \notin \beta_{m-\lfloor\sqrt{m}\rfloor^2} \text{ implies } y_m \notin \beta_{m-\lfloor\sqrt{m}\rfloor^2}.$$

Since ϕ is continuous, the sequence $(y_m)_m$ converges to x. Thus there are $j, m_0 \in \mathbb{N}$ such that $\{x\} \cup \{y_m \,|\, m \geq m_0\} \subseteq \beta_j$ and $(\exists^\infty l \in \mathbb{N})\, z_l \notin \beta_j$, because $\mathcal{B}$ is a limit base of $\mathfrak{X}$. But for $m := (m_0 + j)^2 + j$ we have chosen y_m from $X \setminus \beta_j$, since $j = m - \lfloor\sqrt{m}\rfloor^2$ and $z_l \notin \beta_j$ for infinitely many $l \in \mathbb{N}$. This yields the contradiction.

We define the function $G : \mathbb{Z}^\omega \to \mathbb{Z}^\omega$ for every $q \in \mathbb{Z}^\omega$ and $m \in \mathbb{N}$ by

$$G(q)(m) := \begin{cases} m - \lfloor\sqrt{m}\rfloor^2 & \text{if } \phi(q^{<m}\mathbb{Z}^\omega) \subseteq \beta_{m-\lfloor\sqrt{m}\rfloor^2} \\ -1 & \text{else.} \end{cases}$$

Then G is continuous w.r.t. the Cantor topology, because $G(q)(m)$ only depends on a *finite* prefix of q. For all $p \in dom(\phi)$ we have

$$\begin{aligned} range\big(G(p)\big) \cap \mathbb{N} &= \big\{ i \in \mathbb{N} \,\big|\, (\exists n \in \mathbb{N})\, \phi(p^{<n}\Sigma^\omega) \subseteq \beta_i \big\} \\ &\subseteq \big\{ i \in \mathbb{N} \,\big|\, \phi(p) \in \beta_i \big\}\,. \end{aligned}$$

By (4.2), this implies $\delta_{\mathfrak{X},\beta}\big(G(p)\big) = \phi(p)$ for every $p \in dom(\phi)$. Hence, $\delta_{\mathfrak{X},\beta}$ has the universal property.

□

From Proposition 11 and Theorem 12, we obtain the following characterization of the class of spaces that have an admissible representation.

Theorem 13 (Characterization Theorem).
A weak limit space $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ has an admissible representation, if and only if $\mathfrak{X}$ has a countable limit base and satisfies the T_0–property.

Proof: It remains to show that $\mathfrak{X}$ satisfies the T_0–property, if there is an admissible representation $\delta :\subseteq \Sigma^\omega \to X$ of $\mathfrak{X}$.

Assume that there are points $x \neq y$ in X such that the constant sequence $(x)_n$ converges to y and $(y)_n$ converges to x. From Axiom (L4) we can deduce that every sequence that consists only of x and y converges both to x and to y. Thus for every subset $A \subseteq \Sigma^\omega$ the function $\phi_A : \Sigma^\omega \to X$ defined by

$$\phi_A(p) := \begin{cases} x & \text{if } p \in A \\ y & \text{else.} \end{cases}$$

is continuous. By universality of δ, there is a continuous function $G_A : \Sigma^\omega \to \Sigma^\omega$ with $\phi_A = \delta \circ G_A$. For all $B \neq A$ the function G_B does not equal G_A. This implies

that the cardinality of the set $\mathcal{F}$ of all total continuous functions $f : \Sigma^\omega \to \Sigma^\omega$ is at least $card(2^{\Sigma^\omega})$. This contradicts the existence of a representation of $\mathcal{F}$ (cf. [Wei00,Wei87]) which implies $card(\mathcal{F}) \leq card(\Sigma^\omega)$.
Hence, $\mathfrak{X}$ satisfies the T_0–property.

□

5 Closure Properties

In this section we present some operators on weak limit spaces that preserve the property of having a countable limit base or the one of having an admissible representation. It turns out that the category AWL whose objects are the weak limit spaces with an admissible representation and whose morphisms are the total sequentially continuous functions is bicartesian–closed.

5.1 Coarsest and Finest Limit Relation

Let Y and Z be sets. For $i \in \mathbb{N}$ let $\mathfrak{X}_i = (X_i, \to_{\mathfrak{X}_i})$ be a weak limit space and $f_i : X_i \to Y$ and $g_i : Z \to X_i$ be functions such that $Y = \bigcup_{i\in\mathbb{N}} range(f_i)$. Then we define two limit relations $\to_f$ on Y and $\to^g$ on Z by

$$(y_n)_n \to_f y_\infty \;:\Longleftrightarrow\; \big(\exists i \in \mathbb{N}\big)\big(\exists (x_n)_n \in X_i^{\mathbb{N}}\big)\big(\exists x_\infty \in X_i\big)\big(\exists n_0 \in \mathbb{N}\big)$$
$$\big((x_n)_n \to_{\mathfrak{X}_i} x_\infty \text{ and } (\forall n \in \{n_0, \ldots, \infty\})\, y_n = f_i(x_n)\big)$$

and

$$(z_n)_n \to^g z_\infty \;:\Longleftrightarrow\; \big(\forall i \in \mathbb{N}\big)(g_i(z_n))_n \to_{\mathfrak{X}_i} g_i(z_\infty)\,.$$

Obviously, $\to_f$ is the finest (smallest) limit relation on Y and $\to^g$ is the coarsest (largest) one on Z such that for all $i \in \mathbb{N}$ the functions f_i and g_i are continuous with respect to these limit relations (and the ones of the spaces $\mathfrak{X}_i$).

The following lemma shows how to construct limit bases for the spaces $\mathfrak{Y} = (Y, \to_f)$ and $\mathcal{Z} = (Z, \to^g)$ from given countable limit bases of the spaces $\mathfrak{X}_i$.

Proposition 14.
Let $\mathcal{B}_i$ be a limit base of $\mathfrak{X}_i$ for all $i \in \mathbb{N}$.

(1) If $\mathcal{B}_i$ is countable for all $i \in \mathbb{N}$, then

$$\mathcal{D}_Y := \big\{ f_i(B_0 \cap \ldots \cap B_k) \,\big|\, i, k \in \mathbb{N} \text{ and } \{B_0, \ldots, B_k\} \subseteq \mathcal{B}_i \big\}$$

is a countable limit base of the space $\mathfrak{Y} = (Y, \to_f)$.

(2) The family

$$\mathcal{D}_Z := \big\{ g_i^{-1}(B) \,\big|\, i \in \mathbb{N} \text{ and } B \in \mathcal{B}_i \big\}$$

is a limit base of the space $\mathcal{Z} = (Z, \to^g)$.

Proof:

(1) Since $Y = \bigcup_{i\in\mathbb{N}} range(f_i)$, we have $(y)_n \to_f y$ for every $y \in Y$.
Let $(y_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence of $\mathfrak{Y}$. Then there are some $i, n_0 \in \mathbb{N}$ and a convergent sequence $(x_n)_{n\in\overline{\mathbb{N}}}$ of $\mathfrak{X}_i$ with $y_n = f_i(x_n)$ for all $n \in \{n_0, \ldots, \infty\}$. Since $\mathcal{B}_i$ is limit base of $\mathfrak{X}_i$, there is a sequence $\beta_0, \beta_1, \ldots$ such that

$$\big\{\beta_k \,\big|\, k \in \mathbb{N}\big\} = \big\{B \in \mathcal{B}_i \,\big|\, x_\infty \in B \text{ and } (\forall^\infty n \in \mathbb{N})\, x_n \in B\big\}\,.$$

Let $(z_n)_n \in Y^{\mathbb{N}}$ be a sequence that does not converge to y_∞.
Assume that $(z_n)_n$ is eventually in $f_i(\beta_0 \cap \ldots \cap \beta_k)$ for all $k \in \mathbb{N}$. Then there is a strictly increasing sequence $l_0, l_1, \ldots$ of natural numbers with $\{z_n \,|\, n \geq l_k\} \subseteq f_i(\beta_0 \cap \ldots \cap \beta_k)$ for all $k \in \mathbb{N}$. There exists a sequence $(a_n)_n \in X_i^{\mathbb{N}}$ such that $a_n \in \beta_0 \cap \ldots \cap \beta_k$ and $f_i(a_n) = z_n$ holds for all $k \in \mathbb{N}$ and all $n \in \{l_k, \ldots, l_{k+1} - 1\}$. Since $(z_n)_n$ does not converge to y_∞ and $f_i(a_n) = z_n$ holds for all $n \geq l_0$, $(a_n)_n$ does not converge to x_∞ by definition of $\to_f$. Thus there is some k_0 such that $(a_n)_n$ is not eventually in β_{k_0}, because $\mathcal{B}_i$ is a limit base of $\mathfrak{X}_i$. This contradicts the fact that β_{k_0} contains a_n for all $n \geq l_{k_0}$.
Hence there is some $k \in \mathbb{N}$ such that $(z_n)_n$ is not eventually in $f_i(\beta_0 \cap \ldots \cap \beta_k)$. Obviously $f_i(\beta_0 \cap \ldots \cap \beta_k)$ contains y_∞ and y_n for almost all $n \in \mathbb{N}$. Therefore, the countable family $\mathcal{D}_Y$ is a limit base of $\mathfrak{Y}$.

(2) Obviously, for every $z \in Z$ the constant sequence $(z)_n$ converges to z in $\mathcal{Z}$.
Let $(y_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence of $\mathcal{Z}$, and let $(z_n)_{n\in\mathbb{N}}$ be a sequence that does not converge to y_∞. Then there is some $i \in \mathbb{N}$ such that $\big(g_i(z_n)\big)_n$ does not converge to $g_i(y_\infty)$. Since $\big(g_i(y_n)\big)_{n\in\overline{\mathbb{N}}}$ is a convergent sequence of $\mathfrak{X}_i$ and $\mathcal{B}_i$ is a limit base of $\mathfrak{X}_i$, there is a set $B \in \mathcal{B}_i$ such that $\big(g_i(y_n)\big)_{n\in\overline{\mathbb{N}}}$ is eventually in B and $\big(g_i(z_n)\big)_n$ is not eventually in B. Obviously we have

$$y_\infty \in g_i^{-1}(B),\ (\forall^\infty n)\, y_n \in g_i^{-1}(B) \text{ and } (\exists^\infty n)\, z_n \notin g_i^{-1}(B)\,.$$

We conclude that $\mathcal{D}_Z$ is a limit base of $\mathcal{Z}$.

□

Even if the limit relations of the spaces $\mathfrak{X}_i$ are induced by topologies, the space $\mathfrak{Y}$ is not necessarily a limit space. On the other hand, $\mathfrak{Y}$ is a weak limit space, if the spaces $\mathfrak{X}_i$ are weak limit spaces. This closure property justifies to consider weak limit spaces. $\mathcal{Z}$ is a limit space, if all the spaces $\mathfrak{X}_i$ are limit spaces, and $\mathcal{Z}$ is a topological space, if all the spaces $\mathfrak{X}_i$ are topological ones.

5.2 Cartesian Product

For every $i \in \mathbb{N}$ let $\mathfrak{X}_i = (X_i, \to_{\mathfrak{X}_i})$ be a weak limit space. On the Cartesian Product $\prod_{i\in\mathbb{N}} X_i$ a limit relation $\to_\otimes$ is defined by

$$(y_n)_n \to_\otimes x \ :\Longleftrightarrow\ \big(\forall i \in \mathbb{N}\big)\, \big(pr_i(y_n)\big)_n \to_{\mathfrak{X}_i} pr_i(x)$$

for all sequences $(y_n)_n \in (\prod_{i\in\mathbb{N}} X_i)^{\mathbb{N}}$ and $x \in \prod_{i\in\mathbb{N}} X_i$, where $pr_i : \prod_{j\in\mathbb{N}} X_j \to X_i$ denotes the i–th projection function. We denote the space $\big(\prod_{i\in\mathbb{N}} X_i, \to_{\otimes}\big)$ by $\bigotimes_{i\in\mathbb{N}} \mathfrak{X}_i$. A limit relation for the finite Cartesian Product is constructed accordingly. $\to_{\otimes}$ is the coarsest limit relation on $\prod_{i\in\mathbb{N}} X_i$ such that the projections become continuous. Thus by Proposition 14, $\bigotimes_{i\in\mathbb{N}} \mathfrak{X}_i$ has a countable limit base, if all the spaces $\mathfrak{X}_i$ have countable limit bases. Since the product space $\bigotimes_{i\in\mathbb{N}} \mathfrak{X}_i$ satisfies the T_0–property, iff all the spaces $\mathfrak{X}_i$ do so, we conclude by Theorem 13 that the countable Cartesian Product preserves the existence of an admissible representation. Of course, this also holds for the finite Cartesian Product. If $\delta_i :\subseteq \Sigma^\omega \to X_i$ is an admissible representation of the space $\mathfrak{X}_i$ for $i \in \{1, \ldots, k\}$, then the representation $[\delta_1, \ldots, \delta_k] :\subseteq \Sigma^\omega \to \prod_{i=1}^k X_i$ defined by

$$[\delta_1, \ldots, \delta_k](p) := \big(\delta_1(\pi_{k,1}(p)), \ldots, \delta_k(\pi_{k,k}(p))\big)\,,$$

where $\pi_{k,i}(p)(j) := p(j \cdot k + i - 1)$, can be easily proven to be an admissible representation of the finite Cartesian Product $\bigotimes_{i=1}^k \mathfrak{X}_i$ (cf. [Wei00]).

5.3 Coproduct

For every $i \in \mathbb{N}$ let $\mathfrak{X}_i = (X_i, \to_{\mathfrak{X}_i})$ be a weak limit space. On the set $Y := \bigcup_{i\in\mathbb{N}}(\{i\} \times X_i)$ we define $\to_{\oplus}$ to be finest (smallest) limit relation such that for all i the embedding $e_i : X_i \to Y$ defined by $e_i(x) := (i, x)$ is continuous. The space $\big(\bigcup_{i\in\mathbb{N}}(\{i\} \times X_i), \to_{\oplus}\big)$ is called the *coproduct* (or *disjoint sum*) of the spaces $\mathfrak{X}_i$. We denote it by $\bigoplus_{i\in\mathbb{N}} \mathfrak{X}_i$. The coproduct $\bigoplus_{i=1}^k \mathfrak{X}_i$ of k weak limit spaces is defined accordingly. Both spaces $\bigoplus_{i=1}^k \mathfrak{X}_i$ and $\bigoplus_{i\in\mathbb{N}} \mathfrak{X}_i$ are weak limit spaces. They satisfy the T_0–property, if all the spaces $\mathfrak{X}_i$ do so. Thus, Theorem 13 and Proposition 14 imply that the class of weak limit spaces with a countable limit base and the class of weak limit spaces with an admissible representation are closed under finite and under countable coproduct. If $\delta_i :\subseteq \mathbb{Z}^\omega \to X_i$ is an admissible representation of $\mathfrak{X}_i$ for all $i \in \mathbb{N}$, then the function $\gamma :\subseteq \mathbb{Z}^\omega \to Y$ defined by

$$\gamma(p) := \delta_{|p(0)|}\big(p(1)p(2)p(3)\ldots\big)$$

can be proven to be an admissible representation of $\bigoplus_{i\in\mathbb{N}} \mathfrak{X}_i$.

5.4 Exponentiation

Let $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ and $\mathfrak{Y} = (Y, \to_{\mathfrak{Y}})$ be weak limit spaces. By $\mathcal{C}(\mathfrak{X}, \mathfrak{Y})$ we denote the set of all total continuous functions from $\mathfrak{X}$ to $\mathfrak{Y}$. We say that a sequence $(f_n)_n$ of functions in $\mathcal{C}(\mathfrak{X}, \mathfrak{Y})$ *converges continuously* to a function $f_\infty \in \mathcal{C}(\mathfrak{X}, \mathfrak{Y})$, iff for every sequence $(\xi_n)_n \in \overline{\mathbb{N}}^{\mathbb{N}}$ converging to ∞ and every convergent sequence $(x_n)_{n\in\overline{\mathbb{N}}}$ of $\mathfrak{X}$ the sequence $\big(f_{\xi_n}(x_n)\big)_n$ converges in $\mathfrak{Y}$ to $f_\infty(x_\infty)$. In this case, we write $(f_n)_n \Rrightarrow_c f_\infty$. If $\mathfrak{Y}$ is a limit space, then obviously this definition of continuous convergence is equivalent to the ordinary one $\big($i.e. $(f_n)_n \Rrightarrow_c f_\infty$, iff $(y_n)_n \to_{\mathfrak{X}} x$ implies $(f_n(y_n))_n \to_{\mathfrak{Y}} f_\infty(x)\big)$. By $\mathfrak{C}(\mathfrak{X}, \mathfrak{Y})$ we denote the space $\big(\mathcal{C}(\mathfrak{X}, \mathfrak{Y}), \Rrightarrow_c\big)$. By the next lemma, $\mathfrak{C}(\mathfrak{X}, \mathfrak{Y})$ is a weak limit space.

Lemma 15. *Let $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ and $\mathfrak{Y} = (Y, \to_{\mathfrak{Y}})$ be weak limit spaces. Then $\mathfrak{C}(\mathfrak{X}, \mathfrak{Y}) = \big(\mathcal{C}(\mathfrak{X}, \mathfrak{Y}), \rightrightarrows_c\big)$ is a weak limit space.*

Proof:

Axiom (L1) follows from the continuity of every function in $\mathcal{C}(\mathfrak{X}, \mathfrak{Y})$.

Axiom (L4): Let $(f_n)_n$ be continuously convergent to some function $f_\infty \in \mathcal{C}(\mathfrak{X}, \mathfrak{Y})$ and let $(k_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence in $\overline{\mathbb{N}}$. We have to show that $(f_{k_n})_n$ converges continuously to f_{k_∞}. Let $(y_n)_n \in X^{\mathbb{N}}$ be a sequence converging to some $x \in X$ and let $(\xi_n)_n$ be a sequence in $\overline{\mathbb{N}}$ converging to ∞. If $k_\infty = \infty$, then $\big(f_{k_{\xi_n}}(y_n)\big)_n$ converges to $f_\infty(x)$, because the sequence $(k_{\xi_n})_n$ converges to ∞ and $(f_n)_n$ converges continuously to f_∞.
Otherwise there is some $n_0 \in \mathbb{N}$ such that $k_{\xi_n} = k_\infty$ for all $n \geq n_0$. Since $(y_{n+n_0})_n$ converges to x by Axiom (L4) and f_{k_∞} is continuous, the sequence $\big(f_{k_{\xi_{n+n_0}}})(y_{n+n_0})\big)_n$ converges to $f_{k_\infty}(x)$. This implies $\big(f_{k_{\xi_n}}(y_n)\big)_n \to_{\mathfrak{Y}} f_{k_\infty}(x)$, because $\mathfrak{Y}$ satisfies Axiom (L5).

Axiom (L5): Let $(f_n)_n$ be a sequence such that $(f_{n+1})_n$ converges continuously to some $f_\infty \in \mathcal{C}(\mathfrak{X}, \mathfrak{Y})$. Let $(y_n)_n \in X^{\mathbb{N}}$ be a sequence converging to some $x \in X$ and let $(\xi_n)_n$ be a sequence in $\overline{\mathbb{N}}$ converging to ∞. Then there is some $n_0 \in \mathbb{N}$ such that $\xi_n \geq 1$ for all $n \geq n_0$. Since $(\xi_{i+n_0} - 1)_i$ converges to ∞ and $(y_{i+n_0})_i$ converges to x, the sequence $\big(f_{\xi_{i+n_0}}(y_{i+n_0})\big)_i$ converges to $f_\infty(x)$. Since $\mathfrak{Y}$ satisfies Axiom (L5), $\big(f_{\xi_n}(y_n)\big)_n$ converges to $f_\infty(x)$. We conclude that $(f_n)_n$ converges continuously to f_∞.

□

By the next theorem, the category whose objects are the weak limit spaces and whose morphisms are the total continuous functions is cartesian–closed.

Theorem 16.
Let $\mathfrak{X} = (X, \to_{\mathfrak{X}})$, $\mathfrak{Y} = (Y, \to_{\mathfrak{Y}})$ and $\mathcal{Z} = (Z, \to_{\mathcal{Z}})$ be weak limit spaces.

(1) The evaluation function $eval : \mathcal{C}(\mathfrak{X}, \mathfrak{Y}) \times X \to Y$ defined by $eval(f, x) := f(x)$ is continuous with respect to $\rightrightarrows_c$, $\to_{\mathfrak{X}}$ and $\to_{\mathfrak{Y}}$.
(2) If $h : Z \times X \to Y$ is continuous, then the function $\Lambda(h) : Z \to \mathcal{C}(\mathfrak{X}, \mathfrak{Y})$ defined by $\Lambda(h)(z)(x) := h(z, x)$ is continuous with respect to $\to_{\mathcal{Z}}$ and $\rightrightarrows_c$.

Proof:

(1) Let $(f_n)_n \in (\mathcal{C}(\mathfrak{X}, \mathfrak{Y}))^{\mathbb{N}}$ be a sequence that converges continuously to a function $f_\infty \in \mathcal{C}(\mathfrak{X}, \mathfrak{Y})$, and let $(x_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence of $\mathfrak{X}$. Since $(n)_n$ converges to ∞, the sequence $\big(eval(f_n, x_n)\big)_n = \big(f_n(x_n)\big)_n$ converges to $f_\infty(x_\infty)$ in $\mathfrak{Y}$. Hence *eval* is continuous with respect to $\rightrightarrows_c$, $\to_{\mathfrak{X}}$ and $\to_{\mathfrak{Y}}$.
(2) Let $(z_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence of $\mathcal{Z}$. Furthermore let $(x_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence of $\mathfrak{X}$ and $(\xi_n)_n$ be a sequence in $\overline{\mathbb{N}}$ converging to ∞.

From Axiom (L4) it follows that $(z_{\xi_n})_n$ converges to z_∞ in $\mathcal{Z}$. By continuity of h the sequence $\big(h(z_{\xi_n}, x_n)\big)_n$ converges to $h(z_\infty, x_\infty)$ in $\mathfrak{Y}$, i.e. $\big(\Lambda(h)(z_{\xi_n})(x_n)\big)_n \to_{\mathfrak{Y}} \Lambda(h)(z_\infty)(x_\infty)$. By definition of $\Rightarrow_c$ this means that $\big(\Lambda(h)(z_n)\big)_n$ converges continuously to $\Lambda(h)(z_\infty)$. Thus $\Lambda(h)$ is continuous. □

For defining a representation of $\mathcal{C}(\mathfrak{X}, \mathfrak{Y})$, we use the *effective* representation $\eta : \Sigma^\omega \to \mathrm{F}^{\omega\omega}$ from [Wei00] (or [KW85,Wei87]) of the set $\mathrm{F}^{\omega\omega}$ of all continuous functions $G :\subseteq \Sigma^\omega \to \Sigma^\omega$ having a G_δ–domain. This representation satisfies the extension property, the utm–Theorem, the computable smn–Theorem, and the continuous smn–Theorem, i.e.

1. for every continuous function $G :\subseteq \Sigma^\omega \to \Sigma^\omega$ there is some $p \in \Sigma^\omega$ such that $\eta(p)$ extends G (extension property);
2. the universal function $u_\eta :\subseteq \Sigma^\omega \times \Sigma^\omega \to \Sigma^\omega$ defined by $u_\eta(p, q) := \eta(p)(q)$ is computable (utm–Theorem) and thus continuous;
3. for every computable function $G :\subseteq \Sigma^\omega \times \Sigma^\omega \to \Sigma^\omega$ there is some computable function $H : \Sigma^\omega \to \Sigma^\omega$ with $\eta(H(p))(q) = G(p, q)$ for all $p, q \in \Sigma^\omega$ (computable smn–Theorem);
4. for every continuous function $G :\subseteq \Sigma^\omega \times \Sigma^\omega \to \Sigma^\omega$ there is some continuous function $H : \Sigma^\omega \to \Sigma^\omega$ with $\eta(H(p))(q) = G(p, q)$ for all $(p, q) \in dom(G)$ (continuous smn–Theorem).

Let $\delta_\mathfrak{X} :\subseteq \Sigma^\omega \to X$ and $\delta_\mathfrak{Y} :\subseteq \Sigma^\omega \to Y$ be admissible representations of the weak limit spaces $\mathfrak{X}$ and $\mathfrak{Y}$, respectively. We define the partial function $[\delta_\mathfrak{X} \to \delta_\mathfrak{Y}] :\subseteq \Sigma^\omega \to \mathcal{C}(\mathfrak{X}, \mathfrak{Y})$ by

$$[\delta_\mathfrak{X} \to \delta_\mathfrak{Y}](p) = f \;:\Longleftrightarrow\; \big(\forall q \in dom(\delta_\mathfrak{X})\big)\, \delta_\mathfrak{Y}\big(\eta(p)(q)\big) = f\big(\delta_\mathfrak{X}(p)\big)$$

for all $p \in \Sigma^\omega$ and $f \in \mathcal{C}(\mathfrak{X}, \mathfrak{Y})$. By Theorem 6, for every total continuous function $f : X \to Y$ there is some continuous function $G :\subseteq \Sigma^\omega \to \Sigma^\omega$ realizing f with respect to $\delta_\mathfrak{X}$ and $\delta_\mathfrak{Y}$. Since by the extension property there is some $p \in \Sigma^\omega$ such that $\eta(p)$ extends G, the function $[\delta_\mathfrak{X} \to \delta_\mathfrak{Y}] :\subseteq \Sigma^\omega \to \mathcal{C}(\mathfrak{X}, \mathfrak{Y})$ is surjective and thus a representation of $\mathcal{C}(\mathfrak{X}, \mathfrak{Y})$.

The next theorem states that $[\delta_\mathfrak{X} \to \delta_\mathfrak{Y}]$ is an admissible representation of the weak limit space $\mathfrak{C}(\mathfrak{X}, \mathfrak{Y})$, if $\delta_\mathfrak{X}$ and $\delta_\mathfrak{Y}$ are admissible ones of $\mathfrak{X}$ and $\mathfrak{Y}$, respectively.

Theorem 17 (Admissibility of $[\delta_\mathfrak{X} \to \delta_\mathfrak{Y}]$).
Let $\mathfrak{X} = (X, \tau_\mathfrak{X})$ and $\mathfrak{Y} = (Y, \tau_\mathfrak{Y})$ be weak limit spaces, and let $\delta_\mathfrak{X} :\subseteq \Sigma^\omega \to X$ and $\delta_\mathfrak{Y} :\subseteq \Sigma^\omega \to Y$ be admissible representations of $\mathfrak{X}$ and $\mathfrak{Y}$, respectively. Then $[\delta_\mathfrak{X} \to \delta_\mathfrak{Y}]$ is an admissible representation of the function space $\mathfrak{C}(\mathfrak{X}, \mathfrak{Y})$.

Proof:

Continuity:

Let $(p_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence in $dom\big([\delta_\mathfrak{X} \to \delta_\mathfrak{Y}]\big)$. For all $n \in \overline{\mathbb{N}}$ let $f_n := [\delta_\mathfrak{X} \to \delta_\mathfrak{Y}](p_n)$. In order to show that $(f_n)_{n\in\mathbb{N}}$ converges continuously to

f_∞ let $(x_n)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence of $\mathfrak{X}$ and $(\xi_n)_n$ be a sequence in $\overline{\mathbb{N}}$ that converges to ∞. By Lemma 5, there is a convergent sequence $(q_n)_{n\in\overline{\mathbb{N}}}$ with $\big(\delta_{\mathfrak{X}}(q_n)\big)_{n\in\overline{\mathbb{N}}} = (x_n)_{n\in\overline{\mathbb{N}}}$. Since $\big\{(p_{\xi_n}, q_n) \,\big|\, n \in \overline{\mathbb{N}}\big\} \subseteq dom(u_\eta)$, $\lim\limits_{n\to\infty} p_{\xi_n} = p_\infty$, $\lim\limits_{n\to\infty} q_n = q_\infty$ hold and $\delta_{\mathfrak{Y}} \circ u_\eta$ is continuous, we have

$$\big(f_{\xi_n}(x_n)\big)_n = \big(\delta_{\mathfrak{Y}}(\eta(p_{\xi_n})(q_n))\big)_n \to_{\mathfrak{Y}} \delta_{\mathfrak{Y}}\big(\eta(p_\infty)(q_\infty)\big) = f_\infty(x_\infty)\,.$$

Hence $\big([\delta_{\mathfrak{X}} \to \delta_{\mathfrak{Y}}](p_n)\big)_n$ converges continuously to $[\delta_{\mathfrak{X}} \to \delta_{\mathfrak{Y}}](p_\infty)$.

Universality:

Let $\phi :\subseteq \Sigma^\omega \to \mathcal{C}(\mathfrak{X},\mathfrak{Y})$ be continuous with respect to $\to_{\Sigma^\omega}$ and $\Rightarrow_{\mathrm{c}}$. We define the "universal" function $u_\phi :\subseteq \Sigma^\omega \times \Sigma^\omega \to Y$ by

$$u_\phi(p,q) := \big(\phi(p)\big)\big(\delta_{\mathfrak{X}}(q)\big)$$

for all $(p,q) \in dom(u_\phi) := dom(\phi) \times dom(\delta_{\mathfrak{X}})$.

Let $\big((p_n,q_n)\big)_{n\in\overline{\mathbb{N}}}$ be a convergent sequence in $dom(u_\phi)$. By continuity of ϕ and $\delta_{\mathfrak{X}}$, $\big(\phi(p_n)\big)_n$ converges continuously to $\phi(p_\infty)$ and $\big(\delta_{\mathfrak{X}}(q_n)\big)_n$ converges to $\delta_{\mathfrak{X}}(q_\infty)$. Thus the sequence $\big(u_\phi(p_n,q_n)\big)_n$ converges in $\mathfrak{Y}$ to $u_\phi(p_\infty,q_\infty)$. Hence u_ϕ is continuous with respect to $\to_{\Sigma^\omega}$ and $\to_{\mathfrak{Y}}$.

We define $u' :\subseteq \Sigma^\omega \to Y$ by

$$u'\big(p(0)q(0)p(1)q(1)\ldots\big) := u_\phi(p,q)$$

for all $p,q \in \Sigma^\omega$. Then u' is continuous with respect to $\to_{\Sigma^\omega}$ and $\to_{\mathfrak{Y}}$. By universality of $\delta_{\mathfrak{Y}}$, there is a continuous function $G :\subseteq \Sigma^\omega \times \Sigma^\omega \to \Sigma^\omega$ with

$$u_\phi(p,q) = u'\big(p(0)q(0)p(1)q(1)\ldots\big) = \delta_{\mathfrak{Y}}\big(G(p,q)\big)$$

for all $p,q \in dom(u_\phi)$. The continuous smn–Theorem yields a continuous function $H : \Sigma^\omega \to \Sigma^\omega$ with $\eta(H(p))(q) = G(p,q)$ for all $(p,q) \in dom(G)$. For all $p \in dom(\phi)$ and $q \in dom(\delta_{\mathfrak{X}})$ we obtain

$$\big(\phi(p)\big)\big(\delta_{\mathfrak{X}}(q)\big) = u_\phi(p,q) = \delta_{\mathfrak{Y}}\big(G(p,q)\big) = \delta_{\mathfrak{Y}}\big(\eta(H(p))(q)\big)\,,$$

i.e. $\phi(p) = [\delta_{\mathfrak{X}} \to \delta_{\mathfrak{Y}}](H(p))$ and $\phi = [\delta_{\mathfrak{X}} \to \delta_{\mathfrak{Y}}] \circ (H|_{dom(\phi)})$.

This proves $\phi \leq_{\mathrm{t}} [\delta_{\mathfrak{X}} \to \delta_{\mathfrak{Y}}]$.

□

From two given countable limit bases $\mathcal{B}_{\mathfrak{X}}$ and $\mathcal{B}_{\mathfrak{Y}}$ of the weak limit spaces $\mathfrak{X}$ and $\mathfrak{Y}$, a countable limit base $\mathcal{D}$ of the space $\mathfrak{C}(\mathfrak{X},\mathfrak{Y})$ can be directly constructed by letting

$$\mathcal{D} := \big\{\mathcal{F}(A_0 \cap \ldots \cap A_k, B) \,\big|\, k \in \mathbb{N}\,,\ \{A_0,\ldots,A_k\} \subseteq \mathcal{B}_{\mathfrak{X}}\,,\ B \in \mathcal{B}_{\mathfrak{Y}}\big\}\,,$$

where $\mathcal{F}(A,B) := \big\{f \in \mathcal{C}(\mathfrak{X},\mathfrak{Y}) \,\big|\, f(A) \subseteq B\big\}$. This construction works also in the case that $\mathfrak{X}$ and/or $\mathfrak{Y}$ do not satisfy the T_0–property.

From Sections 5.2 and 5.3 and from Theorems 16 and 17 we conclude that the category AWL of weak limit spaces having an admissible representation is bicartesian–closed. With the help of the utm–Theorem and the computable smn–Theorem for η, the category whose objects are the weak limit spaces equipped with an admissible representation and whose morphisms are the total functions being computable with respect to the representations of the domain and the codomain can be proven to be bicartesian–closed as well.

5.5 Topologization and Quotient Spaces

Let $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ be a weak limit space with a countable limit base. Then the associated topological space $\big(X, \mathit{seq}(\to_{\mathfrak{X}})\big)$ (see Section 2.5) turns out to have also a countable limit base. The idea of the proof is to embed $\big(X, \mathit{seq}(\to_{\mathfrak{X}})\big)$ into the space $\mathfrak{C}\big(\mathfrak{C}(\mathfrak{X}, \mathfrak{Si}), \mathfrak{Si}\big)$, where $\mathfrak{Si}$ is the Sierpiński–space having $\big\{\emptyset, \{0\}, \{0,1\}\big\}$ as its topology. We omit the details.
An interesting consequence of this result is that every T_0–space $\mathfrak{Y} = (Y, \tau_{\mathfrak{Y}})$ which is a topological quotient of a countably–based space $\mathcal{Z} = (Z, \tau_{\mathcal{Z}})$ has an admissible representation. For the proof, one applies Proposition 14(1) to the corresponding quotient mapping q and shows that the associated topology of the resulting weak limit space $(Y, \to_q)$ is equal to $\tau_{\mathfrak{Y}}$.

6 The Category $\mathsf{PQ_L}$

In [MS00], Menni and Simpson introduce the category $\mathsf{PQ_L}$. Interestingly, this category is closely related to AWL.

The objects of $\mathsf{PQ_L}$ are those limit spaces $\mathfrak{X} = (X, \to_{\mathfrak{X}})$ for which there is a countably–based space $\mathcal{Z} = (Z, \tau_{\mathcal{Z}})$ and an "ω–projecting" function $r : Z \to X$. r is called *ω–projecting*, iff r is continuous and for every countably–based space $\mathfrak{A} = (A, \tau_{\mathfrak{A}})$ and every continuous function $f : A \to X$ there is a continuous function $g : A \to Z$ with $f = r \circ g$. The morphisms of $\mathsf{PQ_L}$ are the continuous functions between the limit spaces in $\mathsf{PQ_L}$. It turns out that a limit space $\mathfrak{X}$ is an object of $\mathsf{PQ_L}$, if and only if it has a countable limit base. We give a short sketch of the proof.

"$\Longrightarrow$": If $\mathcal{Z}$ is a topological space with the countable base $\mathcal{B}_{\mathcal{Z}}$ and $r : Z \to X$ is ω–projecting, then the family $\big\{r(B) \,\big|\, B \in \mathcal{B}_{\mathcal{Z}}\big\}$ is a countable limit base of $\mathfrak{X}$. This can be proven in a similar way as Proposition 11.

"$\Longleftarrow$": Let $\{\beta_0, \beta_1, \dots\}$ be a countable limit base of $\mathfrak{X}$. As the underlying set of the countably–based space $\mathcal{Z} = (Z, \tau_{\mathcal{Z}})$ to be constructed, we use

$$Z := \Big\{(p,x) \in 2^{\mathbb{N}} \times X \,\Big|\, p \subseteq \{i \in \mathbb{N} \,|\, x \in \beta_i\} \text{ and } \big(\forall (z_n)_n \in X^{\mathbb{N}}\big)\big((z_n)_n \not\to_{\mathfrak{X}} x \implies (\exists i \in p)(\exists^{\infty} n)\, z_n \notin \beta_i\big)\Big\}.$$

As the topology $\tau_{\mathcal{Z}}$, we take the subspace topology of the product topology of the Scott–topology on $2^{\mathbb{N}}$ and the indiscrete topology $\{\emptyset, X\}$ on X. Since

the Scott–topology on $2^{\mathbb{N}}$ has the countable base $\big\{\{p \subseteq \mathbb{N} \,|\, E \subseteq p\} \,\big|\, E \text{ finite}\big\}$, $\mathcal{Z}$ is countably–based. The function $r : Z \to X$ defined by $r(p, x) := x$ can be proven to be ω–projective by using the ideas of the proof of Theorem 12. We omit the details.

In a similar way, one can show that the category $\mathsf{PQ_W}$ of the weak limit spaces which are ω–projections of countably–based spaces is equivalent to the category of weak limit spaces having a countable limit base. Thus the categorical approach by Menni and Simpson and the Type–2–approach in this paper are essentially equivalent.

Menni and Simpson showed in [MS00] that the category $\mathsf{PQ_L}$ is locally cartesian closed. Since the local exponents for T_0 limit spaces have also the T_0–property, we conclude by the above equivalence and Theorem 13 that the category AL of limit spaces with admissible representations is locally cartesian closed as well.

7 Conclusion

We have shown that a large class of spaces are equipped with an admissible representation. This means that these spaces can be appropriately handled in the framework of TTE, i.e. we have realistic computational models for them. The corresponding category AWL includes all countably–based T_0–spaces and has nice closure properties, in particular it is bicartesian–closed. Its subcategory AL consisting of all limit spaces in AWL is equivalent to the subcategory of $\mathsf{PQ_L}$ consisting of all T_0 limit spaces in $\mathsf{PQ_L}$. The notion of *limit bases* enables to characterize the category AWL in an easy way. A countable limit base of a space $\mathfrak{X}$ yields us a *countable access* to $\mathfrak{X}$. Such a countable access is of course necessary for establishing a *realistic* computational model on $\mathfrak{X}$.

References

Bir36. G. Birkhoff: *On the combination of topologies*, Fundamenta Mathematicae 26, pp. 156–166 (1936)

Dud64. R.M. Dudley: *On Sequential Convergence*, Transactions of the American Mathematical Society 112, pp. 483–507 (1964)

Eng89. R. Engelking: *General Topology*, Heldermann, Berlin (1989)

Fra65. S.P. Franklin: *Spaces in which sequences suffice*, Fundamenta Mathematicae 57, pp. 107–115 (1965)

Grz57. A. Grzegorczyk: *On the definitions of computable real continuous functions*, Fundamenta Mathematicae 44, pp. 61–71 (1957)

Kis60. J. Kisyński: *Convergence du Type L*, Colloquium Mathematicum 7, pp. 205–211 (1960)

Ko91. Ker-I Ko: *Complexity Theory of Real Functions*, Birkhäuser, Boston (1991)

KW85. C. Kreitz and K. Weihrauch: *Theory of representations*, Theoretical Computer Science 38, pp. 35–53 (1985)

Kur66. K. Kuratowski: *Topology*, Vol. 1, Academic Press, New York (1966)

MS00. M. Menni and A. Simpson: *Topological and Limit-space Subcategories of Countably–based Equilogical Spaces*, submitted (2000). Available at `http://www.dcs.ed.ac.uk/home/als`

PER89. M. Pour–El and J. Richards: *Computability in Analysis and Physics*, Springer, Berlin (1989)

SHT99. V. Stoltenberg–Hansen and J.V. Tucker: *Concrete models of computation for topological algebras*, Theoretical Computer Science 219, pp. 347–378 (1999)

Ury26. P. Urysohn: *Sur les classes L de M. Fréchet*, Enseignement Mathématique 25, pp. 77–83 (1926)

Wei87. K. Weihrauch: *Computability*, Springer, Berlin (1987)

Wei95. K. Weihrauch: *A Foundation for Computable Analysis*, EATCS Bulletin Nr. 57, pp. 167–182 (October 1995)

Wei96. K. Weihrauch: *A Foundation for Computable Analysis*, in: Proceedings of DMTCS '96 (Discrete Mathematics and Theoretical Computer Science), Springer, Berlin pp. 66–89 (1997)

Wei00. K. Weihrauch: *Computable Analysis*, Springer, Berlin (2000)

Characterization of the Computable Real Numbers by Means of Primitive Recursive Functions

Dimiter Skordev

Faculty of Mathematics and Informatics
"St. Kl. Ohridski" University of Sofia
5 blvd. J. Bourchier, BG-1164 Sofia, Bulgaria
skordev@fmi-uni.sofia.bg

Abstract. One usually defines the notion of a computable real number by using recursive functions. However, there is a simple way due to A. Mostowski to characterize the computable real numbers by using only primitive recursive functions. We prove Mostowski's result differently and apply it to get other simple characterizations of this kind. For instance, a real number is shown to be computable if and only if it belongs to all members of some primitive recursive sequence of nested intervals with rational end points and with lengths arbitrarily closely approaching 0.

Introduction

Let $\mathbb{N}$ and $\mathbb{Q}$ be the set of the non-negative integers and the set of the rational numbers, respectively. A function $A : \mathbb{N} \longrightarrow \mathbb{Q}$ is said to be recursive if it can be represented in the form

$$A(n) = \frac{u(n) - v(n)}{w(n) + 1} \ , \tag{1}$$

where u, v, w are recursive functions from $\mathbb{N}$ into $\mathbb{N}$ (in the case when the values of A are non-negative the second term in the numerator can be omitted).[1] The notion of a primitive recursive function from $\mathbb{N}$ into $\mathbb{Q}$ is defined in a similar way, namely one must replace "recursive" by "primitive recursive" in the above definition. If we regard a function A from $\mathbb{N}$ into $\mathbb{N}$ as a function from $\mathbb{N}$ into $\mathbb{Q}$ then the above notions are clearly equivalent to the ordinary recursiveness and to the ordinary primitive recursiveness of A, respectively. Of course, we can treat quite similarly also those $\mathbb{Q}$-valued functions that depend on several natural arguments.

One usually defines the notion of a computable (or recursive) real number by using recursive functions.[2] Any of the definitions implies the following statement (according to [2], it can be attributed to S. Mazur): a real number α is

[1] A definition using effective enumeration of the set $\mathbb{Q}$ can also be found in the literature.

[2] Here are several of the places where one can find some (mutually equivalent) definitions of this kind: the paper [4], § 12 of [9], Exercise 15-34 in [5], Lemma 4.2.1 and Exercise 4.2.1 in [10] (other relevant references can be found in Section II.4 of [3]).

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 296–309, 2001.

computable if and only if there is a recursive function A from $\mathbb{N}$ into $\mathbb{Q}$ that satisfies for any n in $\mathbb{N}$ the inequality

$$|A(n) - \alpha| \leq \frac{1}{n+1} \tag{2}$$

It is not permissible to replace "recursive" by "primitive recursive" in this statement (cf. Appendix 1). Nevertheless A. Mostowski showed in [2] that such a replacement is possible if we allow the right-hand side of (2) to be a suitable primitive recursive function from $\mathbb{N}$ into $\mathbb{Q}$ and to depend on the choice of the number α (unfortunately the paper [2] has been not known to us when writing the preliminary version [8] of the present paper). We shall give here another proof of Mostowski's result; certain issues related to this result will be also studied.

It will be useful in Section 2 to consider the notion of primitive recursiveness also for partial functions from $\mathbb{N}$ into $\mathbb{Q}$. We adopt the following definition: a partial function from $\mathbb{N}$ into $\mathbb{Q}$ is called *primitive recursive* if it is the restriction of some primitive recursive total function from $\mathbb{N}$ into $\mathbb{Q}$ to some primitive recursive subset of $\mathbb{N}$. The requirement for a partial function A from $\mathbb{N}$ into $\mathbb{Q}$ to be primitive recursive is equivalent to its representability in the form

$$A(n) = \frac{u(n) - v(n)}{w(n)}\ , \tag{3}$$

where u, v and w are primitive recursive functions from $\mathbb{N}$ into $\mathbb{N}$ and it is assumed that the domain of A is $\{n \mid w(n) \neq 0\}$.

1 Total Approximations and Localizations

Definition 1. *Let A and E be (total) functions from $\mathbb{N}$ into $\mathbb{Q}$. The pair (A, E) is called a* **total approximation** *of a given real number α if*

$$|A(n) - \alpha| \leq E(n)$$

for any n in $\mathbb{N}$ and there are numbers arbitrarily close to 0 among the values of E. The pair (A, E) is called **primitive recursive** *if both A and E are primitive recursive.*

Theorem 1. *A real number is computable if and only if it has a primitive recursive total approximation.*

Proof. Let α be a real number. If (A, E) is a primitive recursive total approximation of α then the function $s : \mathbb{N} \longrightarrow \mathbb{N}$ defined by

$$s(n) = \min\left\{t \mid t \in \mathbb{N}, E(t) \leq \frac{1}{n+1}\right\}$$

is recursive and we have

$$|A(s(n)) - \alpha| \leq \frac{1}{n+1}$$

for any n in $\mathbb{N}$, hence α is computable. For the other direction of the proof suppose that α is computable. Then there is a recursive function $A : \mathbb{N} \longrightarrow \mathbb{Q}$ that satisfies the inequality (2) for any n in $\mathbb{N}$. We shall find now a surjective primitive recursive function $f : \mathbb{N} \longrightarrow \mathbb{N}$ such that $A(f(i))$ is a primitive recursive function of i (this could be done also by an easy application of the lemma from [2]). Firstly we represent the function A in the form (1) and we choose a system of four unary primitive recursive functions in $\mathbb{N}$ that enumerates the set of all quadruples of the form $(n, u(n), v(n), w(n))$, where $n \in \mathbb{N}$. Then we take as f the first one of these four functions. Clearly for all n in $\mathbb{N}$ we have the inequality

$$|A(f(i)) - \alpha| \leq \frac{1}{f(i)+1} \ ,$$

hence the pair

$$\left(\lambda i.A(f(i)), \lambda i.\frac{1}{f(i)+1}\right) \tag{4}$$

is a primitive recursive total approximation of α. □

Remark 1. The way of reasoning in the above proof can be used also in certain more complicated other situations. In the concrete situation considered here, however, a simplification is possible, namely the second part of the proof can replaced by the following somewhat shorter reasoning. Let α be a computable real number. Then consider the set of all quadruples (u, v, w, k) of natural numbers satisfying the inequality

$$\left|\frac{u-v}{w+1} - \alpha\right| < \frac{1}{k+1} \ . \tag{5}$$

This set is not empty and it is recursively enumerable.[3] Hence it can be enumerated by a system U, V, W, K of four primitive recursive functions in $\mathbb{N}$, and the corresponding pair

$$\left(\lambda n.\frac{U(n)-V(n)}{W(n)+1}, \lambda n.\frac{1}{K(n)+1}\right)$$

is a primitive recursive total approximation of α.

Suppose we have some primitive recursive total approximations (A, E) and (B, F) of two given real numbers α and β and we want to find primitive recursive total approximations of the numbers $\alpha+\beta$ and $\alpha\beta$. This turns out to be a little

[3] The recursive enumerability in question could be considered well-known. Still let us mention the following way to see it: we take a recursive function A from $\mathbb{N}$ into $\mathbb{Q}$ satisfying (2) for any n in $\mathbb{N}$, and we observe that (5) holds if and only if

$$\left|\frac{u-v}{w+1} - A(n)\right| + \frac{1}{n+1} < \frac{1}{k+1}$$

for some some n in $\mathbb{N}$.

troublesome in the general case, since, roughly speaking, the values of $E(n)$ and $F(n)$ are not obliged to become small for one and the same values of n. To overcome this problem we could use certain special kinds of total approximations.

Definition 2. *Let (A, E) be a total approximation of a given real number. We call (A, E)* **acceptable** *if the sequence $E(0), E(1), E(2), \ldots$ converges to 0,*[4] *and* **stable** *if the function E is monotonically decreasing.*[5]

As it is easily seen, if (A, E) and (B, F) are acceptable total approximations of the real numbers α and β then $(A + B, E + F)$ and $(AB, |B|E + |A|F + EF)$ are acceptable total approximations of $\alpha + \beta$ and $\alpha\beta$, respectively. Of course, if (A, E) and (B, F) are primitive recursive then so are $(A + B, E + F)$ and $(AB, |B|E + |A|F + EF)$.

Clearly any stable total approximation is acceptable. If (A, E) and (B, F) are stable primitive recursive total approximations of the real numbers α and β then the primitive recursive total approximation $(A + B, E + F)$ of the number $\alpha + \beta$ is a stable one, and the pair $(AB, (|B(0)| + F(0))E + (|A(0)| + E(0))F + EF)$ is a stable primitive recursive total approximation of the number $\alpha\beta$ (unfortunately the stability of (A, E) and (B, F) does not always guarantee the stability of the other total approximation of $\alpha\beta$ considered above).

Theorem 2. *A real number is computable if and only if it has a stable primitive recursive total approximation.*

Proof. In view of Theorem 1, it is sufficient to show that each primitive recursive total approximation of a real number can be transformed into a stable one. Let (A, E) be a primitive recursive total approximation of a real number α. We set

$$\begin{aligned} E'(n) &= \min\{E(i) \mid 0 \le i \le n\} \ , \\ k(n) &= \min\{i \mid 0 \le i \le n,\ E(i) = E'(n)\} \ , \\ A'(n) &= A(k(n)) \ . \end{aligned}$$

Then (A', E') is a stable primitive recursive total approximation of α. □

One often uses intervals with rational end points for the localization of a real number.

[4] This condition is not a good one from a constructive point of view. From such a point of view it would be preferable to impose the stronger requirement that $E(0), E(1), E(2), \ldots$ effectively converges to 0, i.e. to require the existence of a recursive function $\nu : \mathbb{N} \longrightarrow \mathbb{N}$ such that $E(n) \le 1/(k + 1)$ whenever $k, n \in \mathbb{N}$ and $n \ge \nu(k)$ (such a function surely exists in the case of a stable primitive recursive total approximation considered later). A relatively simple equivalent requirement using primitive recursive functions instead of recursive ones is given in Appendix 3.

[5] Only stable primitive recursive total approximations have been considered in the preliminary version [8] of this paper, and they have been called primitive recursive approximations there.

Definition 3. *Let A_0 and A_1 be (total) functions from $\mathbb{N}$ into $\mathbb{Q}$, and let α be a real number. The pair (A_0, A_1) is called a* **total localization** *of α if*

$$A_0(n) \leq \alpha \leq A_1(n)$$

for any n in $\mathbb{N}$ and the set $\{A_1(n) - A_0(n) \mid n \in \mathbb{N}\}$ contains numbers arbitrarily close to 0.

Definition 4. *A total localization (A_0, A_1) of a given real number is called* **nested** *if the function A_0 is monotonically increasing and the function A_1 is monotonically decreasing.*[6]

The next two theorems characterize the computable real numbers in the terms of primitive recursive total localizations.

Theorem 3. *A real number is computable if and only if it has a primitive recursive total localization.*

Proof. We apply Theorem 1. If (A_0, A_1) is a primitive recursive total localization of a real number α then the corresponding pair

$$\left(\frac{A_1 + A_0}{2}, \frac{A_1 - A_0}{2}\right) \tag{6}$$

is a primitive recursive total approximation of α, hence α is computable. Conversely, if α is computable and (A, E) is a primitive recursive total approximation of α, then the corresponding pair $(A - E, A + E)$ is a primitive recursive total localization of α. □

Theorem 4. *A real number is computable if and only if it has a nested primitive recursive total localization.*

Proof. It is sufficient to show how to transform any primitive recursive total localization of a real number into a nested one. Let α be a real number and let (A_0, A_1) be a primitive recursive total localization of α. If we define functions A_0' and A_1' from $\mathbb{N}$ into $\mathbb{Q}$ by setting

$$A_0'(n) = \max\{A_0(i) \mid 0 \leq i \leq n\}\ ,\quad A_1'(n) = \min\{A_1(i) \mid 0 \leq i \leq n\}\ ,$$

then (A_0', A_1') will be a nested primitive recursive total localization of α. □

Remark 2. If α is a real number and (A_0, A_1) is a nested primitive recursive total localization of α, then the pair (6) is a stable primitive recursive total approximation of α. This can be used for proving Theorem 2 in another way.

[6] In [8] only nested primitive recursive total localizations have been considered, and no term for them has been introduced there.

Remark 3. We succeeded to characterize the computable real numbers by means of primitive recursive functions mainly because every recursively enumerable set has a primitive recursive enumeration (cf. the proof of Theorem 1 and especially Remark 1). One could replace the primitive recursive functions in the above characterizations by functions elementary in Kalmár's sense [1] or by lower elementary functions in Skolem's sense [6], and also by elementary definable ones in the sense of [7][7] (all these elementariness notions can be extended to functions from $\mathbb{N}$ into $\mathbb{Q}$ similarly to the extension of recursiveness and primitive recursiveness).

2 Partial Approximations and Localizations

The primitive recursive total approximations and total localizations of the computable real numbers have certain drawbacks. The next example illustrates this for the case of approximations, but the situation is similar also in the case of localizations.

Example 1. There is no pair of primitive recursive operators that transform each stable primitive recursive total approximation of a non-zero computable real number into a primitive recursive total approximation of its reciprocal.[8] In fact, suppose there is a pair of primitive recursive operators with this property. Let $t : \mathbb{N} \longrightarrow \mathbb{N}$ be a recursive function such that t has a primitive recursive graph, but t is not primitive recursive. For any k and n in $\mathbb{N}$ let us set

$$a_k = \frac{1}{t(k)+1}, \quad A_k(n) = \frac{1}{\min\{t(k), n\}+1}, \quad E(n) = \frac{1}{n+1} .$$

Then for any k in $\mathbb{N}$ the pair (A_k, E) is a stable primitive recursive total approximation of the number a_k. Since $A_k(n)$ is primitive recursive also as a function of both k and n, the assumption we made allows us to conclude the existence of $A'_k(n)$ and $E'_k(n)$ that are primitive recursive as functions of k and n and satisfy the condition

$$|A'_k(n) - (t(k)+1)| \leq E'_k(n)$$

for all k and n in $\mathbb{N}$.[9] In particular, we shall have

$$|A'_k(0) - (t(k)+1)| \leq E'_k(0)$$

[7] The elementary definable functions form a subclass of the lower elementary ones, but we unfortunately do not know whether the two classes are different.

[8] We skip the details concerning the notion of a primitive recursive operator acting on $\mathbb{Q}$-valued functions of natural arguments. Hopefully it would be enough to mention that a reduction to ordinary primitive recursive operators is possible through replacing the $\mathbb{Q}$-valued functions by triples of $\mathbb{N}$-valued functions in accordance with the representation (1).

[9] One may use the fact that for any primitive recursive operator its extensions in the sense of [9], § 11, Subsection 3, preserve the primitive recursiveness (actually we need the following instance of the mentioned fact: if Γ is a primitive recursive operator acting on pairs of functions from $\mathbb{N}$ into $\mathbb{Q}$ and transforming them again into such functions, then the function $\lambda kn.\Gamma(A_k, E)(n)$ is primitive recursive).

and this provides us with a primitive recursive upper bound of the function t. On the other hand, such an upper bound cannot exist due to the choice of that function.

In order to avoid the indicated drawbacks, we shall consider now partial approximations and partial localizations of the real numbers. We start with the case of approximations.

Definition 5. *Let A and E be functions from some subset D of $\mathbb{N}$ into $\mathbb{Q}$. The pair (A, E) is called a* **(partial) approximation** *of a given real number α if*

$$|A(n) - \alpha| \leq E(n)$$

for any n in D and there are numbers arbitrarily close to 0 among the values of E. The pair (A, E) is called **primitive recursive** *if both A and E are primitive recursive.*[10]

Theorem 5. *A real number is computable if and only if it has a primitive recursive approximation.*

Proof. Let α be a real number. Similarly to the proof of Theorem 1, if (A, E) is a primitive recursive approximation of α then the function $s : \mathbb{N} \longrightarrow \mathbb{N}$ defined by

$$s(n) = \min\left\{t \mid t \in \mathrm{dom}(E), E(t) \leq \frac{1}{n+1}\right\}$$

is recursive and we have

$$|A(s(n)) - \alpha| \leq \frac{1}{n+1}$$

for any n in $\mathbb{N}$, hence α is computable. On the other hand, if α is computable then it has a primitive recursive total approximation (A, E) by Theorem 1, and clearly (A, E) is a primitive recursive approximation of α. □

Remark 4. The application of Theorem 1 in the above proof brings a certain non-uniformity in it, since the primitive recursive total approximation constructed in the proof of Theorem 1 do not depend in a primitive recursive way on the Gödel number of the given recursive function A from $\mathbb{N}$ into $\mathbb{Q}$ satisfying (2).[11] Nevertheless, Theorem 5 can be proved without introducing the non-uniformity in question. This can be done as follows. As in the proof of Theorem 1, when we suppose that α is a computable real number, we take a recursive function A from $\mathbb{N}$ into $\mathbb{Q}$ satisfying (2), represent A in the form (1) with recursive u, v, w and consider the set of all quadruples of the form $(n, u(n), v(n), w(n))$, where $n \in \mathbb{N}$.

[10] According to the definition of primitive recursiveness for partial functions from $\mathbb{N}$ into $\mathbb{Q}$, this entails the primitive recursiveness of the set D.

[11] The absence of such a primitive recursive dependence is not a defect of the proof of the mentioned theorem and this can be shown by appropriately using the construction from Example 1.

For the new proof, we find a natural number e such that (n, x, y, z) belongs to this set if and only if some t in $\mathbb{N}$ satisfies the condition

$$T_4(e, n, x, y, z, t) \ ,$$

where T_4 is the primitive recursive predicate from Kleene's Normal Form Theorem for the four argument partial recursive functions (the number e can be constructed from the Gödel number of A by an appropriate application of the *s-m-n* Theorem). Let D be the set of the natural numbers i that satisfy the condition

$$T_4(e, (i)_0, (i)_1, (i)_2, (i)_3, (i)_4)$$

(the notations $(i)_j$ have their usual meaning from Recursive Function Theory). If f is the restriction of the function $\lambda i.(i)_0$ to the set D then the corresponding pair (4) is a primitive recursive approximation of α.

The next example shows the existence of a pair of primitive recursive operators that transform each primitive recursive approximation of a non-zero computable real number into a primitive recursive approximation of its reciprocal.[12]

Example 2. Let (A, E) be a primitive recursive approximation of a real number α that is distinct from 0. For all n belonging to the common domain of A and E and satisfying the condition $A(n) \neq 0$ we have

$$\left| \frac{1}{A(n)} - \frac{1}{\alpha} \right| = \frac{|\alpha - A(n)|}{|A(n)||\alpha|} \leq \frac{E(n)}{|A(n)||\alpha|} \ .$$

Let D be the set of the elements n of the mentioned common domain that satisfy the stronger inequality $|A(n)| - E(n) > 0$ (since $|A(n)| - E(n) \geq |\alpha| - 2E(n)$, at least the elements n with $E(n) < |\alpha|/2$ belong to D). Then for all n in D the inequality

$$\left| \frac{1}{A(n)} - \frac{1}{\alpha} \right| \leq \frac{E(n)}{|A(n)|(|A(n)| - E(n))}$$

holds. If A' and E' are, respectively, the restrictions to D of the functions

$$\lambda n. \frac{1}{A(n)}, \quad \lambda n. \frac{E(n)}{|A(n)|(|A(n)| - E(n))}$$

then (A', E') is a primitive recursive approximation of $1/\alpha$.

Similarly to the case of total approximations, we introduce the notions of acceptable and of stable approximation.

Definition 6. *Let (A, E) be an approximation of a given real number, and let D be the common domain of the functions A and E. We call this approximation* **acceptable** *if D contains all sufficiently large natural numbers and $E(n)$*

[12] The needed notion of a primitive recursive operator can be again reduced to the ordinary notion of such operator, this time using the representation (3) of the considered $\mathbb{Q}$-valued functions.

converges to 0 *when* n *tends to infinity.*[13] *The approximation* (A, E) *is called* **stable** *if the set* D *contains* $n+1$*, whenever it contains* n*, and the function* E *is monotonically decreasing.*

Again all stable approximations of a real number are acceptable approximations of it. Note that the approximation (A', E') from Example 2 is acceptable if (A, E) is acceptable, but (A', E') is not necessarily stable when (A, E) is stable.

Theorem 6. *A real number is computable if and only if it has a stable primitive recursive approximation.*

Proof. We use Theorem 5. The "if"-direction follows immediately from it. The other direction of the proof is a slight modification of the corresponding part of the proof of Theorem 2. Namely, we show that each primitive recursive approximation of a real number can be transformed into a stable one. Let (A, E) be a primitive recursive approximation of a real number α, and let D be the common domain of the functions A and E. We denote by D' the set of those numbers from $\mathbb{N}$ that belong to D or are greater than some element of D. Let the functions $E' : D' \longrightarrow \mathbb{Q}$, $k : D' \longrightarrow \mathbb{N}$ and $A' : D' \longrightarrow \mathbb{Q}$ be defined by setting

$$\begin{aligned} E'(n) &= \min\{E(i) \mid 0 \le i \le n,\, i \in D\}\ , \\ k(n) &= \min\{i \mid 0 \le i \le n,\, i \in D,\, E(i) = E'(n)\}\ , \\ A'(n) &= A(k(n))\ . \end{aligned}$$

Then (A', E') is a stable primitive recursive approximation of α. □

Definition 7. *Let* A_0 *and* A_1 *be functions from some subset* D *of* $\mathbb{N}$ *into* $\mathbb{Q}$*, and let* α *be a real number. The pair* (A_0, A_1) *is called a* **(partial) localization** *of* α *if*

$$A_0(n) \le \alpha \le A_1(n)$$

for any n *in* D *and there are numbers arbitrarily close to* 0 *among the values of the function* $A_1 - A_0$*.*

Definition 8. *Let* (A_0, A_1) *be a localization of a given real number, and let* D *be the common domain of* A_0 *and* A_1*. We call this localization* **nested** *if* D *contains* $n+1$*, whenever it contains* n*, the function* A_0 *is monotonically increasing and the function* A_1 *is monotonically decreasing.*

Theorem 7. *A real number is computable if and only if it has a primitive recursive localization.*

Proof. We apply Theorem 5 in the same way as we applied Theorem 1 for the proof of Theorem 3. □

[13] An effective version of this requirement is the following one: there is a recursive function $\nu : \mathbb{N} \longrightarrow \mathbb{N}$ such that $n \in D$ and $E(n) \le 1/(k+1)$ whenever $k, n \in \mathbb{N}$ and $n \ge \nu(k)$.

Theorem 8. *A real number is computable if and only if it has a nested primitive recursive localization.*

Proof. It is sufficient to show how to transform any primitive recursive localization of a real number into a nested one. Let α be a real number and let (A_0, A_1) be a primitive recursive localization of α. Let D be the common domain of A_0 and A_1. We define a set D' as in the proof of Theorem 6 and then we define functions A'_0 and A'_1 from D' into $\mathbb{Q}$ by setting

$$A'_0(n) = \max\{A_0(i) \mid 0 \leq i \leq n,\, i \in D\}\ ,$$
$$A'_1(n) = \min\{A_1(i) \mid 0 \leq i \leq n,\, i \in D\}\ .$$

Then the pair (A'_0, A'_1) is a nested primitive recursive localization of α. □

3 Co-approximations

There is a way to do almost the same as with partial approximations, but without using partial functions.

Definition 9. *Let A and H be (total) functions from $\mathbb{N}$ into $\mathbb{Q}$, and let α be a real number. The pair (A, H) is called a* **co-approximation** *of α if*

$$H(n)|A(n) - \alpha| \leq 1$$

for any n in $\mathbb{N}$, all values of H are non-negative and there are arbitrarily large among them.

Definition 10. *Let (A, H) be a co-approximation of a given real number. We call (A, H)* **normal** *if all values of H belong to $\mathbb{N}$,* **acceptable** *if $H(n)$ diverges to infinity when n tends to infinity,*[14] *and* **stable** *if the function H is monotonically increasing.*[15]

Clearly all stable co-approximations of a real number are acceptable.

Theorem 9. *A real number is computable if and only if it has a primitive recursive co-approximation.*

Proof. We shall use Theorem 5. Let α be a real number. Suppose α is computable and take a primitive recursive approximation (A, E) of α. Let D be the common domain of A and E. We may assume without a loss of generality that all values of E are distinct from 0. Now denote by A' any total primitive recursive extension of A and by H the function from $\mathbb{N}$ into $\mathbb{Q}$ defined as follows:

$$H(n) = \begin{cases} 1/E(n) & \text{if } n \in D, \\ 0 & \text{otherwise.} \end{cases}$$

[14] An effective version of this can be also considered, namely: there is a recursive function $\nu : \mathbb{N} \longrightarrow \mathbb{N}$ such that $H(n) \geq k$ whenever $k, n \in \mathbb{N}$ and $n \geq \nu(k)$.

[15] Only primitive recursive stable normal co-approximations of a real number have been studied in [8] under the name primitive recursive representations of this number.

Then (A', H) is a primitive recursive co-approximation of α. Conversely, suppose α has a primitive recursive co-approximation (A', H). Let D be the set of all n in $\mathbb{N}$ such that $H(n) \neq 0$, and let A be the restriction of A' to D. Then $(A, 1/H)$ is a primitive recursive approximation of α. □

Remark 5. There is no function $H : \mathbb{N} \longrightarrow \mathbb{Q}$ such that every computable real number has a primitive recursive co-approximation with second member H (cf. Appendix 2, where an even stronger statement is proved). From here, taking into account the proof of Theorem 9, we see the non-existence of a function $E : \mathbb{N} \longrightarrow \mathbb{Q}$ such that every computable real number has a primitive recursive approximation with second member E.

Theorem 10. *A real number is computable if and only if it has a normal stable primitive recursive co-approximation.*

Proof. One direction of the proof is clear from Theorem 9. For the other direction suppose α is a computable real number. By Theorem 6, there is a stable primitive recursive approximation (A, E) of α. We may assume that all values of E are distinct from 0. Then the construction from the proof of Theorem 9 is applicable, and it is easy to see that the primitive recursive co-approximation (A', H) obtained by it is now a stable one. For any n in $\mathbb{N}$ let us set $h(n) = [H(n)]$, where $[r]$ denotes the greatest integer not exceeding r. Then the pair (A', h) is a normal stable primitive recursive co-approximation of α. □

Appendix 1

Let $t : \mathbb{N} \longrightarrow \{0, 1\}$ be a recursive function, and let

$$\alpha = \sum_{i=0}^{\infty} \frac{t(i)}{4^i} \ . \tag{7}$$

Clearly α is a computable real number. We shall show now a way of computing the values of t on the base of arbitrary sufficiently close rational approximations of α. Namely, whenever $m \in \mathbb{N}$, $r \in \mathbb{Q}$ and

$$|r - \alpha| \leq \frac{1}{4^{m+1}} \ , \tag{8}$$

the following equality holds:

$$t(m) = \left[\frac{[2 \cdot 4^m r + 1/2] \bmod 4}{2} \right] \ . \tag{9}$$

In fact, the equality

$$2\alpha = \sum_{i=0}^{\infty} \frac{2t(i)}{4^i}$$

implies that

$$[2 \cdot 4^m \alpha] \bmod 4 = 2t(m) \ .$$

On the other hand, the inequality (8) implies that

$$2 \cdot 4^m \alpha \leq 2 \cdot 4^m r + 1/2 \leq 2 \cdot 4^m \alpha + 1 \ ,$$

hence

$$[2 \cdot 4^m r + 1/2] = [2 \cdot 4^m \alpha] + d \ ,$$

where $d = 0$ or $d = 1$. From this equality and the previous one we get

$$[2 \cdot 4^m r + 1/2] \bmod 4 = 2t(m) + d \ ,$$

and from here the equality (9) follows.

Suppose now a primitive recursive function $A : \mathbb{N} \longrightarrow \mathbb{Q}$ satisfies for any n in $\mathbb{N}$ the inequality (2). Then we can satisfy (8) by taking

$$r = A(4^{m+1} - 1) \ ,$$

and the equality (9) with this choice of r leads to the conclusion that t is primitive recursive. Hence if we construct the real number α by using a recursive function $t : \mathbb{N} \longrightarrow \{0, 1\}$ that is not primitive recursive, then there will be no primitive recursive function $A : \mathbb{N} \longrightarrow \mathbb{Q}$ satisfying for any n in $\mathbb{N}$ the inequality (2).

Appendix 2

Let $H : \mathbb{N} \longrightarrow \mathbb{Q}$ be an unbounded non-negative primitive recursive function. We shall construct a computable real number α such that for any primitive recursive function $A : \mathbb{N} \longrightarrow \mathbb{Q}$ the function $\lambda n.H(n)|A(n) - \alpha|$ is unbounded (the result from Appendix 1 can be obtained as a special case of this if we take $H(n) = n + 1$). For the construction of the number α we choose a ternary recursive function z in $\mathbb{N}$ such that any binary primitive recursive function in $\mathbb{N}$ can be obtained from z by substituting some constant for its first argument. Then we define a binary recursive function s and a unary recursive function t in $\mathbb{N}$ as follows:

$$s(k, m) = \min \left\{ n \mid H(n) \geq k \cdot 4^{m+1} \right\} \ , \tag{10}$$

$$t(i) = \overline{\mathrm{sg}}\, z((i)_0, i, s((i)_1, i)) \ , \tag{11}$$

where $\overline{\mathrm{sg}}\, l = 0$ for any l in $\mathbb{N} \backslash \{0\}$, $\overline{\mathrm{sg}}\, 0 = 1$. Making use of the function t, we define the real number α by means of equality (7) from Appendix 1. Suppose that for some primitive recursive function $A : \mathbb{N} \longrightarrow \mathbb{Q}$ the function $\lambda n.H(n)|A(n) - \alpha|$ is bounded. Let k be a positive integer such that $H(n)|A(n) - \alpha| \leq k$ for all n in $\mathbb{N}$. From (10) we conclude that for any m in $\mathbb{N}$ the inequality (8) from Appendix 1 will be satisfied with $r = A(s(k, m))$, hence also the equality (9) will hold with this value of r. The last fact can be written in the form

$$t(m) = f(m, s(k, m)) \ , \tag{12}$$

if we set

$$f(m,n) = \left[\frac{[2 \cdot 4^m A(n) + 1/2] \bmod 4}{2}\right] .$$

Since the function f defined by the above equality is primitive recursive, there is a j in $\mathbb{N}$ such that

$$f(m,n) = z(j,m,n) \tag{13}$$

for all m and n in $\mathbb{N}$. From (12) and (13) we get

$$t(2^j \cdot 3^k) = z(j, 2^j \cdot 3^k, s(k, 2^j \cdot 3^k)) ,$$

and this contradicts the definition (11) of the function t.

Appendix 3

Let E be a non-negative function from $\mathbb{N}$ into $\mathbb{Q}$ (or even into the set of the real numbers). We shall show that the sequence $E(0), E(1), E(2), \ldots$ effectively converges to 0 if and only if there is a monotonically increasing unbounded primitive recursive function $h : \mathbb{N} \longrightarrow \mathbb{N}$ such that $h(n)E(n) \leq 1$ for any n in $\mathbb{N}$. One direction of the proof is obvious. For the other one suppose that $E(0), E(1), E(2), \ldots$ effectively converges to 0 and choose a recursive function $\nu : \mathbb{N} \longrightarrow \mathbb{N}$ such that $E(n) \leq 1/(k+1)$ whenever $k, n \in \mathbb{N}$ and $n \geq \nu(k)$. Let f be a ternary primitive recursive function in $\mathbb{N}$ such that the equality $m = \nu(k)$ holds if and only if $f(m,k,i) = 0$ for some i. For any n in $\mathbb{N}$ let S_n be the set (possibly empty) of all k in $\mathbb{N}$ such that $k \leq n$ holds and some numbers m and i not exceeding n satisfy $f(m,k,i) = 0$. Then we set $h(n) = k+1$ for the greatest k in S_n if S_n is not empty, and we set $h(n) = 0$ otherwise. The function h is evidently primitive recursive. The inequality $h(n)E(n) \leq 1$ holds for any n in $\mathbb{N}$, because $(k+1)E(n) \leq 1$ whenever $f(m,k,i) = 0$ and $n \geq m$. Since S_n is a subset of S_{n+1} for any n in $\mathbb{N}$, it is clear that h is monotonically increasing. To show that h is unbounded, consider an arbitrary k in $\mathbb{N}$, set $m = \nu(k)$, consider some i satisfying $f(m,k,i) = 0$ and choose an integer n satisfying the inequalities $n \geq m$, $n \geq k$, $n \geq i$. Then $k \in S_n$, hence $h(n) \geq k+1$.

It can be seen in a similar way that for any non-negative function H defined on $\mathbb{N}$ the sequence $H(0), H(1), H(2), \ldots$ effectively diverges to infinity if and only if there is a monotonically increasing unbounded primitive recursive function $h : \mathbb{N} \longrightarrow \mathbb{N}$ such that $H(n) \geq h(n)$ for any n in $\mathbb{N}$.

Some Concluding Remarks

The authors interest in simple definitions of computability of real numbers arose from the pedagogical problem how to teach undergraduate students this notion (some of the obtained characterizations of the computable real numbers have been recently used by the author in a lecture course for such students). Hopefully the presented approach could be useful also for closer connecting Computable

Analysis with the problems of numerical computations (estimates of proximity often play a crucial role there and one looks for possibly simple approximation processes and corresponding estimates). The results formulated in Remark 5 point also to the possibility of providing the set of the computable real numbers with some hierarchy concerning the degree of their computability. There are several different ways to introduce such an hierarchy. For example, a real number β could be said to be **less or equally computable** than a real number α if the set of the second members of the primitive recursive approximations of β is a subset of the corresponding set for α (the mentioned results show that there is not a least one among the computable real numbers with respect to the quasi-ordering introduced in this way). Of course, analogues of this can be considered also with using, say, lower elementary functions instead of primitive recursive ones.

Acknowledgments

The author thanks an anonymous referee for many appropriate and useful suggestions. An immense debt of gratitude is owed also to George Barmpalias – he attracted the authors attention to the fact that some essential results of the paper follow immediately from a theorem in Mostowski's paper [2].

References

1. Kalmár, L. A simple example of an unsolvable arithmetical problem. Matematikai és Fizikai Lapok **50** (1943) 1–23 (in Hungarian).
2. Mostowski, A. A lemma concerning recursive functions and its applications. Bull. Acad. Polon. Sci. Cl. III **1** (1953) 277–280.
3. Odifreddi, P. Classical Recursion Theory. The Theory of Functions and Sets of Natural Numbers. North-Holland, Amsterdam/New York/Oxford/Tokyo 1989.
4. Rice, H. G. Recursive real numbers. Proc. of the Amer. Math. Soc. **5** (1954) 784–791.
5. Rogers, H. Jr. Theory of Recursive Functions and Effective Computability. McGraw -Hill Book Company, New York/St. Louis/San Francisco/Toronto/London/Sydney 1967.
6. Skolem, T. A theorem on recursively enumerable sets. In: Abstr. of Short Comm. Int. Congress Math., 1962, Stockholm, p. 11.
7. Skordev, D. On a class of primitive recursive functions. Annuaire de l'Univ. de Sofia, Fac. de Math. **60** (1965/66) 105–111 (in Russian).
8. Skordev, D. A characterization of the computable real numbers by means of primitive recursive functions. In: Computability and Complexity in Analysis (ed. J. Blanck, V. Brattka, P. Hertling, K. Weihrauch), Informatik Berichte **272 : 9** (2000), Fernuniversität – Gesamthochschule in Hagen, pp. 389–394.
9. Uspensky, V. A. Lectures on Computable Functions. Fizmatgiz, Moscow 1960 (in Russian).
10. Weihrauch, K. Computable Analysis. An Introduction. Springer–Verlag, Berlin/Heidelberg 2000.

Effective Fixed Point Theorem over a Non-computably Separable Metric Space

Izumi Takeuti

Graduate School of Informatics, Kyoto University, 606-8501 Japan
takeuti@kuis.kyoto-u.ac.jp

Abstract. This paper shows effective fixed point theorems for computable contractions. Effective fixed point theorem for computable contractions over a computable metric space is easily shown. A function over a computable metric space is represented by a Type-1 function, and the fixed point of a contraction is given by iteration of such Type-1 function. If the contraction is computable, then its fixed point is also computable. If the support space is not computably separable, the method above is not available. The function space of an interval into real numbers is not computably separable with polynomial time computability. This paper show the fixed point theorem for such non-computably separable spaces. This theorem is proved with iteration of Type-2 functionals. As an example of that, this paper shows that Takagi function is a polynomial time computable function.

1 Introduction

When we discuss computable objects in a metric space, we often define the computability and computational complexity of such objects with a separating function, as is studied by Weihrauch [Weihrauch93,Weihrauch00] and others. Our aim is to show two theorems on fixed points of computable contractions. One of the theorems deals with a metric space with a separating function, and the other deals with a metric space without a separating function.

The computability means the computability by total recursive functions in the works of Weihrauch [Weihrauch93,Weihrauch00] or others, for example, by Brattka [Brattka]. We would like to discuss polynomial time computability in this paper.

We use the representations in discussing computational complexity of mathematical objects. A representation is an elements of some structure over natural numbers, for example, $\mathbf{N} \to \mathbf{N}$, $(\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N})$, and so forth. Therefore, we can define computational complexity for representations naturally. We will use the following notation for the relation of mathematical objects and their representations:

$$\text{representation} \vdash \text{represented object}.$$

We will define some representing systems, which are for points of metric spaces, functions over metric spaces, and functionals of functions into functions. We expect a representing system to satisfy the following two properties:
(Existence of the representation) For each x, there is some r such that $r \vdash x$.
(Uniqueness of the represented object) For each x, x' and r, if $r \vdash x$ and $r \vdash x'$ then $x = x'$.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 310–322, 2001.

An object has many representations of it in general. We have not yet proved the existence of representation for functionals, although we have the conjecture that this property holds under some adequate assumption.

In Section 3, we will show the effective fixed point theorem over computably separable metric spaces, which is Theorem 1. This theorem states that a computable contraction over a computably separable metric space has a computable fixed point.

We write $I \stackrel{conti}{\longrightarrow} \mathbf{R}$ for the set of all the continuous functions of the unit interval $I = [0, 1]$ into $\mathbf{R}$, and we write $\mathcal{C}_{I,\mathbf{R}}$ for the set of all the computable functions in $I \stackrel{conti}{\longrightarrow} \mathbf{R}$ under a computational complexity $\mathcal{C}$. The function space $\mathcal{C}_{I,\mathbf{R}}$ is computably separable if the complexity class $\mathcal{C}$ is stronger than or equal to E^3, which is the third class of recursive functions by Grzegorczyk [Grzegorczyk]. As is shown in Proposition 4.8, if the complexity class $\mathcal{C}$ is equal to polynomial time computability, we do not know the adequate separating function for the computable functions in $\mathcal{C}_{I,\mathbf{R}}$. Nevertheless, we discuss the polynomial time computability of objects over the function space $I \stackrel{conti}{\longrightarrow} \mathbf{R}$.

In Section 4, We will give a sufficient condition for the claim that the fixed point of contraction over the function space $I \stackrel{conti}{\longrightarrow} \mathbf{R}$ is computable, which is Theorem 2. We will show a contraction whose fixed point is Takagi function, which is explained in the literature [YHK], and prove that the contraction satisfies the fixed point theorem. Thus, we show that Takagi function is polynomial time computable.

2 Foundation

Notation 2.1 Let $\mathbf{N} = \{0, 1, 2, ...\}$ be the set of 0 and positive integers. The elements of $\mathbf{N}$ are called *natural numbers*.

Notation 2.2 We write P for the set of all the polynomial time computable functions over $\mathbf{N}$. A function f belongs to P iff there is a Turing machine which computes the binary expression of $f(x)$ from the binary expression of x in time $O((\log x)^k)$ for some k independent to x. The input size is as large as $\log x$.

Notation 2.3 We write E^3 for the third class of primitive recursive functions by Grzegorczyk [Grzegorczyk]. A function f belongs to E^3 iff it is computable in time $O(\exp^k n)$ for the input size n and some constant k independent to n. The notation $\exp^k$ is defined as $\exp^0 n = n$ and $\exp^{k+1} n = \exp^k(\exp n)$.

Notation 2.4 We write PR for the set of all the primitive recursive functions.

Notation 2.5 The classes P, E^3 and PR are called *computational complexity classes*. We use $\mathcal{C}$ as a variable which ranges over P, E^3 and PR. The notation $\mathcal{C}_{\mathbf{N}^m,\mathbf{N}^n}$ stands for the set of all functions of $\mathbf{N}^m$ into $\mathbf{N}^n$ in the class $\mathcal{C}$.

Definition 2.6 ($\mathcal{C}$-Function) A functions in $\mathcal{C}_{\mathbf{N}^m,\mathbf{N}^n}$ is called a *$\mathcal{C}$-function*.

Remark 2.7 Here we will give the standard encoding of pairs and lists of natural numbers. We will use them in encoding connective segments in Definition 4.2. There are many ways to encode pairs, or lists, and there are no essential differences among them. The essential properties are Remarks 2.9 for pairs, and Remark 2.11 for lists.

Notation 2.8 The functions $left, right : \mathbf{N} \to \mathbf{N}$ and $pair : \mathbf{N}^2 \to \mathbf{N}$ is defined as follows. Here m and n are natural numbers.

$$left(0) = 0,\ left(2n) = 2right(n),\ left(2n+1) = 2right(n)+1,$$
$$right(2n) = right(2n+1) = left(n),$$
$$pair(0,0) = 0,$$
$$pair(2m,n) = 2pair(n,m),\ pair(2m+1,n) = 2pair(n,m)+1.$$

Remark 2.9 The functions *left*, *right* and *pair* are P-functions, which play the roles of encoding and decoding of pairing. Indeed, we have the following equations:

$$n = pair(left(n), right(n)), m = left(pair(m,n)), n = right(pair(m,n)).$$

Notation 2.10 We write $\langle n_1, n_2, ..., n_l\rangle$ for the code of the list $(n_1, n_2, ..., n_l)$, which is defined as follows:

$$\underbrace{\langle\,\rangle}_{\text{empty list}} = 0, \quad \underbrace{\langle 0,0,...,0\rangle}_{\text{length } l} = 2^{l-1}\ (l \geq 1),$$
$$\langle n_1, n_2, ..., n_{i-1}, 2^m + n_i, n_{i+1}, ..., n_l\rangle$$
$$= 2^{l(m+1)+i-1} + \langle n_1, n_2, ..., n_i, ..., n_l\rangle \text{ where } n_i < 2^m.$$

Remark 2.11 The following functions belong to P.

(length) $\langle n_1, n_2, ..., n_l\rangle \mapsto l : \mathbf{N} \to \mathbf{N}$

(component) $(i, \langle n_1, n_2, ..., n_l\rangle) \mapsto n_i$ (if $i \leq l$) : $\mathbf{N}^2 \to \mathbf{N}$

(concatenation)

$$(\langle n_1, n_2, ..., n_l\rangle, \langle n_{l+1}, n_{l+2}, ..., n_{l'}\rangle) \mapsto \langle n_1, n_2, ..., n_{l'}\rangle : \mathbf{N}^2 \to \mathbf{N}$$

Moreover, it holds that if $l \geq 1$ then $2^{l-1} \leq \langle n_1, n_2, ..., n_l\rangle < (4 \max_i n_i)^l$.

3 Metric Space

Definition 3.1 (Numbered Metric Space) A triple $X = (|X|, d, s)$ is an *numbered metric space* iff $(|X|, d)$ is a metric space, and s is a function of $\mathbf{N}$ into $|X|$ such that the image $s(\mathbf{N})$ is dense in $|X|$. We sometimes write X for $|X|$. The function s is called a *separating function*.

Definition 3.2 (Computable Metric Space) A numbered metric space $X = (|X|, d, s)$ is a *computable metric space* under computational complexity $\mathcal{C}$ iff there is a $\mathcal{C}$-function $f : \mathbf{N}^3 \to \mathbf{N}$ which satisfies the following condition:

$$\frac{f(m,n,n') - 1}{m} \leq d(s(n), s(n')) \leq \frac{f(m,n,n') + 1}{m} \text{ if } m \geq 1.$$

A computable metric space under computational complexity $\mathcal{C}$ is also called a *$\mathcal{C}$-metric space*.

Remark 3.3 There have been several definitions of computable metric spaces. The definition used by Brattka [Brattka] is equivalent to this definition where the computational complexity is Turing computability. This definition above is stronger than the definition which was written by Weihrauch [Weihrauch93].

Example 3.4 Let $|\cdot - \cdot|$ be the function of $\mathbf{R}^2$ into $\mathbf{R}$ which maps (x, y) into $|x - y|$. Let $s_{\mathbf{R}}$ be the function of $\mathbf{N}$ into $\mathbf{R}$ such that

$s_{\mathbf{R}}(0) = 0$

$s_{\mathbf{R}}(4^n + m) = (m - 2^{2n+1})/2^n$ where $0 \leq m < 3 \cdot 4^n$

Then, $(\mathbf{R}, |\cdot - \cdot|, s_{\mathbf{R}})$ is a P-metric space.

Example 3.5 Let I be the unit interval $[0, 1]$. Let $|\cdot - \cdot|$ be the distance function above. Let s_I be the function of $\mathbf{N}$ into I such that

$s_I(0) = 0$

$s_I(2^n + m) = (2m + 1)/2^{n+1}$ where $0 \leq m < 2^n$

Then, $(I, |\cdot - \cdot|, s_I)$ is a P-metric space.

Notation 3.6 Hereafter we write $\mathbf{R}$ and I for the triples above, and regard them as numbered metric spaces.

Definition 3.7 (Representation of a Point) Let $X = (X, d, s)$ be a numbered metric space. For a function $f : \mathbf{N} \to \mathbf{N}$ and a point $x \in X$, the function f *represents* the point x iff for each $n \in \mathbf{N}$ if $n \geq 1$ then $d(x, s(f(n))) \leq 1/n$. We write $f \vdash x$ when f represents x.

Lemma 3.8 (Existence) *For each $x \in X$, there is a function $f : \mathbf{N} \to \mathbf{N}$ such that $f \vdash x$.*

Proof. Because $s(\mathbf{N})$ is dense in X. □

Lemma 3.9 (Uniqueness) *For each $x, x' \in X$, if $f \vdash x$ and $f \vdash x'$ for the same function $f : \mathbf{N} \to \mathbf{N}$, then $x = x'$.*

Proof. Obvious. □

Definition 3.10 ($\mathcal{C}$-Point) Let $X = (|X|, d, s)$ be a $\mathcal{C}$-metric space. A point $x \in X$ is a *$\mathcal{C}$-point* iff there is a $\mathcal{C}$-function $f : \mathbf{N} \to \mathbf{N}$ such that $f \vdash x$. We write $\mathcal{C}(X)$ for the set of all the $\mathcal{C}$-points of X.

Definition 3.11 ($\mathcal{C}$-Fundamental Sequence) Let $X = (|X|, d, s)$ be a $\mathcal{C}$-metric space. A sequence of points $\{x_1, x_2, ...\}$ ($x_n \in X$ for each n) is a $\mathcal{C}$-*fundamental sequence* in X iff $d(x_m, x_n) \leq 1/m + 1/n$ for each $m, n \in \mathbf{N}$ and that there is a function $f \in \mathcal{C}$ such that $x_n = s(f(n))$ for each n.

Remark 3.12 A point $x \in X$ is a $\mathcal{C}$-point iff there is a $\mathcal{C}$-fundamental sequence $\{x_n\}_n$ such that $|x - x_n| \leq 1/n$ for each n.

Definition 3.13 ($\mathcal{C}$-Complete) A $\mathcal{C}$-metric space X is $\mathcal{C}$-*complete* iff each $\mathcal{C}$-fundamental sequence in X converges.

Definition 3.14 ($\mathcal{C}$-Dense) The separating function s is $\mathcal{C}$-*dense* iff all the points in X are $\mathcal{C}$-points.

Definition 3.15 (Computably Separable) A $\mathcal{C}$-metric space is computably separable under computational complexity $\mathcal{C}$, or $\mathcal{C}$-*Separable*, iff the separating function is $\mathcal{C}$-dense. A $\mathcal{C}$-separable $\mathcal{C}$-metric space is called $\mathcal{C}$-*separable metric space*.

Remark 3.16 **(Polynomial Time Computable Point)** The definition of P-points above is quite natural. If $f \vdash x \in \mathbf{R}$ then $s(f(2^l))$ is an approximation of x with error $O(2^{-l})$. Thus it gives the significant digits of size $O(l)$. This fact means that we can calculate an output of size l in polynomial time of l. This is a nice symmetry for P-function, that is, if f is a P-function then we can calculate the value for an input of size l in polynomial time of l.

Remark 3.17 **(Harmonic Modulus)** The conditions of convergence above are written as $d(x, s(f(n))) \leq 1/n$ or $d(x, x_n) \leq 1/n$. We would like to call this modulus $(-)_n \leq 1/n$ *harmonic modulus*. On the other hand, the condition of convergence is written as $d(x, x_n) \leq 2^{-n}$ in many literatures. We would like to call this modulus $(-)_n \leq 2^{-n}$ *geometric modulus*.

If we use the geometric modulus in describing the definition of P-points, we say that a point x is P-point iff there is a function $f : \mathbf{N} \to \mathbf{N}$ such that $d(x, s(f(n))) \leq 2^{-n}$ and $f(n)$ can be calculate in time $O(n^k)$ for some k independent to n.

It seems more natural to use harmonic modulus than geometric modulus in defining P-points. That is because it seems more easy to say that f is P-function than that $f(n)$ can be calculate in time $O(n^k)$. Moreover, 2^n is not P-function anyway.

Remark 3.18 For the notions defined above except for completeness, a notion for P implies that for E^3, and that for E^3 implies that for PR. Namely, a P-function is an E^3-function, and an E^3-function is a PR-function. The inverses do not hold generally. The opposite direction holds for the notion of completeness. A PR-complete space is E^3-complete, and an E^3-complete space is P-complete.

Remark 3.19 Let $X = (X, d, s)$ be a $\mathcal{C}$-metric space. Let Y be a subset such that $s(\mathbf{N}) \subset Y \subset X$. Then (Y, d, s) is also a $\mathcal{C}$-metric space. The $\mathcal{C}$-metric space (Y, d, s) is $\mathcal{C}$-complete iff $Y \supset \mathcal{C}(X)$. The $\mathcal{C}$-metric space (Y, d, s) is $\mathcal{C}$-separable iff $Y \subset \mathcal{C}(X)$.

Definition 3.20 (Representation of a Function) Let $X = (|X|, d_X, s_X)$ and $Y = (|Y|, d_Y, s_Y)$ be numbered metric spaces. Let f, g and h be functions such as $f : |X| \to |Y|$ and $g, h : \mathbf{N}^2 \to \mathbf{N}$. Then, the pair of functions (g, h) *represents* the function f iff the following conditions hold:

$$\forall m, n.\ m \neq 0 \Rightarrow d_Y(f(s_X(n)), s_Y(g(m, n))) \leq 1/m$$
$$\forall m, n \in \mathbf{N}. \forall x, y \in |X|,$$
$$m \neq 0,\ d_X(x, s_X(0)) \leq n,\ d_X(y, s_X(0)) \leq n,\ d_X(x, y) \leq h(m, n)$$
$$\Rightarrow d_Y(f(x), f(y)) \leq 1/m$$

If a pair of functions (g, h) represents a function f then we write $(g, h) \vdash f$.

For $h' : \mathbf{N} \to \mathbf{N}$, $(g, h') \vdash f$ iff $(g, h) \vdash f$, where the function h is put as $h(m, n) = h'(m)$. We also say that (g, h') *represents* f.

Remark 3.21 The first component of the representation mentions the approximation of the value over the separating set, and the second component mentions the modulus of locally uniform continuity.

Remark 3.22 Let f be a function of X to Y. If X is bounded, then the following two conditions are equivalent.

(1) There are $g, h : \mathbf{N}^2 \to \mathbf{N}$ such that $(g, h) \vdash f$.
(2) There are $g : \mathbf{N}^2 \to \mathbf{N}$ and $h' : \mathbf{N} \to \mathbf{N}$ such that $(g, h') \vdash f$.

If X is not bounded, it does not always hold that the two condition above are equivalent.

Hereafter, we discuss functions of bounded domains mainly. Thus we often deal with the representation (g, h') in the condition (2) above.

Remark 3.23 Weihrauch defined the notion of computable functions with Type-2 machines [Weihrauch00]. As for functions over $\mathbf{R}$, or over compact spaces, his definition and our definition are equivalent to each other, although our definition uses functions $g, h : \mathbf{N}^2 \to \mathbf{N}$, which are called Type-1 object. Our definition implies his definition in general. We will discuss this point later in Section 5.2.

Lemma 3.24 (Existence) *Let X and Y be numbered metric spaces. Suppose that each bounded closed set in X is compact. Then each continuous function $f : X \to Y$ has a representation g, h such that $g, h : \mathbf{N}^2 \to \mathbf{N}$ and $(g, h) \vdash f$.*

If X is compact, then each continuous function $f : X \to Y$ has a representation (g, h) such that $g : \mathbf{N}^2 \to \mathbf{N}$, $h : \mathbf{N} \to \mathbf{N}$ and $(g, h) \vdash f$.

Proof. The function g exists because there is a fundamental sequence $s_Y(y_n)$ which converges into $s_Y(f(x))$ for each $x \in X$. The function h exists because a continuous function of a compact domain is uniformly continuous. □

Lemma 3.25 (Uniqueness) *Let X and Y be computable metric spaces. If two continuous functions $f, f' : X \to Y$ have the same representation, that is, $(g,h) \vdash f$ and $(g,h) \vdash f'$, then $f = f'$.*

Proof. That is because f and f' are continuous, $f(s_X(n)) = f'(s_X(n))$ for each n, and $s_X(\mathbf{N})$ is dense in X. □

Definition 3.26 (Computable Function) Let $X = (|X|, d_X, s_X)$ and $Y = (|Y|, d_Y, s_Y)$ be computable metric spaces. A function $f : |X| \to |Y|$ is a *$\mathcal{C}$-function* of X into Y iff there is functions $g, h \in \mathcal{C}_{\mathbf{N}^2,\mathbf{N}}$ such that $(g,h) \vdash f$. We write $\mathcal{C}_{X,Y}$ for the set of all $\mathcal{C}$-functions of X into Y.

Proposition 3.27 *For $x \in \mathcal{C}(X)$ and $f \in \mathcal{C}_{X,Y}$, the value $f(x)$ belongs to $\mathcal{C}(Y)$. For $f \in \mathcal{C}_{X,Y}$ and $g \in \mathcal{C}_{Y,Z}$, the composition $g \circ f$ belongs to $\mathcal{C}_{X,Z}$.*

Proof. Easy. □

Definition 3.28 (Contraction) For a metric space (X,d), a function $f : X \to X$ is a *contraction of maximum magnification* α iff $\alpha < 1$ and for any $x, y \in X$, $d(f(x), f(y)) \leq \alpha \cdot d(x,y)$.

Theorem 1. (Fixed Point Theorem over Numbered Metric Spaces) *Let $X = (X, d, s)$ be a $\mathcal{C}$-complete $\mathcal{C}$-metric space, $f \in \mathcal{C}_{X,X}$ be a contraction, and $\bar{f} : \mathbf{N}^2 \to \mathbf{N}$ be a $\mathcal{C}$-function such that*

$$\forall m, n.\ m \neq 0 \Rightarrow d(f(s_X(n)), s(\bar{f}(m,n))) \leq 1/m$$

Suppose that there is a sequence of functions $g_k \in \mathcal{C}$ such that if $d(s(n), s(0)) \leq k$ then $\bar{f}(m,n) \leq g_k(m)$.

Then there is a $\mathcal{C}$-point x such that $x = f(x)$.

Proof. The proof consists of several steps.

Step 1. The function $h : \mathbf{N} \to \mathbf{N}$ is defined by the induction below:

$$h(0) = h(1) = 0$$
$$h(2n) = h(2n+1) = \bar{f}(2^i, h(n)) \quad (i \in \mathbf{N},\ 2^i \leq n < 2^{i+1})$$

It holds that $h(2^n) = h(2^n + m)$ if $0 \leq m < 2^n$.

Step 2. Put $x_n = s(h(2^n))$ and $L = d(x_0, f(x_0))$. Then the followings hold.

$$d(x_n, f^n(x_0)) \leq 1/2^{n-1} + \alpha/2^{n-2} + \cdots + \alpha^{n-2}/2 + \alpha^{n-1}$$

$$\begin{cases} = \dfrac{\alpha^n - 2^{-n}}{\alpha - 1/2} & (\alpha \neq 1/2) \\ = n2^{1-n} & (\alpha = 1/2) \end{cases}$$

$$d(f^n(x_0), f^{n'}(x_0)) \leq \alpha^n L + \alpha^{n+1} L + \cdots + \alpha^{n'-2} L + \alpha^{n'-1} L$$
$$= (\alpha^n - \alpha^{n'})L/(1-\alpha) \quad (n \leq n')$$

Step 3. Put k, k' be integers such that

$$k \geq 2,\ \alpha^k \geq 1/2$$

and that

$$2^{k'} \geq (L/(1-\alpha)) + (1/(\alpha - (1/2)))$$

or

$$2^{k'} \geq 2L + 2 \ \text{ if } \alpha = 1/2.$$

Then it holds that
$$d(x_n, x_{n'}) \leq 2^{-(n/k)+k'} + 2^{-(n'/k)+k'}.$$
Step 4. For n, an integer i_n is put as $2^{i_n} \leq 2^{k'+1}n^k < 2^{i_n+1}$. Then $h(2^{i_n}) = h(2^{k'+1}n^k)$ and $n < 2^{(i_n/k)-k'}$. Therefore, $d(x_{i_n}, x_{i_{n'}}) < 1/n + 1/n'$. Thus, $\{x_{i_n}\}_n$ is a fundamental sequence.
Step 5. For each $n \in \mathbf{N}$,
$$d(x_0, x_n) \leq d(x_0, f^n(x_0)) + d(x_n, f^n(x_0)) \leq 1 + \alpha L/1-\alpha.$$
Hence for each $n \in \mathbf{N}$,
$$d(s(0), s(h(n))) \leq 1 + \alpha L/1-\alpha.$$
Put $l \geq 1 + \alpha L/1-\alpha$. Then, the value of h is upper-bounded by a $\mathcal{C}$-function g_l. Hence the recursion above is a limited recursion, that is, the function h is a $\mathcal{C}$-function.
Step 6. The sequence $\{x_{i_n}\}_n$ is a $\mathcal{C}$-fundamental sequence. Because X is $\mathcal{C}$-complete, this sequence $\{x_{i_n}\}_n$ converges at a point x. The sequence $\{f^n(x_0)\}_n$ also converges at x. Therefore x is a fixed point of f. □

Remark 3.29 For each $\mathcal{C}$-function over I or over $\mathbf{R}$, there always exists the functions $g_k \in \mathcal{C}$ in the statement above. Therefore, the fixed point theorem always holds in I and in $\mathbf{R}$.

Lemma 3.30 (Fixed Point Theorem in PR**)** *Let $X = (X, d, s)$ be a* PR*-complete* PR*-metric space, and $f \in \mathrm{PR}_{X,X}$ is a contraction. Then, there is a* PR*-point x such that $x = f(x)$.*

Proof. In the proof of previous lemma, the recursion always defines a PR-function without the evidence of the upper-bounding function. □

4 Function Space

Notation 4.1 We write $I \overset{conti}{\longrightarrow} \mathbf{R}$ for the set of continuous functions of I into $\mathbf{R}$, and regard $I \overset{conti}{\longrightarrow} \mathbf{R}$ as a metric space, with a distance $d : f, g \mapsto d(f, g) = \max_{x \in I} |g(x) - f(x)|$, which is the distance induced from sup-norm.

Definition 4.2 (Connected Segment) Let $q = ((x_0, y_0), (x_1, y_1), ..., (x_n, y_n))$ be a sequence of $n + 1$ points in $I \times \mathbf{R}$. Suppose that it satisfies the followings:
$$x_0 = 0,\ x_n = 1,\ x_i < x_{i+1} \text{ for } 0 \leq i \leq n - 1$$
Then this q determines the function $f : I \overset{conti}{\longrightarrow} \mathbf{R}$ as below:
$$f(x) = y_i + (y_{i+1} - y_i)\frac{x - x_i}{x_{i+1} - x_i} \quad (x_i \leq x \leq x_{i+1})$$
We call functions of this form *connected segments.*

Let s_I and $s_{\mathbf{R}}$ be separating functions of I and $\mathbf{R}$. Then we will define a separating function s for $I \overset{conti}{\longrightarrow} \mathbf{R}$ by using connected segments. For $n \in \mathbf{N}$, there is a unique sequence $(n_1, n_2, ..., n_l) \in \mathbf{N}^*$ such that $N = \langle n_1, n_2, ..., n_l \rangle$. For each component n_i, we assign a point $(x_i, y_i) = (s_I(\mathit{left}(n_i)), s_{\mathbf{R}}(\mathit{right}(n_i))) \in I \times \mathbf{R}$. If this sequence $((x_1, y_1), (x_2, y_2), ..., (x_l, y_l))$ satisfies the condition:
$$x_1 = 0,\ x_l = 1,\ x_i < x_{i+1} \text{ for } 1 \leq i \leq l - 1$$
Then $s(n)$ is a function which is determined by this sequence. Otherwise, $s(n)$ is a constant function such as $s(n)(x) = 0$.

Notation 4.3 Hereafter we fix the notation d for the distance of $I \overset{conti}{\longrightarrow} \mathbf{R}$ which is induced from sup-norm, and s for this enumeration of connected segments as the separating function of $I \overset{conti}{\longrightarrow} \mathbf{R}$.

Remark 4.4 There are P-functions $\bar{s} : \mathbf{N}^3 \to \mathbf{N}$ and $\tilde{s} : \mathbf{N}^2 \to \mathbf{N}$ such that
$\forall i, m, n.\ m \neq 0 \Rightarrow |s(i)(s_I(n)) - s_{\mathbf{R}}(\bar{s}(i, n, m))| \leq 1/m$
$\forall i, m \in \mathbf{N}. \forall x, y \in I.\ m \neq 0,\ |y - x| \leq \tilde{s}(i, m) \Rightarrow |s(i)(y) - s(i)(x)| \leq 1/m$
Thus, $s(i) \in \mathrm{P}_{I,\mathbf{R}}$.

Remark 4.5 There are functions $\bar{d} : \mathrm{P}_{\mathbf{N}^3,\mathbf{N}}$ such that
$\bar{d}(m, n, n') - 1 \leq m \cdot d(s(n), s(n')) \leq \bar{d}(m, n, n') + 1$
Thus, $(I \overset{conti}{\longrightarrow} \mathbf{R}, d, s)$ is a P-metric space.

Notation 4.6 We write $I \overset{conti}{\longrightarrow} \mathbf{R}$ for a numbered metric space $(I \overset{conti}{\longrightarrow} \mathbf{R}, d, s)$.

Proposition 4.7 *If $\mathcal{C} = \mathrm{E}^3$ or $\mathcal{C} = \mathrm{PR}$, then $\mathcal{C}_{I,\mathbf{R}} = \mathcal{C}(I \overset{conti}{\longrightarrow} \mathbf{R})$. Thus, the $\mathcal{C}$-metric space $(\mathcal{C}_{I,\mathbf{R}}, d, s)$ is $\mathcal{C}$-complete and $\mathcal{C}$-separable.*

Proof. Easy. □

Proposition 4.8 $\mathrm{P}(I \overset{conti}{\longrightarrow} \mathbf{R}) \subset \mathrm{P}_{I,\mathbf{R}}$ *and* $\mathrm{P}(I \overset{conti}{\longrightarrow} \mathbf{R}) \neq \mathrm{P}_{I,\mathbf{R}}$. *Thus, the* P*-metric space* $\mathrm{P}_{I,\mathbf{R}}$ *is* P*-complete but not* P*-separable.*

Proof. In order to make a connected segments as an approximation of a curve with error $\leq 1/n$, we have to make the connected segment of size $O(\sqrt{n})$ in general. On the other hand, a P-fundamental sequence has a connected segment of size $O((\log n)^k)$ with error $\leq 1/n$. The size $O(\sqrt{n})$ is greater than $O((\log n)^k)$. □

Remark 4.9 The previous lemma states for the special separating function which is defined by the connective segments. We have a further conjecture as below. Indeed, we have not proved this conjecture yet. We cannot regard $\mathrm{P}_{I,\mathbf{R}}$ as P-separable metric space before we find out a separating function which is dense in $\mathrm{P}_{I,\mathbf{R}}$.

Conjecture 4.10 No separating function $s : \mathbf{N} \to (I \overset{conti}{\longrightarrow} \mathbf{R})$ is P-dense in a metric space $(\mathrm{P}_{I,\mathbf{R}}, d)$, where d is the distance defined above.

Notation 4.11 The operation T over $I \overset{conti}{\longrightarrow} \mathbf{R}$ is defined as follows:

$$T(f)(x) = \begin{cases} \dfrac{f(2x)}{2} + x & (0 \leq x \leq 1/2) \\ \dfrac{f(2 - 2x)}{2} + 1 - x & (1/2 \leq x \leq 1) \end{cases}$$

This operation T is a contraction with magnification $1/2$, and the fixed point of T is called Takagi function.

Proposition 4.12 *There is a* P*-function* f *such that* $T(s(n)) = s(f(n))$*, where* $s(\ )$ *is the enumeration of the connected segments.*

Proof. Easy. □

Lemma 4.13 *Takagi function is* E^3*-computable. Hence, it is* PR*-computable.*

Proof. Because $f \in \mathrm{P}$, the iterating function $(m, n) \mapsto f^m(n)$ belongs to E^3. Then, so does the function $(2^m, n) \mapsto f^m(n)$. Therefore, by the fixed point theorem (Theorem 1), Takagi function is in $\mathrm{E}^3(I \xrightarrow{conti} \mathbf{R})$. □

Remark 4.14 It is easy to observe that Takagi function $t(\ )$ is in $\mathrm{P}_{I,\mathbf{R}}$. We will show this fact as the consequence of fixed point theorem. Unfortunately, the operator T does not satisfies the hypothesis of the fixed point theorem (Theorem 1), because there are no P-functions g_k in the theorem. Actually, Takagi function $t(\ )$ is not in $\mathrm{P}(I \xrightarrow{conti} \mathbf{R})$. Therefore we will show that there is a functional

$$(\bar{T}, \tilde{T}) : (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N}) \longrightarrow (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N})$$

such that $(\bar{T}, \tilde{T})^n(\bar{f}, \tilde{f})$ represents a function t_n, and that t_n converges into $t(\ )$ as $n \to \infty$.

Definition 4.15 (Functional) A function over functions is called a *functional*. For a typical example, $F : (\mathbf{N} \to \mathbf{N}) \to \mathbf{N} \to \mathbf{N}$ is a functional. The computability of functionals is defined by Odifreddi [Odifreddi]. We also regard functions with more general arities as functionals. For example, $F : (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N}) \to (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N})$ is also a functional.

Definition 4.16 (Representation of Functionals) Let X_1, X_2, X_3 and X_4 be numbered metric spaces. Suppose that X_1 and X_3 are bounded. Let F be a function in $(X_1 \xrightarrow{conti} X_2) \to (X_3 \xrightarrow{conti} X_4)$. Let $\hat{F}$ be a functional in $(\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N}) \to (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N})$. Then, the functional $\hat{F}$ *represents* F iff for any $\bar{f} : \mathbf{N}^2 \to \mathbf{N}, \tilde{f} : \mathbf{N} \to \mathbf{N}$ and $f : X_1 \xrightarrow{conti} X_2$, if $(\bar{f}, \tilde{f}) \vdash f$ then $\hat{F}(\bar{f}, \tilde{f}) \vdash F(f)$. We write $\hat{F} \vdash F$ when $\hat{F}$ represents F.

Remark 4.17 If the domain X_1 is not bounded, then the representation (g, h) of a function $f : X_1 \to X_2$ has the arity such as $(g, h) : (\mathbf{N}^2 \to \mathbf{N})^2$ in general. This holds also for X_3. Therefore, if neither X_1 nor X_3 is bounded, we have to consider the representation $\hat{F}$ of arity $(\mathbf{N}^2 \to \mathbf{N})^2 \to (\mathbf{N}^2 \to \mathbf{N})^2$. In this paper we discuss only the case that X_1 and X_3 are bounded. Therefore we deal with only the representations of arity $(\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N}) \to (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N})$.

Conjecture 4.18 (Existence) Let X_1, X_2, X_3 and X_4 be numbered metric spaces. Suppose that X_1 and X_3 are compact. Regard $X_1 \xrightarrow{conti} X_2$ and $X_3 \xrightarrow{conti} X_4$ as metric spaces with the following distance functions:

$$\text{For } f, g \in X_1 \xrightarrow{conti} X_2, \quad d(f, g) = \max_{x \in X_1} d_{X_2}(f(x), g(x))$$

$$\text{For } f, g \in X_3 \overset{conti}{\longrightarrow} X_4,\ \ d(f,g) = \max_{x \in X_3} d_{X_4}(f(x), g(x))$$

Then, each uniformly continuous function in $(X_1 \overset{conti}{\longrightarrow} X_2) \to (X_3 \overset{conti}{\longrightarrow} X_4)$ has the representation $\hat{F} = (\bar{F}, \tilde{F})$, and both functionals $\bar{F} : (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N}) \to \mathbf{N}^2 \to \mathbf{N}$ and $\tilde{F} : (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N}) \to \mathbf{N} \to \mathbf{N}$ are continuous.

Lemma 4.19 (Uniqueness) *Let X_1, X_2, X_3 and X_4 be numbered metric spaces. Suppose that X_1 is compact. If $\hat{F} \vdash F$ and $\hat{F} \vdash F'$, then $F = F'$*

Proof. Each f in the domain of F and F' has the representation $(\bar{f}, \tilde{f})$ such that $(\bar{f}, \tilde{f}) \vdash f$, by Lemma 3.24. Thus $\hat{F}(\bar{f}, \tilde{f}) \vdash F(f)$ and $\hat{F}(\bar{f}, \tilde{f}) \vdash F'(f)$. Therefore $F(f) = F'(f)$ because of Lemma 3.25. □

Theorem 2. (Fixed Point Theorem for Functionals) *Let $X = (|X|, d_X, s_X)$ and $Y = (|Y|, d_Y, s_Y)$ be computable metric spaces. Suppose that X is compact and Y is $\mathcal{C}$-complete. We regard $X \overset{conti}{\longrightarrow} Y$ as a metric space with the distance function $f, g \mapsto \max_{x \in |X|} d_Y(f(x), g(x))$. Let F be contraction of $X \overset{conti}{\longrightarrow} Y$ into $X \overset{conti}{\longrightarrow} Y$. Let $\hat{F}$ be a functional such that $\hat{F} \vdash F$.*

Suppose that the following functions f' and f'' are $\mathcal{C}$-functions for some $\bar{g} \in \mathcal{C}_{\mathbf{N}^2,\mathbf{N}}$, $\tilde{g} \in \mathcal{C}_{\mathbf{N},\mathbf{N}}$ and $g : X \overset{conti}{\longrightarrow} Y$ such that $(\bar{g}, \tilde{g}) \vdash g$.

$$f'(2^k, m, n) = \bar{f}_k(m, n),\ f''(2^k, m) = \tilde{f}_k(m) \text{ where } (\bar{f}_k, \tilde{f}_k) = \hat{F}^k(\bar{g}, \tilde{g})$$

Then, there is a function $f \in \mathcal{C}_{X,Y}$ such that f is a fixed point of F.

Proof. It is obvious that the fixed point of F is represented by the functions $(\bar{f}, \tilde{f})$ which are defined as:

$$\bar{f}(m, n) = f'(2^{k+1}, 2m, n) \qquad \tilde{f}(m) = f'(2^{k+1}, 2m)$$

where $2^k < m \leq 2^{k+1}$. □

Notation 4.20 We define functions in class P as follows:

$$h : \mathbf{N}^2 \to \mathbf{N};\quad |2s_I(n) - s_I(h(m,n))| < 1/m\ (m \geq 1, s_I(n) \leq 1/2)$$
$$|2 - 2s_I(n) - s_I(h(m,n))| < 1/m\ (m \geq 1, s_I(n) \geq 1/2)$$
$$g : \mathbf{N}^3 \to \mathbf{N};\quad \left| \frac{s_{\mathbf{R}}(n) + s_I(n')}{2} - s_{\mathbf{R}}(g(m,n,n')) \right| < 1/m\ (m \geq 1)$$

and that $g(m, n, n') = O(m^2), h(m, n) = O(m^2)$. This estimation of growth rate is possible, for the definition of s_I and $s_{\mathbf{R}}$.

Notation 4.21 we define functionals $\bar{T}$, $\tilde{T}$, and $(\bar{T}, \tilde{T})$ as follows:

$$\bar{T} : (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N}) \to \mathbf{N}^2 \to \mathbf{N};$$
$$\bar{T}(\bar{f}, \tilde{f})(m, n) = g(4m, \bar{f}(4m, h(\tilde{f}(4m), n)), h(4m, n))$$
$$\tilde{T} : (\mathbf{N} \to \mathbf{N}) \to \mathbf{N} \to \mathbf{N};$$
$$\tilde{T}(\tilde{f})(m) = \max(\tilde{f}(\lceil 2m/3 \rceil), 4m)$$

where $\lceil - \rceil$ is the ceiling function such that $\lceil x \rceil$ is a integer and $\lceil x \rceil - 1 < x \leq \lceil x \rceil$.

$$(\bar{T}, \tilde{T}) : (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N}) \to (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N});$$
$$(\bar{T}, \tilde{T})(\bar{f}, \tilde{f}) = \left(\bar{T}(\bar{f}, \tilde{f}), \tilde{T}(\tilde{f}) \right)$$

Proposition 4.22 $(\bar{T}, \tilde{T}) \vdash T$

Notation 4.23 We write $\hat{0}$ for the constant functions with the constant value 0 for any arities.

Proposition 4.24 *The following functions t' and t'' belong to $\mathrm{P}_{\mathbf{N}^2,\mathbf{N}}$ and to $\mathrm{P}_{\mathbf{N},\mathbf{N}}$.*

$$t'(2^k, m, n) = \bar{f}_k(m, n),\ t''(2^k, m, n) = \tilde{f}_k(m)$$

where $(\bar{f}_k, \tilde{f}_k) = (\bar{T}, \tilde{T})^k(\hat{0}, \hat{0})$

Proposition 4.25 *Easy.*

Corollary 4.26 *Takagi function is in $\mathrm{P}_{I,\mathbf{R}}$.*

Proof. By Theorem 2, Proposition 4.22 and Proposition 4.24. □

5 Discussion

5.1 Related Works

There have been several works on computability of objects in some metric spaces. There are two ways to define the computability over a metric spaces.

One is to use computable structure, which is given by Pour-El and Richard [Pour-El&Richards]. For a metric space $X = (|X|, d)$, a computable structure S is a set of sequences of points in $|X|$ which satisfies some properties. One of the properties is that for each computable function $f : \mathbf{N} \to \mathbf{N}$ and a sequence $\{x_n\}_n \in S$, the new sequence $\{x_{f(n)}\}_n$ also belongs to S. Then we will regard that a objects is computable iff the objects is defined by elements of S. For example, a point $x \in |X|$ is computable iff there are a sequence $\{x_n\}_n \in S$ such that the sequence $\{x_n\}$ converges into x with the computable modulus $f : \mathbf{N} \to \mathbf{N}$, that is, it holds that $d(x, x_n) \leq 1/m$ if $n \geq f(m)$ and $m \geq 1$. When we use a computable structure S for a metric space $X = (|X|, d)$, we have to guarantee in advance that all the elements in S are computable.

Another way to define the computability is to use the separating function, which is given by Weihrauch [Weihrauch93]. A triple $X = (|X|, d, s)$ is a computable metric space iff $(|X|, d)$ is a metric space, and $d : |X|^2 \to \mathbf{R}$ is computable with respect to $s : \mathbf{N} \to |X|$. We can guarantee the computability of a sequence $\{x_n \in |X|\}_n$ by using this separating function s, as far as s is adequate to computability. Thus, the sequence $\{x_n\}_n$ is computable iff there is a computable function $f : \mathbf{N}^2 \to \mathbf{N}$ and $s(f(m, n))$ converges into x_n as $m \to \infty$ with a computable modulus.

Our discussion in Section 4 shows that we can discuss computability without the separating function. Therefore, our work suggests that there are many other ways to discuss the computability of the elements of computable structure.

All the works above discussed computability without the limitation of complexity. Our work intends to discuss polynomial time computability over metric space.

5.2 Type-1 Objects and Type-2 Objects

Functions over natural numbers are called *Type-1 objects*. A function in $\mathbf{N}^2 \to \mathbf{N}$ or in $\mathbf{N}^3 \to \mathbf{N}$ is also called a Type-1 object. A pairing of Type-1 objects is also called a Type-1 object, such as $(\bar{f}, \tilde{f}) : (\mathbf{N}^2 \to \mathbf{N}) \times (\mathbf{N} \to \mathbf{N})$. Functionals, which is functions over functions, are called Type-2 objects. For example, $F : (\mathbf{N} \to \mathbf{N}) \to \mathbf{N} \to \mathbf{N}$ is a Type-2 object. The machines which represents Type-2 objects are called Type-2 machines, The word of Type-2 machine appears in the works by Weihrauch [Weihrauch00].

A computable point in a computable metric space is represented by a fundamental sequence which converges into the point. Hence, computable points in a computable metric space are represented by Type-1 objects.

The representation of a function f over a metric space plays the role of a function over Type-1 objects, namely, the representation of a point x is mapped to the representation of $f(x)$. Therefore, it is natural that a Type-2 function represents this function f. Although, computable functions over computable metric spaces are represented by Type-1 object. That is because the domain of the function is separable, so we can restrict the arguments into only the points in the separating set.

On the other hand, we should represent the functions over the function space $\mathrm{P}_{I,\mathbf{R}}$. by functionals, which is Type-2 object. That is because we does not have a separating function for the function space $\mathrm{P}_{I,\mathbf{R}}$.

Acknowledgements

The author would like to thank Yasugi Mariko, Kamo Hiroyasu, and anonymous referees for discussions and comments. This work was presented at the Forth Workshop on Computability and Complexity in Analysis. The author also thanks the participants of the workshop for the comments.

References

Brattka. Brattka, Vasco: Computable invariance. *Theoret. Comput. Sci.* 210 (1999) 3–20.

Grzegorczyk. Grzegorczyk, Andrzej: Some classes of recursive functions. *Rozprawy Math.*, 4 (1953) 1–46.

Odifreddi. Odifreddi, Piergiorgio G.: *Classical recursion theory.* Studies in logic and the foundations of mathematics, vol. 125, North-Holland, 1989.

Pour-El&Richards. Pour-El, M.B., and Richards, J. I.: *Computability in Analysis and Physics.* Springer-Verlag, 1989.

Weihrauch93. Weihrauch, Klaus: Computability on computable metric spaces. *Theoret. Comput. Sci.* 113 (1993) 191–210.

Weihrauch00. Weihrauch, Klaus: Computable Analysis, Springer-Verlag, 2000.

YHK. Yamaguti, M., Hata, M., and Kigami, J.: *Hurakutaru no sûri* (in Japanese). Iwanami kôza ôyô sûgaku, Iwanami shoten, 1993.

Computational Dimension of Topological Spaces

Hideki Tsuiki

Division of Mathematics
Faculty of Integrated Human Studies, Kyoto University
tsuiki@i.h.kyoto-u.ac.jp

Abstract. When a topological space X can be embedded into the space $\Sigma^\omega_{\perp,n}$ of $n\perp$-sequences of Σ, then we can define the corresponding computational notion over X because a machine with $n+1$ heads on each tape can input/output sequences in $\Sigma^\omega_{\perp,n}$. This means that the least number n such that X can be topologically embedded into $\Sigma^\omega_{\perp,n}$ serves as a degree of complexity of the space. We prove that this number, which we call the computational dimension of the space, is equal to the topological dimension for separable metric spaces. First, we show that the weak inductive dimension of $\Sigma^\omega_{\perp,n}$ is n, and thus the computational dimension is at least as large as the weak inductive dimension for all spaces. Then, we show that the Nöbeling's universal n-dimensional space can be embedded into $\Sigma^\omega_{\perp,n}$ and thus the computational dimension is at most as large as the weak inductive dimension for separable metric spaces. As a corollary, the 2-dimensional Euclidean space $\mathbb{R}^2$ can be embedded in $\{0,1\}^\omega_{\perp,2}$ but not in $\Sigma^\omega_{\perp,1}$ for any character set Σ, and infinite dimensional spaces like the set of closed/open/compact subsets of $\mathbb{R}^m$ and the set of continuous functions from $\mathbb{R}^l$ to $\mathbb{R}^m$ can be embedded in $\Sigma^\omega_\perp$ but not in $\Sigma^\omega_{\perp,n}$ for any n.

1 Introduction

In order to perform computation over the set of reals, we need to represent them as (infinite) sequences of characters. However, it is known that there is no one-to-one representation, or equivalently, no embedding of real numbers into Σ^ω which induces reasonable notion of computation over reals, and thus redundant representations are commonly used [Wei00][BH00]. In a previous paper [Tsu01b], the author used, instead of Σ^ω, the set $\Sigma^\omega_{\perp,1}$ of $1\perp$-sequences of Σ. Here, an $n\perp$-sequence of Σ is an infinite sequence of Σ in which at most n cells are allowed to be left undefined (denoted by $\perp$). He proposed a machine, called an IM2-machine, which input/output $1\perp$-sequences with two heads on each input/output tape, composed a topological embedding of the set of real numbers into $\Sigma^\omega_{\perp,1}$, and thus induced a notion of computation over reals, which is shown to be equivalent to the standard notion of computation over reals.

It is easy to extend this input/output mechanism of an IM2-machine to the set $\Sigma^\omega_{\perp,n}$ of $n\perp$-sequences of Σ by putting $n+1$ heads on each input/output tape, and therefore, we can obtain a computational notion over a topological space X when X can be embedded into $\Sigma^\omega_{\perp,n}$. Thus, we define the least number n such that a space X can be embedded into $\Sigma^\omega_{\perp,n}$ as the computational dimension

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 323–335, 2001.

of X. The computational dimension is a degree of computational complexity of the space in that it is equal to the number of extra heads required to define computation over the space by an IM2-machine.

The main theorem of this paper is that the computational dimension and the usual topological dimension coincide for separable metric spaces. First, we show that the weak inductive dimension of $\Sigma^\omega_{\perp,n}$ is n, and therefore the computational dimension is at least as large as the weak inductive dimension for all spaces. Then, we show that the Nöbeling's universal n-dimensional space can be embedded into $\Sigma^\omega_{\perp,n}$ and thus the computational dimension is at most as large as the weak inductive dimension for separable metric spaces.

From this theorem, the 2-dimensional Euclidean space can be embedded into $\Sigma^\omega_{\perp,2}$, but not in $\Sigma^\omega_{\perp,1}$ for any character set Σ, and infinite dimensional spaces like the spaces of closed/open/compact subsets of $\mathbb{R}^l$ and the space of continuous functions from $\mathbb{R}^l$ to $\mathbb{R}^m$ cannot be embedded in $\Sigma^\omega_{\perp,n}$ for any n.

In the next section, we review the notion of Gray code embedding and IM2-machines following [Tsu01b]. Then, we introduce the computational dimension in Section 3. In Section 4, we show that the weak inductive dimension of $\Sigma^\omega_{\perp,n}$ is n. In Section 5, we study the embedding of the 2-dimensional Euclidean space to $\Sigma^\omega_{\perp,2}$, and in Section 6, we prove that the computational dimension and the weak inductive dimension coincide for separable metric spaces. In Section 7, we study the embeddings of infinite dimensional spaces in $\Sigma^\omega_\perp$.

2 Gray Code Embedding and Computability by IM2-Machines

We write $\Sigma^\omega_\perp$ for the set of infinite sequences of Σ in which undefined cells ($\perp$) are allowed to exist. That is, $\Sigma^\omega_\perp$ is the set of infinite sequences of $\Sigma \cup \{\perp\}$. We write $\Sigma^\omega_{\perp,n}$ for the set of infinite sequences of Σ in which at most n undefined cells are allowed to exist, Gray code embedding G (Definition 1 below) is an embedding of the unit open interval $\mathcal{I} = (0,1)$ (or the whole real line $\mathbb{R}$) to the set $\{0,1\}^\omega_{\perp,1}$. It is based on the Gray code expansion, which is another expansion of real numbers.

Figure 1 shows the usual binary expansion and the Gray code expansion of the unit open interval. Here, a horizontal line means that the corresponding bit has value 1 on the line and value 0 otherwise. In this way, Gray code expansion of (0,1) is composed from that of $(0, 1/2)$ by taking the mirroring image on $(1/2, 1)$ with the first bit on. As is the case for the usual binary expansion, we have two expansions to dyadic numbers. Here, a dyadic number is a rational number of the form $m \times 2^{-n}$ for integers m and n. For example, we have two Gray code expansions 111000... and 101000... for $3/4$, corresponding to the two binary expansions 11000... and 10111.... However, the two expansions are different only at one digit (in this case the second). This means that the second digit does not contribute to the fact that this number is $3/4$. Therefore, by defining the value of such a digit as $\perp$, we define the Gray code embedding G of $\mathcal{I}$ to $\{0,1\}^\omega_{\perp,1}$ as follows.

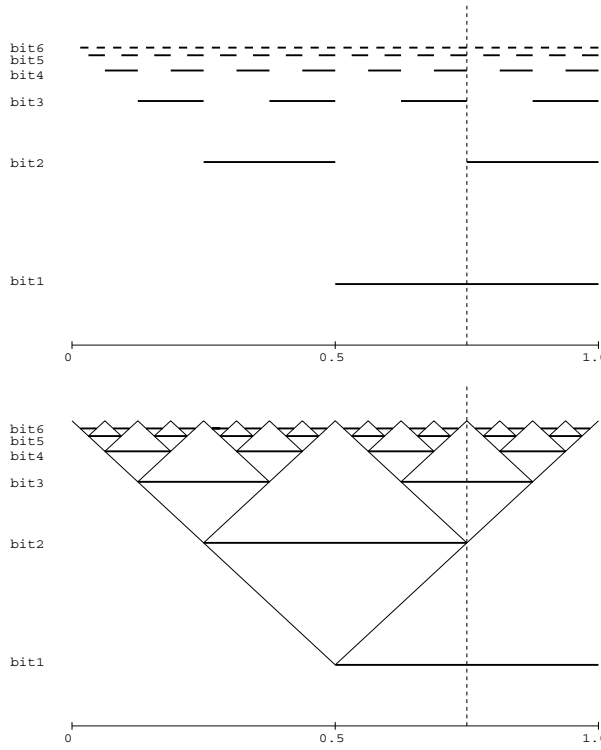


Fig. 1. Binary Expansion and Gray code Expansion of real numbers.

Definition 1. The Gray code embedding G is a function from $\mathcal{I}$ to $\{0,1\}^{\omega}_{\perp,1}$ defined coinductively as follows. Let $G(x) = a_0 a_1 \ldots$. The first character a_0 is $1, \perp$, or 0, according as x is bigger than, equal to, or less than $1/2$. The rest $a_1 a_2 \ldots$ is defined to be $G(f(x))$ where

$$f(x) = \begin{cases} 2x & (x \leq 1/2) \\ 2(1-x) & (x > 1/2) \end{cases} .$$

G can be extended to a function from $(-1,1)$ to $\{0,1\}^{\omega}_{\perp,1}$ by adding the sign bit as the first bit, and to the whole real line by composing it with some computable embedding of $\mathbb{R}$ into $(-1,1)$ such as the function $f(x) = 2{*}arctan(x)/\pi$.

Note that G comes to be an injective function to $\{0,1\}^{\omega}_{\perp,1}$. Moreover, we can show that G maps $\mathcal{I}$ homeomorphically into $\{0,1\}^{\omega}_{\perp,1}$, and therefore, G is actually a topological embedding. Here, the topology on $\Sigma^{\omega}_{\perp,1}$ is the subspace topology of the Scott topology on the cpo $\Sigma^{\omega}_{\perp}$ (see Section 4).

Now, we define a machine which inputs/outputs sequences in $\Sigma^{\omega}_{\perp,1}$. First, we consider outputs. We consider that our machine calculates a real number x $(0 < x < 1)$ by producing shrinking intervals $(a_1, b_1), (a_2, b_2), \ldots$ infinitely so that

their intersection is $\{x\}$, and output $G(x)$ on a tape based on this approximation information. We consider that the tape is filled with $\perp$ at the beginning. When we know that $x < 1/2$ or $x > 1/2$, we can write 0 or 1 as the first digit, respectively. However, when $x = 1/2$, neither will happen and therefore we cannot fill the first cell. However, if we know that $1/4 < x < 3/4$, we can fill the second cell with 1. After that, we can fill the first cell with 0 or 1 when we know that $1/4 < x < 1/2$ or $1/2 < x < 3/4$, respectively, and the third cell with 0 when we know that $3/8 < x < 5/8$. In particular, when $x = 1/2$, the first cell is unfilled eternally, and the sequence $1000\ldots$ is output from the second cell. This kind of output can be expressed naturally if we consider two heads $\mathrm{H}_1(O)$ and $\mathrm{H}_2(O)$ on the output tape O. They are placed at the first and the second cell at the beginning, and move automatically when a character is output: after an output from $\mathrm{H}_1(O)$, $\mathrm{H}_1(O)$ is moved to the place of $\mathrm{H}_2(O)$ and $\mathrm{H}_2(O)$ is moved to the next cell, and after an output from $\mathrm{H}_2(O)$, $\mathrm{H}_2(O)$ is moved to the next cell. In this way, $\mathrm{H}_1(O)$ and $\mathrm{H}_2(O)$ are always located at the first and the second unfilled cells, respectively.

Next, we consider an input of a sequence in $\Sigma^\omega_{\perp,1}$. When we have only one head on an input tape, the machine stucks when the head comes to the $\perp$ cell because it will never be filled. Therefore, our machine has two heads $\mathrm{H}_1(I_i)$ and $\mathrm{H}_2(I_i)$ on each input tape I_i, which move the same way as the heads of an output tape, and the machine proceeds depending on an input from one of them. This means that when both cells under $\mathrm{H}_1(I_i)$ and $\mathrm{H}_2(I_i)$ are filled, we have two applicable rules. Thus, our machine has many computational paths to an input and all the paths should produce valid results. This property is called indeterminism. In this way, we define an IM2-machine (Indeterministic Multihead Type-2 machine), which inputs/outputs sequences in $\Sigma^\omega_{\perp,1}$. See [Tsu01b] for the detailed definition of the machine.

This notion of an IM2-machine can easily be generalized to a machine with $n+1$ heads on each input/output tape. The heads $H_1(T), H_2(T), \ldots, H_{n+1}(T)$ of a tape T move as follows: when $H_i(T)$ $(1 \leq i \leq n+1)$ is used, $H_j(T)$ $(i \leq j \leq n)$ move to the position of $H_{j+1}(T)$ and H_{n+1} moves to the next position. Note that an $n+1$-head machine can manipulate sequences in $\Sigma^\omega_{\perp,n}$. Therefore, we define an IM2-machine of type $(\Sigma^\omega_{\perp,n_1}, \ldots, \Sigma^\omega_{\perp,n_k}, \Sigma^\omega_{\perp,n_0})$ as a machine which has k input tapes with $n_i + 1$ heads $(i = 1, \ldots, k)$ and one output tape with $n_0 + 1$ heads.

Since an IM2-machine has indeterministic behavior, it defines a partial multi-valued function $F :\subseteq \Sigma^\omega_{\perp,n_1} \times \ldots \times \Sigma^\omega_{\perp,n_k} \rightrightarrows \Sigma^\omega_{\perp,n_0}$, which is a subset of $\Sigma^\omega_{\perp,n_1} \times \ldots \times \Sigma^\omega_{\perp,n_k} \times \Sigma^\omega_{\perp,n_0}$ considered as a partial function from $\Sigma^\omega_{\perp,n_1} \times \ldots \times \Sigma^\omega_{\perp,n_k}$ to (the power set of $\Sigma^\omega_{\perp,n_0}$) $- \emptyset$.

Definition 2. An IM2-machine M with k inputs realizes a partial multi-valued function $F :\subseteq \Sigma^\omega_{\perp,n_1} \times \ldots \times \Sigma^\omega_{\perp,n_k} \rightrightarrows \Sigma^\omega_{\perp,n_0}$ if all the computational paths M have with the input tapes filled with $(p_1, \ldots, p_k) \in dom(F)$ produce infinite outputs, and the set of outputs forms a subset of $F(p_1, \ldots, p_k)$. We say that F is IM2-computable when it is realized by an IM2-machine.

Definition 3. Let H_i be embeddings of X_i to $\Sigma^\omega_{\perp,n_i}$ $(i = 0, 1, \ldots, k)$. A multi-valued function $F :\subseteq X_1 \times \ldots \times X_k \rightrightarrows X_0$ is realized by an IM2-machine M if $H_0 \circ F \circ (H_1^{-1}, \ldots, H_k^{-1})$ is realized by M. In this case, we say that F is $(H_1, H_2, \ldots, H_k, H_0)_\perp$-computable.

When H_i are the Gray code embedding G of $\mathbb{R}$ in $\{0,1\}^\omega_{\perp,1}$, we say that $F :\subseteq \mathbb{R}^k \to \mathbb{R}$ is Gray code computable. In this definition, we add the suffix $_\perp$ to the type of the computability to distinguish it from the usual representation-based computability notion by a Type-2 machine [Wei00].

Definition 4. A partial function is $(H_1, H_2, \ldots, H_k, H_0)_\perp$-computable if it is $(H_1, H_2, \ldots, H_k, H_0)_\perp$-computable as a multi-valued function.

By generalizing the proofs in [Tsu01b], one has the followings:

Theorem 1. *If a partial function is $(H_1, H_2, \ldots, H_k, H_0)_\perp$-computable, then it is continuous.*

Theorem 2. *A multi-valued function $F :\subseteq \mathcal{I}^k \to \mathcal{I}$ is Gray-code-computable iff it is $((\rho)^k, \rho)$-computable in the sense of [Wei00]. Here, ρ is the signed digit representation or some equivalent ones.*

In [Tsu01b], the author gave some basic algorithms by IM2-machines like addition with respect to the Gray code embedding. In [Tsu01a], he showed that the behavior of an IM2-machine can be naturally expressed in a parallel logic programming language GHC.

3 The Computational Dimension of a Topological Space

Now, we study whether topological spaces other than $\mathbb{R}$ have similar embeddings in $\Sigma^\omega_{\perp,n}$. As we have shown, if a space X can be embedded into $\Sigma^\omega_{\perp,n}$, then we can define computational notion on X based on $n+1$-head IM2-machines. Thus, the least number n such that X can be embedded into $\Sigma^\omega_{\perp,n}$ for some Σ has a meaning as a computational complexity of the space.

Definition 5. The computational dimension of a space X is the least number n such that X can be embedded into $\Sigma^\omega_{\perp,n}$. If X can be embedded into $\Sigma^\omega_\perp$ and X cannot be embedded into $\Sigma^\omega_{\perp,n}$ for every n, then we define the computational dimension of X as ∞.

We show that this computational dimension coincides with the usual topological dimension for separable metric spaces. There are several definitions of the topological dimension of a space: the covering dimension, the strong inductive dimension, and the weak inductive dimension [HW48,Nag65]. It is known that these three dimensions are equivalent and have many good properties for a separable metric space. However, we need to develop dimension theory also to T_0 spaces and only the weak inductive dimension has the properties we need for such a general space.

We write $B_P(O)$ for the boundary of O in a topological space P, or $B(O)$ when it is not ambiguous.

Definition 6. The weak inductive dimension ind of a topological space X is defined to be
i) ind $X = -1$ if $X = \emptyset$,
ii) ind $X \leq n$ if for every neighborhood U of a point $p \in X$ there exists an open set V such that $x \in V \subset U$ and ind $B(V) \leq n-1$.
If ind $X \leq n$ and ind $X \not\leq n-1$, then we define ind $X = n$. If ind $X \not\leq n$ for every n, then ind $X = \infty$.

The following proposition is straightforward and we use this in calculating the dimension.

Proposition 1. *If X has an open base $\mathcal{O}$ such that every element $U \in \mathcal{O}$ satisfies* ind $B(U) \leq n-1$, *then* ind $X \leq n$.

Lemma 1. *Let P be a subspace of a topological space X and $O \subset X$. Then, $B_P(O \cap P) \subset B_X(O) \cap P$.*

For a counter example to $B_P(O \cap P) = B_X(O) \cap P$, consider the Scott topology of three point poset $a < b > c$ for X, $\{a, c\}$ for P, and $\{b, c\}$ for O.

Proposition 2 (Heredity of ind **).** *If* ind $X \leq n$ *and P is a subspace of X, then* ind $P \leq n$.

Proof. By induction on n. It is trivial for the case $n = -1$. Assume it for $n-1$. Since ind $X \leq n$, for all $x \in P$ and $O \ni x$, there exists $x \in O' \subset O$ such that ind $B(O') \leq n-1$. Since $B_P(O' \cap P) \subset B(O')$, by induction hypothesis, we have ind $B_P(O' \cap P) \leq n-1$.

This heredity property does not hold for T_0 spaces when we consider the covering dimension and the strong inductive dimension. See Appendix of [HW48] for the detail. Since this heredity holds and that dimension is preserved by homeomorphisms, we have the following:

Proposition 3. *If* ind $X >$ ind Y, *then there is no topological embedding of X in Y.*

4 The Weak Inductive Dimension of $n\perp$-Sequence Spaces

We write $\Sigma^\omega_{\perp,n,m}$ for the subspace $\Sigma^\omega_{\perp,n} - \Sigma^\omega_{\perp,m-1}$ of $\Sigma^\omega_\perp$ $(n \geq m)$, and $\Sigma^\omega_{\perp,-,m}$ for the subspace $\Sigma^\omega_\perp - \Sigma^\omega_{\perp,m-1}$ of $\Sigma^\omega_\perp$. When $\alpha \in \Sigma^\omega_\perp$, we write $\alpha[j]$ for the j-th component of α, and we write $\alpha|_k$ for the compact element of $\Sigma^\omega_\perp$ such that $\alpha|_k[n] = \alpha[n]$ for $n \leq k$ and $\alpha[n] = \perp$ for $n > k$.

The Scott topology of a complete partial order (cpo) P is defined so that a subset O is open iff it is upward closed and for each directed subset S of P with $\sqcup S \in O$, $s \in O$ for some $s \in S$. We say that an element x of P is compact if for each directed subset S of P with $x \leq \sqcup S$, $x \leq s$ for some $s \in S$. In the cpo $\Sigma^\omega_\perp$, d is a compact element iff $d[k] \neq \perp$ for finitely many k. $\Sigma^\omega_\perp$ has the base $\{d\uparrow \mid d \in P \text{ is compact}\}$. Here, we write $d\uparrow$ for the subset $\{x \in P | d \leq x\}$ of P.

Definition 7. The length of a poset P is the maximal length n of a strictly increasing chain $a_0 < a_1 < \ldots < a_n$ in P. If there is an arbitrary long chain in P, then we define that the length of P is infinite.

Proposition 4. *1) The length of Σ^ω is 0.*
2) The length of $\Sigma^\omega_{\perp,n}$ is n.
3) The length of $\Sigma^\omega_\perp$ is infinite.

It is easy to prove that the dimension of a cpo Q with the Scott topology is equal to the length of Q; it is a finite number n only when Q consists only of compact elements. However, when we consider the Scott topology on a cpo Q, and consider its subspace topology on a subposet P, then the length of P and the dimension of P do not coincide. For example, one can consider the image $im(G) \subset \Sigma^\omega_\perp$ of the Gray code embedding G. It has length 0 because there is no order relation among elements of $im(G)$, whereas it has dimension 1 because it is homeomorphic to $\mathcal{I}$.

It does not hold generally that the closure of an open set O is $\{x \mid x \leq y \text{ for } \exists y \in O\}$. As a counter example, consider the case $X = \{0,1\}^\omega_\perp$, $O = \cup\{d\uparrow \mid d = 0^n 1 \perp^\omega \text{ for some } n\}$. Since O includes $0^n 10^\omega$ for all n, $Cl(O)$ includes the increasing sequence $0\perp^\omega < 00\perp^\omega < 000\perp^\omega < ...$, and therefore includes its limit 0^ω. However, this property holds when O is a base element $d\uparrow$. We prove the following stronger statement:

Lemma 2. *Suppose that P is a closed subspace of $\Sigma^\omega_\perp$ and d is a compact element of P. Then, $B_P(d\uparrow \cap P) \ni \alpha$ iff $d \sqcup \alpha$ exists in P and $d \not\leq \alpha$.*

Proof. The if part is trivial. $B_P(d\uparrow \cap P) \ni \alpha$ means that all the open sets containing α intersect with $d\uparrow$ in P. Therefore, $\alpha|_k\uparrow$ intersects with $d\uparrow$, and thus $d \sqcup \alpha|_k$ is a member of P for all k. In this way, we have an increasing sequence $d \sqcup \alpha|_1 \leq d \sqcup \alpha|_2 \leq \ldots$ whose limit $d \sqcup \alpha$ exists because $\Sigma^\omega_\perp$ is a cpo. From the closedness of P, $d \sqcup \alpha$ is in P.

Proposition 5. *If $P \subset \Sigma^\omega_{\perp,-,m}$ is a closed subset of $\Sigma^\omega_\perp$ and d is a compact element of P, then $B_P(d\uparrow \cap P) \subset \Sigma^\omega_{\perp,-,m+1}$.*

Proof. If all the elements of $B_P(d\uparrow \cap P)$ have infinite number of bottom components, then we have nothing to prove. Suppose that $\alpha \in B_P(d\uparrow \cap P)$ has finite number of bottom components. Then, from Lemma 2, $d \not\leq \alpha$ and $d \sqcup \alpha$ exists in P. Since $d \sqcup \alpha$ is strictly greater than α, $d \sqcup \alpha$ has fewer bottom components than α has. At the same time, $d \sqcup \alpha$ has at least m bottom components. This means that α has at least $m+1$ bottom components and therefore $\alpha \in \Sigma^\omega_{\perp,-,m+1}$.

Lemma 3. *Suppose that P is a closed subset of $\Sigma^\omega_\perp$. $B_{P \cap \Sigma^\omega_{\perp,n}}(d\uparrow \cap P \cap \Sigma^\omega_{\perp,n}) = B_P(d\uparrow \cap P) \cap \Sigma^\omega_{\perp,n}$.*

Proof. Use Lemma 1 for $\subset$ and Lemma 2 for $\supset$.

Proposition 6. *Let $P \subset \Sigma^\omega_{\perp,-,m}$ be a closed subset of $\Sigma^\omega_\perp$ and $n \geq m$. Then,* ind $(P \cap \Sigma^\omega_{\perp,n}) \leq n - m$.

Proof. By induction on $n - m$. First, consider the case that $n = m$. We have $B_{P\cap\Sigma^\omega_{\perp,n}}(d \uparrow \cap P \cap \Sigma^\omega_{\perp,n}) = B_P(d \uparrow \cap P) \cap \Sigma^\omega_{\perp,n}$ by Lemma 3 and $B_P(d \uparrow \cap P) \cap \Sigma^\omega_{\perp,n} = \emptyset$ since $B_P(d \uparrow \cap P) \subset \Sigma^\omega_{\perp,-,m+1}$ by Proposition 5. Therefore, ind $B_{P\cap\Sigma^\omega_{\perp,n}}(d\uparrow \cap P \cap \Sigma^\omega_{\perp,n}) = -1$ for all compact element d. By Proposition 1, we have ind $(P \cap \Sigma^\omega_{\perp,n}) \leq n - m$.

Next, consider the case $n > m$. We need to show ind $B_{P\cap\Sigma^\omega_{\perp,n}}(d \uparrow \cap P \cap \Sigma^\omega_{\perp,n}) = B_P(d\uparrow \cap P) \cap \Sigma^\omega_{\perp,n} \leq n - m - 1$ for all compact element d. We have $B_P(d\uparrow \cap P) \subset \Sigma^\omega_{\perp,-,m+1}$ by Proposition 5. Since $B_P(d\uparrow \cap P)$ is closed, we have ind $B_P(d\uparrow \cap P) \leq n - (m + 1)$ by the induction hypothesis.

Theorem 3. ind $\Sigma^\omega_{\perp,n} = n$.

Proof. ind $\Sigma^\omega_{\perp,n} \leq n$ by applying Proposition 6 to the case $P = \Sigma^\omega_\perp$ and $m = 0$. For ind $\Sigma^\omega_{\perp,n} \geq n$, when $|\Sigma| \geq 2$, we use the embedding of $\mathbb{R}^n$ in $\Sigma^\omega_{\perp,n}$ constructed in Section 5. It is well known that ind $\mathbb{R}^n = n$ ([Eng78]). Therefore, by Proposition 2, we have ind $\Sigma^\omega_{\perp,n} \geq n$. When $|\Sigma| = 1$, we prove it in proposition 7.

When $\Sigma = \{1\}$, $\Sigma^\omega_\perp$ is isomorphic to $P_\omega = \{a \mid a \subset N\}$ for $N = \{0, 1, 2, 3, \ldots\}$ by identifying $\alpha \in \Sigma^\omega$ with $a \in P_\omega$ when $\alpha[k] = 1$ iff $k \in a$. With this correspondence, $\Sigma^\omega_{\perp,n}$ corresponds to $P^{(n)}_\omega = \{a \in P_\omega \mid |N - a| \leq n\}$. Since there is the top element N in P_ω, every non-empty open set U includes N, and therefore, its closure is the whole space P_ω. This means that the boundary $B(U)$ is the complement $P_\omega - U$. This is also the case for $P^{(n)}_\omega$. That is, $B_X(U) = X - U$ for $X = P^{(n)}_\omega$.

Proposition 7. ind $P^{(n)}_\omega = n$.

Proof. Since we have already shown ind $P^{(n)}_\omega \leq n$, we need to prove ind $P^{(n)}_\omega \not\leq n - 1$. We prove it by induction on n.

It is immediate when $n = 0$. When $n > 0$, let $X = P^{(n)}_\omega$ and show that there exists an open set $U \subset X$ such that ind $B(U) \not\leq n-1$. Then, for each non-empty open subset $V \subset U$, we have $B(V) \supset B(U)$ because $B(V) = X - V$, and by heredity (Proposition 2), ind $B(V) \not\leq n - 1$.

Take $U = \{0\} \uparrow \cap X$. Then, $B(U) = X - U = \{a \in X \mid 0 \notin a\}$. This set is isomorphic to $\{a \mid a \subset N', |N' - a| \leq n - 1\}$ for $N' = \{1, 2, 3, \ldots\}$. Therefore, $B(U)$ is isomorphic to $P^{(n-1)}_\omega$, and thus ind $B(U) \not\leq n-1$. Note that the topological structure of $B(U)$ as a subspace of X and that of $P^{(n-1)}_\omega$ are the same.

From Theorem 3 and Proposition 3, we have the followings.

Corollary 1. *The computational dimension of a space is at least as large as its weak inductive dimension.*

Corollary 2. *There are no embeddings of* $\mathbb{R}^n$ *and* $\mathcal{I}^n$ *in* $\Sigma^\omega_{\perp,n-1}$ *for any character set* Σ.

Proof. $\mathrm{ind}\,\mathbb{R}^n = \mathrm{ind}\,\mathcal{I}^n = n$ ([Eng78]).

5 Embeddings of Finite Dimensional Spaces

In this section, we construct an embedding of $\mathbb{R}^2$ in $\Sigma^\omega_{\perp,2}$ for $\Sigma = \{0,1\}$, namely, interleaving $G(x)$ and $G(y)$ to define the name of $(x,y) \in \mathbb{R}^2$. Thus, the computational dimension of $\mathbb{R}^2$ is 2.

Let F be the function from $\Sigma^\omega_{\perp,1} \times \Sigma^\omega_{\perp,1}$ to $\Sigma^\omega_{\perp,2}$ which maps $(a_1a_2\ldots, b_1b_2\ldots)$ to $a_1b_1a_2b_2\ldots$. Then, F is a topological homeomorphism from $\Sigma^\omega_{\perp,1} \times \Sigma^\omega_{\perp,1}$ into $\Sigma^\omega_{\perp,2}$. Since $\mathbb{R}$ can be embedded in $\Sigma^\omega_{\perp,1}$ by the Gray code embedding G, we can topologically embed $\mathbb{R}^2$ into $\Sigma^\omega_{\perp,2}$ by $F \circ (G,G)$. In the same way, $\mathbb{R}^n$ can be embedded into $\Sigma^\omega_{\perp,n}$. Combining this fact with Corollary 1, we have

Theorem 4. $\mathbb{R}^n$ *has the computational dimension* n.

In order to show that the computability notion induced on $\mathbb{R}^2$ by $F \circ (G,G)$ in Definition 3 and 4 is the standard one, we show that the encoding F and the decoding F^{-1} can be expressed by an IM2-machine.

First, we consider encoding. That is, we construct an IM2-machine of type $(\Sigma^\omega_{\perp,1}, \Sigma^\omega_{\perp,1}, \Sigma^\omega_{\perp,2})$ which inputs two sequences in $\Sigma^\omega_{\perp,1}$ and outputs its interleaving in $\Sigma^\omega_{\perp,2}$.

The machine has two input tapes I_i $(i = 1,2)$ with two heads $\mathrm{H}_1(I_i)$ and $\mathrm{H}_2(I_i)$ on each I_i $(i = 1,2)$, and one output tape O with three heads $\mathrm{H}_1(O)$, $\mathrm{H}_2(O)$, and $\mathrm{H}_3(O)$. It has 4 states (c,s) for $c \in \{1,2\}$ and $s \in \{1,2\}$ with (1,1) the initial state. c indicates the tape to input the next character from, and s indicates the pair of heads used to output the next character: $s = 1$ means to output from $\mathrm{H}_1(O)$ and $\mathrm{H}_3(O)$, $s = 2$ means to output from $\mathrm{H}_2(O)$ and $\mathrm{H}_3(O)$. We need 16 rules corresponding to the combination of 4 states, 2 heads and 2 input characters. We abbreviate them into two rules with variables by allowing pattern matching on the left hand side of a rule, and using the *not* function defined as $not\ 1 = 2$ and $not\ 2 = 1$ on the right hand side.

$$\begin{aligned}(c,s)\ \mathrm{H}_1(I_c)(x) &\Rightarrow (not\ c, 1)\ \mathrm{H}_s(O)(x)\\ (c,s)\ \mathrm{H}_2(I_c)(x) &\Rightarrow (not\ c, not\ s)\ \mathrm{H}_3(O)(x)\end{aligned}$$

The first rule is read as when the state is (c,s) and it inputs the character x from the first head, then it changes its state to $(not\ c, 1)$ and output the character x from the s-th head. For the decoding function to the first component, we consider a machine of type $(\Sigma^\omega_{\perp,2}, \Sigma^\omega_{\perp,1})$ with the following rules:

$$\begin{aligned}(1,1)\ \mathrm{H}_1(I)(x) &\Rightarrow (2,1)\ \mathrm{H}_1(O)(x)\\ (1,2)\ \mathrm{H}_2(I)(x) &\Rightarrow (2,1)\ \mathrm{H}_1(O)(x)\\ (1,s)\ \mathrm{H}_3(I)(x) &\Rightarrow (2, not\ s)\ \mathrm{H}_2(O)(x)\\ (2,1)\ \mathrm{H}_1(I)(x) &\Rightarrow (1,1)\\ (2,2)\ \mathrm{H}_2(I)(x) &\Rightarrow (1,1)\\ (2,s)\ \mathrm{H}_3(I)(x) &\Rightarrow (1, not\ s)\end{aligned}$$

Thus, $f : \mathbb{R}^2 \to \mathbb{R}$ is $(F \circ (G, G), G)_\perp$-computable iff it is $(G, G, G)_\perp$-computable as a function in two arguments, and thus it is (ρ, ρ, ρ)-computable, where ρ is a representation equivalent to the Cauchy representation by Theorem 2.

For other finite dimensional spaces, it is immediate to show that S^1 $(= [0, 1]/\sim$ with $\sim$ defined as $0 \sim 1)$ can be embedded into $\Sigma^\omega_{\perp,1}$. From Figure 1, one can find that G can be modified to a function from S^1 by assigning $\perp 000000$ as $\mathrm{G}(0) = \mathrm{G}(1)$. Since $\mathrm{ind}\, S^1 = 1$, we have the following.

Proposition 8. *The computational dimension of S^1 is 1.*

More generally, we show in the next section that the computational dimension and the topological dimension coincide for all the separable metric spaces.

6 The Coincidence of the Computational Dimension and the Topological Dimension

Definition 8. *Let $\mathcal{I}$ be the unit open interval $(0, 1)$ and Q^n_i be the subspace of $\mathcal{I}^n$ with i dyadic coordinates, where a dyadic number is a rational number of the form $m \times 2^{-n}$ for integers m and n. We define*

$$N^n_k = Q^n_0 \cup \ldots Q^n_k.$$

That is, N^n_k is the subspace of $\mathcal{I}^n$ consisting of all points which have at most k dyadic coordinates.

It is known that N^n_k has dimension k ([Eng78]). The space N^{2n+1}_n is essentially the same as the Nöbeling's universal n-dimensional space, and it is universal for the class of all n-dimensional separable metric spaces in the following sense.

Proposition 9. *For any n-dimensional separable metric space X, there is a topological embedding of X in N^{2n+1}_n.*

Proof. See [Eng78], for example.

Thus, if we can embed N^{2n+1}_n in $\Sigma^\omega_{\perp,n}$, it means that we can embed any n-dimensional topological space in $\Sigma^\omega_{\perp,n}$, and therefore the computational dimension and the weak inductive dimension coincide for all the separable metric spaces.

Theorem 5. *There is an embedding from N^m_n to $\Sigma^\omega_{\perp,n}$ for $m \geq n$ and $\Sigma = \{0, 1\}$.*

Proof. We start with the embedding G^m of $\mathcal{I}^m$ in $\Sigma^\omega_{\perp,m}$ we constructed in Theorem 4. Since it is an interleaving of the Gray code, the number of $\perp$ which appear in $G^m(x)$ is equal to the number of dyadic coordinates of $x \in \mathcal{I}^m$. Therefore, when we restrict it to N^m_n, then the image has at most n dyadic coordinates. That is, $G^m(N^m_n) \subset \Sigma^\omega_{\perp,n}$. Thus, this restriction of G^m to N^m_n is a homeomorphic embedding to $\Sigma^\omega_{\perp,n}$.

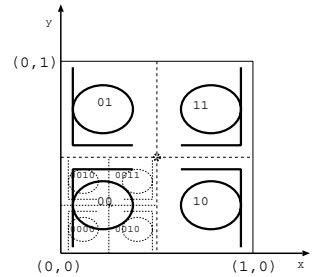


Fig. 2. The coding of N_1^2.

Corollary 3. *The computational dimension and the weak inductive dimension coincide for all the separable metric spaces.*

Proof. It is immediate from Theorem 5 when ind X is finite. When ind X is infinite, separable metric spaces are second countable T_0 spaces and as we note in the next section, every second countable T_0 spaces can be embedded into $\Sigma_\perp^\omega$. Therefore, the computational dimension of X is also infinite.

Corollary 4. *If a separable metric space can be embedded in $\Sigma_{\perp,n}^\omega$ for some alphabet Σ, then it can be embedded in $\{0,1\}_{\perp,n}^\omega$.*

Figure 2 depicts how the code G^2 of N_1^2 is composed. We split N_1^2 into four sub-areas, and assign the first two bits of $G^2(x)$ as 00, 01, 10, and 11 in each sub-areas. Note that N_1^2 is the unit square $\mathcal{I}^2$ minus points whose coordinates are both dyadic numbers. On the boundaries, we assign $\perp 0$ or $\perp 1$ on $x = 1/2$, and $0\perp$ or $1\perp$ on $y = 1/2$. Note that the center point $(1/2, 1/2)$ is not included in N_1^2. The rest of the bits are coded coinductively; we have the $1/2$ reduction of the code of N_1^2 on the right lower subsquare, which is flipped horizontally and vertically to fill all the subsquares so that they agree on the boundaries $x = 1/2$ and $y = 1/2$.

7 Embedding of Infinite Dimensional Objects

We consider infinite dimensional spaces which appear in computable analysis, like the set $\mathcal{A}^{(n)}$ of the closed subsets of $\mathbb{R}^n$, $\mathcal{O}^{(n)}$ of the open subsets of $\mathbb{R}^n$, $\mathcal{K}^{(n)}$ of the compact subsets of $\mathbb{R}^n$, and $C(A, \mathbb{R}^n)$ of the continuous functions from a subset $A \subset \mathbb{R}^m$ to $\mathbb{R}^n$. As for the topological structures on them, we have some possibilities. In this section, we consider $\mathcal{A}^{(n)}$ with the following three topological structures and discuss how they are embedded into $\Sigma_\perp^\omega$.

Let $Cb^{(n)}$ be the set of all open rational cubes of $\mathbb{R}^n$. $\tau_<^{\mathcal{A}}$ is the topology of $\mathcal{A}^{(n)}$ which has $\{A \subset \mathbb{R}^n \mid A \cap J \neq \emptyset\}$ as a subbase element for $J \in Cb^{(n)}$. In the same way, $\tau_>^{\mathcal{A}}$ is the topology which has $\{A \subset \mathbb{R}^n \mid A \cap \overline{J} = \emptyset\}$ as a

subbase element for $J \in Cb^{(n)}$, and $\tau^{\mathcal{A}}$ is the topology with the union of these two as the subbase. These three topologies are the final topologies of the three representations $\psi_<$, $\psi_>$ and ψ in [Wei00].

Let M be a homeomorphism from $\mathbb{R}$ to $(0,1)$. Then, since $\mathbb{R}$ includes infinite number of disjoint open intervals $(0,1),(1,2),(2,3),\ldots$, we can assign to $(a_0,a_1,\ldots) \in \mathbb{R}^\omega$ the closed set $\{M(a_0), M(a_1)+1, M(a_2)+2,\ldots\}$. It is easy to show that this map from $\mathbb{R}^\omega$ to $\mathcal{A}^{(1)}$ is a topological embedding with respect to $\tau^{\mathcal{A}}_<$, $\tau^{\mathcal{A}}_>$, and $\tau^{\mathcal{A}}$ on $\mathcal{A}^{(1)}$. In the same way, we can construct embeddings from $\mathbb{R}^\omega$ to $\mathcal{A}^{(n)}$ for every n. Therefore, $(\mathcal{A}^{(n)}, \tau^{\mathcal{A}}_<)$, $(\mathcal{A}^{(n)}, \tau^{\mathcal{A}}_>)$, and $(\mathcal{A}^{(n)}, \tau^{\mathcal{A}})$ are infinite dimensional spaces, and they have the computational dimension ∞ if they have embedding in $\Sigma^\omega_\perp$.

On the other hand, these spaces have natural embeddings into $P_\omega = \{1\}^\omega_\perp$. More generally, one can define an embedding of a second countable T_0-space X into P_ω, by fixing a subbase $O = \{O_i \mid i \in N\}$ with numbering, assigning i-th cell to O_i, and defining the embedding E of X as $E(x)[i] = 1$ iff $x \in O_i$ [Eng68]. This corresponds, in the standard representation theory, to considering the standard representation of S restricted to complete names where S is an effective topological space $S = (X, O, \mu)$ with $\mu(i) = O_i$, or when $\{(i,j) \mid O_i = O_j\}$ is r.e., considering the standard representation in a computable topological space $S = (X, O, \mu)$ [Wei00]. Applying this to the above-mentioned subbase, we define embeddings $E_<$, $E_>$, and E of $(\mathcal{A}, \tau^{\mathcal{A}}_<)$, $(\mathcal{A}, \tau^{\mathcal{A}}_>)$, and $(\mathcal{A}, \tau^{\mathcal{A}})$ to P_ω, respectively. Thus, these spaces have the computational dimension ∞.

When we use $\{0,1\}^\omega_\perp$ instead of $\{1\}^\omega_\perp$, we can not only express positive properties like $n \in O_i$ but also negative properties like $n \notin Cl(O_i)$. Therefore, we can define a new embbeding $\hat{E}$ of $(\mathcal{A}^{(n)}, \tau^{\mathcal{A}})$ to $\{0,1\}^\omega_\perp$ which is based on the subbase of $(\mathcal{A}^{(n)}, \tau^{\mathcal{A}}_<)$. For the embedding E, it is obvious that $E(x)$ includes infinite number of $\perp$-cells for each $x \in \mathcal{A}^{(n)}$. Our result that the computational dimension of $(\mathcal{A}^{(n)}, \tau^{\mathcal{A}})$ is infinite shows that $\hat{E}(x)$ also includes infinite number of $\perp$-cells for some $x \in \mathcal{A}^{(n)}$, and also says that one cannot think of an embbeding with finite number of $\perp$-cells for each element, even if we use bigger alphabet as Σ.

8 Conclusion

We have proposed a way of defining computation over a topological space X by considering an embedding of X in the sets $\Sigma^\omega_{\perp,n}$ $(n = 0,1,\ldots)$ of sequences of Σ in which bottoms are allowed to exist. We have shown that the computational dimension, which is the number of bottoms required in the name space, and the usual topological dimension coincide for separable metric spaces.

Since our theory is based on an embedding of a space into the domain $\Sigma^\omega_\perp$, it can be considered as a variant of domain theoretic approaches ([ES98], [Gia99], [SHT99],[Bla97]). On the other hand, each element of $\Sigma^\omega_{\perp,n}$ has a unique textual representation as an $n\perp$-sequence and we have the notion of a machine which operates on such extended sequences. In this sense, it can also be considered as a variant of more concrete representation-based approaches [Wei00].

One of the benefits of our embedding-based approach is that, since we consider embeddings in name spaces, we can study properties of computable functions over $\Sigma^\omega_{\perp,n}$ to study properties of computable functions over topological spaces in general. For example, multi-valued functions over X play an important role in computable analysis. They are represented by multi-valued functions over name spaces in our approach, whereas they are represented by single-valued functions over Σ^ω which are given multi-valued meaning through redundant representations in the standard representation theory. Our result relating the dimension of a space and the number of heads required to perform computation over the space is another example.

Acknowledgement

The author thanks Andreas Knobel and Izumi Takeuti for many illuminating discussions, and anonymous referees for invaluable comments.

References

BH00. Vasco Brattka and Peter Hertling. Topological properties of real number representations. *Theoretical Computer Science*, 2000. to appear.

Bla97. Jens Blanck. Domain representability of metric spaces. *Annals of Pure and Applied Logic*, 83:225–247, 1997.

Eng68. Ryszard Engelking. *Outline of General Topology.* North-Holland, Amsterdam, 1968.

Eng78. Ryszard Engelking. *Dimension Theory.* North-Holland, Amsterdam, 1978.

ES98. Abbas Edalat and Philipp Sünderhauf. A domain-theoretic approach to computability on the real line. *Theoretical Computer Science*, 210(1):73–98, 1998.

Gia99. Pietro Di Gianantonio. An abstract data type for real numbers. *Theoretical Computer Science*, 221:295–326, 1999.

HW48. Witold Hurewicz and Henry Wallman. *Dimension Theory.* Princeton University Press, 1948.

Nag65. Juniti Nagata. *Modern Dimension Theory.* North-Holland, Amsterdam, 1965.

SHT99. Viggo Stoltenberg-Hansen and John V. Tucker. Concrete models of computation for topological algebras. *Theoretical Computer Science*, 219:347–378, 1999.

Tsu01a. Hideki Tsuiki. Implementing real number computation in GHC. *Computer Software (in Japanese)*, 2001. to appear.

Tsu01b. Hideki Tsuiki. Real number computation through gray code embedding. *Theoretical Computer Science*, 2001. to appear.

Wei00. Klaus Weihrauch. *Computable Analysis, an Introduction.* Springer-Verlag, Berlin, 2000.

Some Properties of the Effective Uniform Topological Space

Yoshiki Tsujii*, Mariko Yasugi**, and Takakazu Mori

Faculty of Science, Kyoto Sangyo University
Kita-ku, Kyoto 603-8555, Japan
tsujiiy@cc.kyoto-su.ac.jp

Abstract. We develop the theory of the computability structure and some notions of computable functions on a uniform topological space, and will apply the results to some real functions which are discontinuous in the Euclidean space.
Keywords: Effective uniform topological space, Computability structure, Computable function, Binary tree structure, Cylinder function, Amalgamated space of reals, Function with jump points, Partial computability.

1 Introduction

We have recently been speculating on computability problems of some discontinuous real functions such as the Gaußian function ([20]), Rademacher functions ([23]) and Walsh functions ([7]). These are functions either on the whole real line or on the interval $[0, 1)$, jumping at integer points or at some dyadic rationals, and assuming integer values.

Throughout this article, knowledge of the notion of computability of a real number or a sequence of real numbers as well as the notion of computable functions over reals will be assumed. (A real number is computable if it can be approximated by a recursive sequence of rational numbers with recursive modulus of convergence.) For basic facts on computability results in the elementary calculus, we refer the reader to [11] and [22].

In traditional computable analysis, a real function f on a compact interval with computable end-points can be computable ("Grzegorczyk-computable") only if it is continuous. Namely, f is "Grzegorczyk-computable" if f preserves *sequential computability* and if f is *effectively uniformly continuous*, that is, f is uniformly continuous with a recursive modulus of continuity.

For a function on an open interval, the computability has been defined by effectively approximating the interval with compact intervals.

On the other hand, one often computes the values of a discontinuous function at jump points and draws the graph of such a function. For example, how does one draw a graph of a Gaußian function? One would first compute the values at integer points, that is, at jump points, plot them, and then draw horizontal

* Corresponding author.
** This work has been supported in part by the Scientific Grant of Japan No.12440031.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 336–356, 2001.

segments rightward. It is thus natural to attribute some notions of computability to some discontinuous functions.

There have been various approaches to this problem. In addition to the references as mentioned above, we quote only a few: [1], [2], [3], [5], [9], [11], [15], [16], [17], [18], [19], [20], [21], [23]. Computability theory on abstract algebras has been studied for example in [13] and [14]. The functional space approach, which has been initiated by [11], is especially useful and interesting.

Dealt in a function space, points of discontinuity of a discontinuous function are sort of smoothed out (by integration etc.). For this reason one can develop the computability theory for discontinuous functions in a function space, but at the same time, the value at a jump point cannot be talked about in such a theory.

On the other hand, some discontinuous functions will become continuous if one changes the topology of the domain of definition.

In a topological space with some kind of hierarchy of neighborhoods, one can develop a theory of the computability structure and define computable functions.

Here we propose such a theory of the computability structure on a uniform topological space, and will show that some real functions which are discontinuous with respect to the Euclidean topology will become continuous with respect to some uniform topology.

Although a uniform topological space with a countable index is topologically equivalent to a metric space, there is a point of working with the uniform topology. Namely, in viewing a discontinuous function as a continuous function by changing topology, there is *no need of computing new metrics*. What is involved is simply to describe the real line from a different viewpoint.

For the reader's convenience, we will first list the axioms of uniform topology with the index set $\mathbf{N}$, the set of natural numbers, and some of its consequences (Section 2). (For the definitions and properties concerning uniform topology, we have consulted [6] and [10].) A special kind of functions, called cylinder functions, are defined in terms of uniform topology. Cylinder functions are uniformly continuous.

The notion of effective uniform topology and the theory of the computability structure on a uniform topological space is developed in Section 3. There are three axioms, which are similar to the axioms for the computability structure on a metric space (cf. [9] and [21]).

The notion of relative computability (with respect to a computability structure) is first defined in Section 4, and with this notion, some sufficient conditions for computability will be proved.

A real-valued function f is *relatively computable* if f preserves *sequential computability* and if there is a recursive modulus of continuity for f with respect to each computable sequence. If furthermore the value of f at any point is approximated by the values of f at points in an *effectively approximating set*, then f is called *computable*.

A uniformly continuous function (on the whole space) is called uniformly computable if there is a recursive modulus of uniform continuity.

In Section 5, we will show that, as an example, the tree topology of a binary tree is an effective uniform topology, and that the family of all the recursive

paths in the binary tree form a computability structure. Furthermore, an effective enumeration of all the eventually zero paths is an effective separating set.

As an application, we show that the system of Rademacher functions forms a *uniformly computable sequence of functions*. Some other examples are also given in this section.

The next example, denoted by $\mathbf{A_R}$, is an *amalgamation* of the discrete space of integers and the union of all the open intervals of real numbers with end points of adjacent integers with relativized open interval topology (Section 6). As a set, $\mathbf{A_R}$ is identical with the set of real numbers, but it becomes a new effectively uniform topological space, which we will denote by $\mathcal{A}$.

The set of computable numbers of $\mathcal{A}$ is identical with the set of computable numbers (in $\mathbf{R}$), and the set of computable sequences in $\mathcal{A}$ is a subset of computable sequences (in $\mathbf{R}$). Some functions which jump at integer points (such as the Gaußian function) in the Euclidean topology will become continuous in $\mathcal{A}$, and can be shown to be computable.

In this section, examples and counter-examples for various notions in the preceding sections are supplied.

In the last section, Section 7, we will briefly discuss the computability of partial functions, and then list some further work we hope to carry out.

As mentioned above, there are various different approaches in a similar spirit, and it is of interest to compare our theory with others. In this article, however, we confined ourselves to see how far we can proceed with our theory.

2 Preliminaries: Uniform Topological Space

In the following, $\mathbf{N}$ will denote the set of natural numbers. Let X be a non-empty set.

Definition 2.1. (Uniform Topology: cf. [6] and [10]) Suppose there is a system of maps

$$V_n : X \longrightarrow \mathcal{P}(X)$$

for each $n \in \mathbf{N}$, where $\mathcal{P}(X)$ denotes the power set of X.

For each $x \in X$, $V_n(x)$ is called a *neighborhood* of x of *level* n, and $\{V_n\}$ is called a neighborhood system for X.

If this system satisfies the axioms $A_i, i = 1 - 5$ below, then it is called a *uniform topology* on X, and the structure $\langle X, \{V_n\}\rangle$ is called a *uniform topological space*.

A_1: $x \in V_n(x)$ for each $x \in X$ and $n \in \mathbf{N}$.

A_2: $\bigcap_{n\in\mathbf{N}} V_n(x) = \{x\}$

A_3: For every $n, m \in \mathbf{N}$, there is an l (independent of x) such that for every x

$$V_l(x) \subset V_n(x) \cap V_m(x).$$

A_4: For every $n \in \mathbf{N}$, there is an $m \in \mathbf{N}$ (independent of x, y) such that for every $x, y \in X$

$$x \in V_m(y) \text{ implies } y \in V_n(x).$$

A_5: For every $n \in \mathbf{N}$ there is an $m \in \mathbf{N}$ (independent of x, y, z) such that for every $x, y, z \in X$

$$x \in V_m(y) \text{ and } y \in V_m(z) \text{ imply } x \in V_n(z).$$

Note. A_1 in fact follows from A_2.

Corollary 1. $\mathcal{T} = \langle X, \{V_n(x) | n = 1, 2, 3, \cdots; x \in X\}\rangle$ is a topological space with $\{V_n(x) | n = 1, 2, 3, \cdots; x \in X\}$ as a system of neighborhoods. (We will subsequently denote $\mathcal{T}$ with $\langle X, \{V_n\}\rangle$ for short.)

Topological convergence with respect to $\{V_n\}$ can be defined as usual.

Definition 2.2. (Topological Convergence) Let $\mathcal{T} = \langle X, \{V_n\}\rangle$ be a uniform topological space.

(1) Let $\{x_k\}$ be a sequence from X and let $x \in X$. $\{x_k\}$ is said to *converge* to x (with respect to $\{V_n\}$) if

$$\forall n \exists j \forall k \geq j (x_k \in V_n(x)).$$

(2) Let $\{x_k\}$ be a sequence from X which satisfies the following.

$$\forall n \exists j \forall k \geq j (x_k \in V_n(x_j)).$$

Then, $\{x_k\}$ is called a *Cauchy sequence*.

(3) If every Cauchy sequence converges, then the space $\mathcal{T}$ is called *complete*.

Metric Example. Every metric space can be regarded as a uniform topological space with the index set $\mathbf{N}$ and $V_n(x) = \mathbf{B}(x, \frac{1}{2^n})$, where $\mathbf{B}(x, \frac{1}{2^n})$ is the open ball of size $\frac{1}{2^n}$ around x.

Note. Conversely, it is known classically that a uniform topological space with a countable index set can be metrized, and the two kinds of convergence are equivalent (cf.[6]).

The reason why we nevertheless work with a uniform topology with the index set $\mathbf{N}$ has been explained in the introduction.

Definition 2.3. (Continuity of Real Function) (1) A real-valued function f from a uniform topological space X is *continuous* if the following holds: for every $p = 1, 2, 3, \cdots$, and for every $x \in X$, there is an n such that

$$\text{for every } y \in V_n(x), |f(x) - f(y)| \leq \tfrac{1}{2^p}.$$

(2) A real-valued function f from a uniform topological space X is *uniformly continuous* if the following holds: for every $p = 1, 2, 3, \cdots$, there is an n such that for every $x \in X$

$$\text{for every } y \in V_n(x), |f(x) - f(y)| \leq \tfrac{1}{2^p}.$$

Note. The correspondence of $n (= n_p)$ to p can be called a modulus of uniform continuity for f.

As a special kind of functions, we will mention the following for a later use.

Definition 2.4. (Cylinder Function) A real-valued function f from X will be called a *cylinder function* if there is an n such that, for every $x \in X$,

$$\forall y \in V_n(x)(f(x) = f(y)).$$

From Definitions 2.3 and 2.4 follows immediately the next proposition.

Proposition 2.1. (Uniform Continuity of Cylinder Function) A cylinder function is uniformly continuous.

3 Topological Computability

Let $\mathcal{T} = \langle X, \{V_n\}\rangle$ be a uniform topological space. We will define some notions and objects for $\mathcal{T}$ which have computational meanings. Examples will be presented in Sections 5 and 6.

Definition 3.1. (Effective Uniform Topology) A uniform topology $\{V_n\}$ on X is called an *effective uniform topology* if there are recursive functions $\alpha_1, \alpha_2, \alpha_3$ which satisfy the following.

For every $n, m \in \mathbf{N}$ and every $x \in X$, $V_{\alpha_1(n,m)}(x) \subset V_n(x) \cap V_m(x)$ (effective A_3).

For every $n \in \mathbf{N}$ and every $x, y \in X$, $x \in V_{\alpha_2(n)}(y)$ implies $y \in V_n(x)$ (effective A_4).

For every $n \in \mathbf{N}$ and every $x, y, z \in X$,

$$x \in V_{\alpha_3(n)}(y) \text{ and } y \in V_{\alpha_3(n)}(z) \text{ imply that } x \in V_n(z)$$

(effective A_5).

Effective convergence and the computability structure for $\mathcal{T}$ can be defined similarly to the counterparts of metric spaces (See [9] and [21]).

Definition 3.2. (Effective Convergence) (1) A sequence $\{x_k\}$ for X is said to *effectively converge* to a point x in X if there is a recursive function γ satisfying $\forall x \forall k \geq \gamma(n)(x_k \in V_n(x))$.

(2) A multiple sequence $\{x_{\nu k}\}$ from X is said to *effectively converge* to a sequence $\{x_\nu\}$ if there is a recursive function β satisfying

$$\forall \nu \forall n \forall l \forall k \geq \beta(\nu, n)(x_{\nu k} \in V_n(x_\nu)).$$

Note. Although the convergence of a sequence of points to a point is classically natural and adequate, it is important to consider the convergence of a double (or a multiple) sequence to a sequence when the effectivity is the central subject. The same applies also to some other notions, and hence we will subsequently give two definitions, a property concerning a point and the one concerning a sequence of points.

Definition 3.3. (Computability Structure) Let $\mathcal{S}$ be a family of sequences from X. (As usual, we include multiple sequences, such as double sequences and triple sequences, when we talk about sequences.)

$\mathcal{S}$ is called a *computability structure* if it satisfies the following.

C1: (Non-emptiness) $\mathcal{S}$ is nonempty.

C2: (Re-enumeration) If $\{x_k\} \in \mathcal{S}$ and α is a recursive function,

then $\{x_{\alpha(i)}\}_i \in \mathcal{S}$.

More generally, if $\{x_{\nu_1,\nu_2,\cdots,\nu_k}\} \in \mathcal{S}$, and if $\alpha_1, \alpha_2, \cdots, \alpha_k$ are recursive functions, then $\{x_{\alpha_1(\nu),\alpha_2(\nu),\cdots,\alpha_k(\nu)}\} \in \mathcal{S}$, where ν stands for $\nu_1, \nu_2, \cdots, \nu_k$.

C3: (Limit) If $\{x_{lk}\}$ belongs to $\mathcal{S}$, $\{x_l\}$ is a sequence from X, and $\{x_{lk}\}$ converges to $\{x_l\}$ effectively, then $\{x_l\} \in \mathcal{S}$. That is, $\mathcal{S}$ is closed with respect to effective convergence.

Similarly to C2 above, we can also read l as a finite sequence of indices ν.

A sequence belonging to $\mathcal{S}$ is called a *computable sequence*. An element x of X is called *computable* if the sequence $\{x, x, \cdots\}$ is computable.

For an effective uniform topological space with a computability structure $\mathcal{S}$, we will denote $\mathcal{C}_\mathcal{T} = \langle X, \{V_n\}, \alpha_1, \alpha_2, \alpha_3, \mathcal{S}\rangle$.

Note. In Definition 1.4 of [21], the first axiom of the computability structure for a metric space requires that the sequence of distances for any two sequences in $\mathcal{S}$ be a computable double sequence of real numbers. In the topological computability, we need not talk about distances. Instead, we only require that $\mathcal{S}$ be non-empty.

The subsequent universe of discourse will be $\mathcal{C}_\mathcal{T}$. We will list some definitions concerning computability structures for later use.

Definition 3.4. (Effective Approximation) Suppose a computable sequence $\{e_k\}$ satisfies the following. For each computable sequence $\{x_l\}$, there is a recursive function ν such that

$$\forall n \forall l (e_{\nu(n,l)} \in V_n(x_l)).$$

Then $\{e_k\}$ is called an *effective approximating set* of the computability structure $\mathcal{S}$, and $\mathcal{S}$ is said to be *effectively approximated*.

The notion of effective approximation is weaker than the following effective separability, and will be used, for example, in Proposition 4.1.

Definition 3.5. (Effective Separability) Let $\{e_k\}$ be an effective approximating set. Suppose furthermore that it is dense in X, that is, $\forall n \forall x \exists k (e_k \in V_n(x))$. Then the space $\mathcal{C}_\mathcal{T}$ is said to be *effectively separable*, and $\{e_k\}$ is called an *effective separating set*.

Note. In the metric space treated as in [21], an effective separating set is defined first, and, as a corollary, it follows that a computable sequence is effectively approximated by it. In Section 6 (Example 5), we will see an example of an effective approximating set which is not dense in the space.

Notation. In the following, we let $\mathcal{E}_{\mathcal{T}} = \langle X, \{V_n\}, \alpha_1, \alpha_2, \alpha_3, \mathcal{S}, \{e_k\}\rangle$ denote an effective uniform topological space with a computability structure and an effective *approximating* set.

Definition 3.6. (Relative Effective Completeness) (1) A sequence from X, say $\{x_j\}$, is said to be *effectively Cauchy* if there is a recursive function α such that

$$\forall n \forall j \geq \alpha(n)(x_j \in V_n(x_{\alpha(n)})).$$

(2) A double sequence from X, say $\{x_{mj}\}$, is said to be *effectively Cauchy* if there is a recursive function α such that

$$\forall n \forall m \forall j \geq \alpha(m,n)(x_{mj} \in V_n(x_{m\alpha(m,n)})).$$

(2) The space $\mathcal{E}_{\mathcal{T}}$ is said to be *relatively effectively complete* (with respect to $\mathcal{S}$) if every computable and effectively Cauchy double sequence $\{x_{mj}\}$ effectively converges to a sequence in X.

Note. 1) The definition of an effective Cauchy sequence above is appropriate due to the effective version of A5 (Definition 3.1).

2) In (2) above, the limit sequence is computable by virtue of the third axiom C3 (Definition 3.3).

3) Effective completeness with respect to a computability structure does *not* imply completeness of the whole space, as will be seen in Example 4 of Section 6.

4) In [21], the definition (2) above was a corollary if the space was complete.

5) We gave the definition of the effectivity of a Cauchy sequence for a double sequence since a simple Cauchy sequence is hardly of use in relation to the effectivity. (See 2) above.)

The remark below has been added according to a referee's advice.

Remark 1. Any double sequence that effectively converges (Definition 3.2) is effectively Cauchy.

This can be shown as follows. Suppose that $\{x_{mj}\}$ effectively converges to $\{x_m\}$, that is, for a recursive function $\beta(m,n)$, it holds that

$$\forall j \geq \beta(m,n)(x_{mj} \in V_n(x_m)).$$

Put

$$\alpha(m,n) = \beta(m, \max\{\alpha_2(\alpha_3(n)), \alpha_3(n)).$$

where the functions α_2 and α_3 are in Definition 3.1. Then,

$$x_{m\alpha(m,n)} \in V_{\alpha_2(\alpha_3(n))}(x_m)$$

and so

(a) $x_m \in V_{\alpha_3(n)}(x_{m\alpha(m,n)})$

and

(b) $\forall j \geq \alpha(m,n)(x_{mj} \in V_{\alpha_3(n)}(x_m))$.

(a) and (b) imply that $x_{mj} \in V_n(x_{m\alpha(m,n)})$ for $j \geq \alpha(m,n)$.

4 Computable Functions

In [21], we defined the (uniformly) computable functions on an effectively compact space, and then later extended a computable function (not necessarily uniform) as defined on an effectively σ-compact set. Here we take a slightly different approach. We do *not* assume compactness or σ-compactness on a space, and, this way, we can deal with computability of a function relative to a computability structure.

In defining some notions of computable functions, we have the following examples (in a metric space) in mind.

Example A. Let $X = (0, 1]$ and define $V_l(x) = B(x, \frac{1}{2^l}) \cap X$. Consider a function $f(x) = \frac{1}{x}$ defined on X.

Example B. Let X and $V_l(x)$ be as above. Consider a sequence of functions $\{f_n(x) = x^{-1+\frac{1}{n}}\}$ defined on X.

Note. Here the space $(0, 1]$ is considered in the usual Euclidean topology. When we extend a function such as $\frac{1}{x}$ to the closed interval $[0, 1]$ so that $f(0) = 0$, for example, we would view it as a disjoint union of $\{0\}$ and $(0, 1]$ so that f will be a continuous function on the whole set. Such a case will be discussed in Section 6.

The next definition concerns the computability of a function with respect to the computability structure $\mathcal{S}$. We find it natural to first consider some notion of computability on computable objects.

Definition 4.1. (Relative Computability) (1) A real-valued function f from an effective uniform space $\mathcal{C}_\mathcal{T}$ is called *relatively computable* (with respect to $\mathcal{S}$) if the following hold.

(i) f preserves sequential computability, that is, for any computable sequence $\{x_m\}$ from X, $\{f(x_m)\}$ is a computable sequence of real numbers.

(ii) For any $\{x_m\} \in \mathcal{S}$ there exists a recursive function $\gamma(m, p)$ such that $y \in V_{\gamma(m,p)}(x_m)$ implies $|f(y) - f(x_m)| \leq \frac{1}{2^p}$.

(2) A sequence of real-valued functions from an effective uniform space $\mathcal{C}_\mathcal{T}$, say $\{f_n\}$, is called *relatively computable* (with respect to $\mathcal{S}$) if the following hold.

(iii) $\{f_n\}$ preserves sequential computability, that is, for any computable sequence $\{x_m\}$ from X, $\{f_n(x_m)\}$ is a computable double sequence of real numbers.

(iv) For any $\{x_m\} \in \mathcal{S}$ there exists a recursive function $\gamma(n, m, p)$ such that $y \in V_{\gamma(n,m,p)}(x_m)$ implies $|f_n(y) - f_n(x_m)| \leq \frac{1}{2^p}$.

Let us note that a function f is relatively computable in the sense of (1) if and only if the sequence $\{f, f, f, \cdots\}$ is relatively computable in the sense of (2).

Note. Notice that in the definition above, the continuity of the function is *not* assumed for relative computability. A counter-example will be given in Section 6 (Example 8).

Proposition 4.1. (A Special Case of Sufficiency) Assume that $\{f_n\}$ satisfies the following.

(i) $\{f_n\}$ preserves sequential computability.

(ii) There exists an effective *approximating* set $\{e_k\}$ (Definition 3.4) satisfying that $\{f_n\}$ is effectively uniformly continuous with respect to $\{e_k\}$, that is, there exists a recursive function $\gamma_0(n,p)$ (independent of k) for which

$$y \in V_{\gamma_0(n,p)}(e_k) \text{ implies } |f_n(y) - f_n(e_k)| \leq \tfrac{1}{2^p}$$

for any k.

Then $\{f_n\}$ is relatively computable (with respect to $\mathcal{S}$). Furthermore $\{f_n\}$ is effectively uniformly continuous with respect to any $\{x_m\} \in \mathcal{S}$ (instead of $\{e_k\}$).

Proof: For the sake of brevity, we will prove the proposition for the case of a single function instead of a sequence of functions. So we show that for $\{x_m\} \in \mathcal{S}$ there exists a recursive function $\gamma(p)$ independent of m such that

$$y \in V_{\gamma(p)}(x_m) \text{ implies } |f(y) - f(x_m)| \leq \tfrac{1}{2^p}.$$

Recall that α_3 and α_2 in Definition 3.1 satisfy

(a) $y \in V_{\alpha_3(n)}(x)$ and $x \in V_{\alpha_3(n)}(e)$ imply $y \in V_n(e)$,

and

(b) $e \in V_{\alpha_2(l)}(x)$ implies $x \in V_l(e)$.

Put $n_p = \gamma_0(p+1)$ and $l_p = \alpha_3(n_p)$. Since $\{e_k\}$ is an effective approximating set, there exists a recursive function $\nu(n,m)$ such that

$$(e_{m,p} =) e_{\nu(\alpha_2(l_p),m)} \in V_{\alpha_2(l_p)}(x_m)$$

for m= 1,2, $\cdots$.

It follows that $x_m \in V_{l_p}(e_{mp}) = V_{\alpha_3(n_p)}(e_{mp})$ by (b) above, and so (a) implies $y \in V_{n_p}(e_{mp})$ for any $y \in V_{\alpha_3(n_p)}(x_m)$. Note that $x_m \in V_{n_p}(e_{mp})$. Put $\gamma(p) = \alpha_3(\gamma_0(p+1)) = \alpha_3(n_p)$, which is independent of m. If $y \in V_{\gamma(p)}(x_m) = V_{\alpha_3(n_p)}(x_m)$, we have

$$|f(x_m) - f(y)| \leq |f(x_m) - f(e_{mp})| + |f(e_{mp}) - f(y)| \leq \frac{1}{2^{p+1}} + \frac{1}{2^{p+1}} = \frac{1}{2^p}.$$

The construction of γ above is uniform in f, and hence goes through for a sequence of functions as well.

From the proof above, we can deduce the following Proposition, which presents a sufficient condition for the relative computability of functions.

Proposition 4.2. (A Sufficient Condition) Assume that $\{f_n\}$ satisfies the following.

(i) $\{f_n\}$ preserves sequential computability.

(ii) There exist an effective approximating set $\{a_k\}$ and a recursive function $\gamma_0(n,k,p)$ such that

(ii-a) for any n, k and p, $y \in V_{\gamma_0(n,k,p)}(a_k)$ implies $|f_n(y) - f_n(a_k)| \leq \frac{1}{2^p}$, and

(ii-b) for any $\{x_m\} \in \mathcal{S}$, there is a recursive function $\nu_0(n, m, p)$ (depending on $\{x_m\}$) for which hold

$$x_m \in V_{\alpha_3(\gamma_0(n,\overline{\nu}_n,p+1))}(a_{\overline{\nu}_n})$$

where $\overline{\nu}_n = \nu_0(n, m, p)$.

Then $\{f_n\}$ is relatively computable.

For the computability of functions, we adopt the following definition.

Definition 4.2. (Computable Function) (1) A function f is called *computable* if the following hold.

(i) f preserves sequential computability.

(ii) f is relatively computable, and there exist an effective approximating set, say $\{e_k\} \in \mathcal{S}$, and a recursive function $\gamma_0(k, p)$ for which

$$y \in V_{\gamma_0(k,p)}(e_k) \text{ implies } |f(y) - f(e_k)| \leq \tfrac{1}{2^p}$$

and

$$\bigcup_{k=1}^{\infty} V_{\gamma_0(k,p)}(e_k) = X$$

for p.

(2) A sequence of functions, say $\{f_n\}$, is called *computable* if there is a recursive function $\gamma_0(n, k, p)$ (depending also on n) for which (i) and (ii) above hold (for every n).

An example which is relatively computable but not computable will be given in Section 6 (Example 7).

Corollary 2. Suppose that the assumptions in Proposition 4.2 hold for $\{f_n\}$, and furthermore suppose that the corresponding $\{e_k\}$ satisfies

$$\bigcup_{k=1}^{\infty} V_{\gamma_0(n,k,p)}(e_k) = X \text{ for any } n \text{ and } p.$$

Then $\{f_n\}$ is computable.

Example A. $X = (0, 1]$ and $V_n(x) = B(x, \frac{1}{n}) \cap X$. A single function $f(x) = \frac{1}{x}$ defined on X is computable.

On the interval $I_j = (\frac{1}{2^j}, 1]$, the function $f(x) = \frac{1}{x}$ is uniformly continuous: if $x, y \in I_j$ and $|x - y| \leq 2^{-(2j+p)}$, $|f(x) - f(y)| \leq \frac{1}{2^p}$.

Let $\{e_k\}$ be an effective enumeration of rationals in X and $\eta(k)$ be the recursive function such that $\eta(k) = q$ with $\frac{1}{2^q} < e_k \leq \frac{1}{2^{q-1}}$. Put $\gamma_0(k, p) = 2^{2(\eta(k)+1)+p}$, then the property for $\gamma_0(k, p)$ in Definition 4.2 holds. The relative computability (Definition 4.1) is similarly shown.

For function convergence, we have the examples A and B above in mind.

Definition 4.3. (Convergence of Functions) (1) $\{f_n\}$ is said to *relatively effectively converge* to f if, for any $\{x_m\} \in \mathcal{S}$, there exist recursive functions $\eta(m, p)$ and $\delta(m, p)$ such that

for $n \geq \eta(m,p)$ and $y \in V_{\delta(m,p)}(x_m)$, it holds that $|f_n(y) - f(y)| \leq \frac{1}{2^p}$.

(2) $\{f_n\}$ is said to *effectively converge* to f if it relatively effectively converges to f, and, furthermore, there exist an effective approximating set $\{e_k\} \in \mathcal{S}$ and recursive functions $\eta_0(k,p)$ and $\delta_0(k,p)$ such that

for $n \geq \eta_0(k,p)$ and $y \in V_{\delta_0(k,p)}(e_k)$, it holds that $|f_n(y) - f(y)| \leq \frac{1}{2^p}$

and

$$\bigcup_{k=1}^{\infty} V_{\delta_0(k,p)}(e_k) = X.$$

Examples A and B. $\{x^{-1+\frac{1}{n}}\}$ effectively converges to $\frac{1}{x}$.

An example of a sequence of functions which relatively effectively converges but does not effectively converge will be given in Example 8.

For the next proposition we define equi-computability of functions.

Definition 4.4. (Equi-computability) $\{f_n\}$ is called *relatively equi-computable* if the following hold.

(i) $\{f_n\}$ preserves sequential computability.

(ii) For any $\{x_m\} \in \mathcal{S}$ there exists a recursive function $\gamma(m,p)$ (independent of n) such that

$$y \in V_{\gamma(m,p)}(x_m) \text{ implies } |f_n(y) - f_n(x_m)| \leq \tfrac{1}{2^p}.$$

Proposition 4.3. (Limit Function) Let $\{f_n\}$ be relatively equi-computable, and let $\{f_n\}$ relatively effectively converge to f. Then $f(x)$ is relatively computable.

Proof: We will show that (i) and (ii) in Definition 4.1 hold. (i) (Proof of sequential computability) Suppose $\{x_m\} \in \mathcal{S}$. Then $\{f_n(x_m)\}$ is a computable double sequence of reals, and it holds that

$$|f_n(x_m) - f(x_m)| \leq \tfrac{1}{2^p} \text{ if } n \geq \eta(m,p),$$

for $\{f_n\}$ relatively effectively converges to $\{f\}$. It follows that $\{f(x_m)\}$ is a computable sequence of reals.

(ii) Fix $\{x_m\} \in \mathcal{S}$. The convergence of $\{f_n\}$ to f means that

(a) for $n \geq \eta(m,p)$ and $y \in V_{\delta(m,p)}(x_m)$ it holds that $|f_n(y) - f(y)| \leq \frac{1}{2^p}$.

The equi-computability of $\{f_n\}$ means that

(b) $y \in V_{\gamma(m,p)}(x_m)$ implies $|f_n(y) - f_n(x_m)| \leq \frac{1}{2^p}$.

Put $\gamma_0(m,p) = \alpha_1(\delta(m,p), \gamma(m,p))$ where $\alpha_1(n,m)$ is in Definition 3.1, that is, $V_{\alpha_1(n,m)}(x) \subset V_n(x) \cap V_m(x)$.

For $n \geq \eta(m,p)$ and $y \in V_{\gamma_0(m,p)}(x_m)$, we have

$$|f(x_m) - f(y)| \leq |f(x_m) - f_n(x_m)| + |f_n(x_m) - f_n(y)| + |f_n(y) - f(y)| \leq \frac{3}{2^p}$$

by (1) and (2), noting that $x_m \in V_{\delta(m,p)}(x_m)$, $y \in V_{\gamma(m,p)}(x_m)$ and $y \in V_{\delta(m,p)}(x_m)$ hold.

Example 10 in Section 6 will be an example of such a convergence.

Corollary 3. (A Sufficiency for Limit Computability) Let $\{f_n\}$ be relatively equi-computable and let $\{e_k\}$ be an effective *separating* set. Suppose $\{f_n\}$ effectively converges to f. Furthermore suppose that the corresponding $\delta(k,p)$ in Definition 4.3 for $\{e_k\}$ and $\gamma(k,p)$ in Definition 4.4 for $\{e_k\}$ satisfy

$$\bigcup_{k=1}^{\infty} V_{\alpha_1(\delta(k,p),\gamma(k,p))}(e_k) = X.$$

Then $f(x)$ is computable .

Example. Examples A and B satisfy the assumptions in the corollary above.

Definition 4.5. (Uniform Computability) (1) A function f on X is called *uniformly computable* if f preserves sequential computability and if there is a recursive modulus of uniform continuity for f.

(2) A sequence of functions $\{f_n\}$ is called uniformly computable if it preserves sequential computability and there is a recursive function α such that

$$y \in V_{\alpha(n,p)}(x) \Rightarrow |f_n(x) - f_n(y)| \leq \frac{1}{2^p}.$$

Uniform computability is the strongest of all the notions of computability of a function.

An example of a computable function which is not uniformly computable is given by $\frac{1}{x}$ on the space $(0,1]$ as in Example A. See also Examples 6 and 9 in Section 6.

5 Example: Binary Tree and Cylinder Function

Let Ω be the set of infinite binary sequences, that is, a sequence in Ω is of the form $\langle s_1, s_2, \cdots, s_n, \cdots \rangle$, where each $s_n = 0$ or $= 1$.

Since the space Ω has been well-studied, we will only review some facts and notations concerning it.

We will denote the set of finite binary sequences by Ω^*. Thus, an element of Ω^* is of the form $\langle s_1, s_2, \cdots, s_n \rangle$, where each s_i is either 0 or 1.

Elements of Ω will be denoted by σ, τ etc., and $\sigma|n$ will denote the finite sequence $\langle \sigma(1), \sigma(2), \cdots, \sigma(n) \rangle$.

We will give an outline of the tree topology on Ω, by first defining a system of maps denoted by $\mathcal{U}$.

Definition 5.1. (Base System) For each $n = 1, 2, \cdots$, a map $U_n : \Omega \to \mathcal{P}(\Omega)$ is defined as follows. For each $\sigma \in \Omega$ and for each $n = 1, 2, 3, \cdots$, put

$$U_n(\sigma) = \{\tau |\, \tau|n = \sigma|n\}.$$

Each such $U_n(\sigma)$ is called a cylinder set. The collection of such sets is known to be a base system for Ω.

Put $\mathcal{U} = \{U_n\}_n$, and denote the space $\langle \Omega, \mathcal{U} \rangle$ by $\mathcal{T}_{\mathcal{B}}$.

The following propositions follow immediately from the definition.

Proposition 5.1. (Tree Topology) (1) $\mathcal{T}_{\mathcal{B}} = \langle \Omega, \mathcal{U} \rangle$ is a uniform topological space.

(2) The topology determined by $\langle \Omega, \mathcal{U} \rangle$ induces a neighborhood system $\{U_\varepsilon\}$, where $U_\varepsilon = \{\sigma | \sigma | n = \varepsilon\}$ for $\varepsilon \in \Omega^*$ of length n.

$U_n(\sigma)$ can be also written as $U_{\sigma|n}$.

(3) For bases $U_n(\sigma)$ and $U_m(\tau)$, where $m \geq n$, if $U_n(\sigma) = U_n(\tau)$, then $U_m(\tau) \subset U_n(\sigma)$; otherwise, $U_m(\tau) \cap U_n(\sigma) = \phi$.

(4) The space $\langle \Omega, \mathcal{U} \rangle$ is separable and compact.

Proposition 5.2. (Effective Tree Topology) The space $\mathcal{T}_{\mathcal{B}} = \langle \Omega, \mathcal{U} \rangle$ is an *effective uniform topological space.*

Proof: It suffices to show that there are recursive functions for $A_i, i = 3, 4, 5$. For A_3, define l to be $\alpha_1(n, m) = \max(m, n)$. For simplicity, assume $l = n$. Then $U_l(\sigma) = U_n(\sigma)$ and $U_n(\sigma) \subset U_m(\sigma)$, and hence $U_l(\sigma) \subset U_m(\sigma) \cap U_n(\sigma)$.

For A_4, let $\alpha_2(n) = n$. If $\sigma \in U_n(\tau)$, then $\sigma | n = \tau | n$, and hence $\tau \in U_n(\sigma)$.

For A_5 also, let $\alpha_3(n) = n$. If $\sigma \in U_n(\tau)$ and $\tau \in U_n(\rho)$, then $\sigma | n = \tau | n = \rho | n$, which implies $\sigma \in U_n(\rho)$.

A computability structure for Ω is defined as follows. (See Definition 3.3 for the computability structure.)

Proposition 5.3. (Computability Structure on Ω) (1) Let $\mathcal{S}_\Omega$ be the family of recursive sequences from Ω. Then $\mathcal{S}_\Omega$ forms a computability structure. (As usual, we include multiple sequences such as double sequences, triple sequences, etc., when we talk about sequences.)

(2) An effective enumeration of "eventually constant 0" sequences is an effective separating set (Definition 3.5) for $\langle \Omega, \mathcal{U}, \mathcal{S}_\Omega \rangle$.

(3) The space $\langle \Omega, \mathcal{U}, \mathcal{S}_\Omega \rangle$ is relatively effectively complete (Definition 3.6).

We will consider a special type of real-valued functions on Ω, that is, a function which is determined by finite information of a fixed length. Such a function is a cylinder function (Definition 2.4).

The following facts are generally known.

Proposition 5.4. (Cylinder Function on Ω) (1) A real-valued function on Ω is a *cylinder function* if and only if there is an n such that for every σ and for every $\tau \in U_n(\sigma)$, $f(\tau) = f(\sigma)$.

Let us call the least such n the *length* of f.

(2) For any cylinder function, say f, there are finitely many disjoint cylinder sets whose union is the whole space, and f is constant on each cylinder set. As an obvious consequence, a cylinder function has only finitely many values.

Proposition 5.5. (Computability of Cylinder Function) A cylinder function on Ω with computable values is uniformly computable (cf. Definition 4.5). (There are only finitely many values by virtue of (ii) of Proposition 5.4.)

As an example of (2) of Definition 4.5, we will consider the following.

Definition 5.2. (Rademacher Functions) Let $n = 1, 2, \cdots$. The n-th *Rademacher function* $\phi_n(\sigma)$ on Ω is defined as follows.

$$\phi_n(\sigma) = \begin{cases} 1, & \sigma(n) = 0 \\ -1, & \sigma(n) = 1. \end{cases}$$

As an immediate consequence, we have that the n-th Rademacher function ϕ_n is a cylinder function of length n, and $\{\phi_n\}$ is uniformly equi-continuous on Ω with a common modulus of uniform continuity for all n. (We have consulted [4] and [12] for Rademacher functions and Walsh analysis.)

Proposition 5.6. (Computability of Rademacher System) The Rademacher functions $\{\phi_n\}$ form a uniformly computable sequence of functions.

With a same kind of reasoning, we can claim that the system of Walsh functions is a uniformly computable sequence of functions. This fact is related to some results in [7], but we will not elaborate on it here.

We will give some other examples of functions on a tree.

Example 1. Let f be a real-valued function on Ω:

$$f(\sigma) = \Sigma \frac{\sigma_n}{2^n}.$$

(We regard this as a function from a topological space to reals, not as a representation.) f is a uniformly computable function.

Example 2. Instead of a binary tree, we can consider a ternary tree Ω^3. We can easily see that an effective uniform (tree) topology can be introduced to this tree. The real-valued function f:

$$f(\sigma) = \Sigma \frac{\sigma_n}{3^n}$$

is a uniformly computable function ($\sigma_n = 0, 1, 2$).

Example 3. Let Ω_c^3 be the Cantor space:

$$\Omega_c^3 = \{\sigma \in \Omega^3 | \sigma_k = 1 \quad \text{for} \quad k \geq \min\{j | \sigma_j = 1\} \quad \text{if there is such} \quad j\}.$$

Define $U_n(\sigma) = \{\tau \in \Omega_c^3 | \tau | n = \sigma | n\}$. Then Ω_c^3 with $\{U_n\}$ is an effective uniform space and the function f:

$$f(\sigma) = \Sigma \frac{\sigma_n}{3^n}$$

is a uniformly computable function.

6 Example: Amalgamated Space $\mathcal{A}$ and Functions with Integer-Jumps

If we regard the real space $\mathbf{R}$ as the disjoint union of the integer set and the open intervals with consecutive integer ends, to which uniform topology can be introduced, then we can treat functions which jump at integer points and are continuous otherwise (in the Euclidean topology) as continuous functions.

Definition 6.1. (Set of Real Numbers as an Amalgamation) $\mathbf{Z}$ and $\mathbf{R}$ respectively will denote the set of integers and the set of reals. We assume $k \in \mathbf{Z}$.

(1) We will use the following notations.

$$J_{\mathbf{Z}} := \mathbf{Z}, J_k := (k, k+1), \mathbf{J} = \bigcup_k J_k, \mathbf{A_R} := \mathbf{J} \cup J_{\mathbf{Z}}.$$

(2) We define a basis of uniformity (a basis of neighborhoods) for $J_{\mathbf{Z}}$ and J_k above as follows.

For $J_{\mathbf{Z}}$, $V_n^{\mathbf{Z}}(x) = \{x\}$ for every $n = 1, 2, 3, \cdots$ and for every $x \in J_{\mathbf{Z}}$; for J_k, $k \in \mathbf{Z}$, $V_n^k(x) = (x - \frac{1}{2^n}, x + \frac{1}{2^n}) \cap J_k$, for all $x \in J_k$ and for $n = 1, 2, 3, \cdots$.

(3) The *amalgamated space* is the pair $\mathcal{A} = \langle \mathbf{A_R}, \mathcal{W} \rangle$, where $\mathbf{A_R}$ is the set defined in (1) and $\mathcal{W}$ denotes a system of maps $\{W_n\}$ from $\mathbf{A_R}$ to $\mathcal{P}(\mathbf{A_R})$, the power set of $\mathbf{A_R}$, defined by

$$W_n(x) = V_n^{\mathbf{Z}}(x) \text{ if } x \in J_{\mathbf{Z}}; W_n(x) = V_n^k(x) \text{ if } x \in J_k.$$

Note. Obviously, as sets, $\mathbf{J_Z}$ is nothing but $\mathbf{Z}$ and $\mathbf{A_R}$ is nothing but $\mathbf{R}$. We use the notations $\mathbf{J_Z}$ and $\mathbf{A_R}$ so that it will be clear which topology is discoursed.

The classical uniformity of the topology determined by $\{W_n\}$ will be included in its effective uniformity as stated below.

Proposition 6.1. (Effective Uniformity of the Spaces) (i) The spaces $\mathcal{J}_Z = \langle J_{\mathbf{Z}}, \{V_n^{\mathbf{Z}}\} \rangle$ and $\mathcal{J} = \langle \mathbf{J}, \{V_n^k\} \rangle$ are effective uniform topological spaces.

(ii) The amalgamated space $\mathcal{A} = \langle \mathbf{A_R}, \{W_n\} \rangle$ is an effective uniform topological space.

Proof: (ii) follows from (i) and the definition of $\mathcal{W} = \{W_n\}$.

For (i), we can take the following recursive functions for both cases.

A_1 and A_2 are trivial.

A_3 Put $\alpha_1(m, n) = \max(m, n)$.

A_4 Put $\alpha_2(n) = n$.

A_5 Put $\alpha_3(n) = n + 1$.

Definition 6.2. ($\mathcal{A}$-Sequence) (1) A sequence from $\mathbf{A_R}$, say $\{x_j\}$, is called an $\mathcal{A}$-sequence if $\{x_j\}_j \subset \mathbf{J_Z}$ or $\{x_j\}_j \subset \mathbf{J}$.

(2) A multiple sequence from $\mathbf{A_R}$, say $\{x_{\nu j}\}$, is called an $\mathcal{A}$-sequence if, for each ν, $\{x_{\nu j}\}_j \subset \mathbf{J_Z}$ or $\{x_{\nu j}\}_j \subset \mathbf{J}$.

Definition 6.3. ($\mathcal{A}$-Recursive Sequence) (1) A sequence of rational numbers, say $\{r_i\}$, is called $\mathcal{A}$-*recursive* if the following holds.

(1-a) $\{r_i\}$ is an $\mathcal{A}$-sequence.

(1-b) There are recursive functions α, β, γ such that

$$r_i = (-1)^{\alpha(i)} \frac{\gamma(i)}{\beta(i)}$$

where β never assumes value 0.

(2) A multiple sequence of rationals, say $\{r_{\nu i}\}$, is called $\mathcal{A}$-recursive if it is an $\mathcal{A}$-sequence and if it can be recursively represented as in (1-b) with recursive functions $\alpha(\nu, i), \beta(\nu, i)$ and $\gamma(\nu, i)$.

Remark 2. An effective enumeration of all rational numbers is *not* $\mathcal{A}$-recursive.

We will state some classical facts on $\mathcal{A}$.

Proposition 6.2. (Classical Properties of $\mathcal{A}$)

(1) Every real number can be represented as the limit of an $\mathcal{A}$-sequence of rational numbers.

(2) For each $k \in \mathbf{Z}$, $\{k\} \cup J_k \cup \{k+1\}$ is not complete (in $\mathcal{A}$).

(3) The space $\mathcal{A} = \langle \mathbf{A_R}, \mathcal{W} \rangle$ is not complete.

Definition 6.4. ($\mathcal{A}$-Computable Sequence in $\mathcal{A}$) (1) A sequence from $\mathbf{A_R}$, say $\{x_l\}$, is called $\mathcal{A}$-*computable* if there is an $\mathcal{A}$-*recursive* double sequence of rationals (Definition 6.3), say $\{r_{lj}\}$, and a recursive function δ satisfying

$$\forall n \forall l \forall j \geq \delta(n, l)(r_{lj} \in W_n(x_l)).$$

(2) A multiple sequence from $\mathbf{A_R}$, say $\{x_{\nu l}\}$, is called $\mathcal{A}$-computable if there is an $\mathcal{A}$-recursive sequence of rationals, say $\{r_{\nu l i}\}$ and a recursive function δ satisfying

$$\forall \nu \forall n \forall l \forall j \geq \delta(\nu, n, l)(r_{\nu l j} \in W_n(x_{\nu l})).$$

Proposition 6.3. (Closure under Effective Limit) If a double $\mathcal{A}$-computable sequence $\{x_{ij}\}$ converges effectively to a sequence $\{x_i\}$ as j tends to ∞ with respect to the topology $\mathcal{W}$, then the latter is an $\mathcal{A}$-computable sequence.

Proof: The proof of the corresponding property in Proposition 1 in Chapter 0 of [11] for $\mathbf{R}$ goes through, if we note that there the decidability of order relation $a < b$ for computable real numbers a and b is not used. We only have to check that the recursive sequence of rationals constructed there satisfies the property of being $\mathcal{A}$-sequence.

Proposition 6.4. ($\mathcal{A}$-Computable Elements)
(i) For a single real number, it is computable (in $\mathbf{R}$) if and only if it is $\mathcal{A}$-computable.
(ii) An $\mathcal{A}$-sequence is computable (in $\mathbf{R}$) if and only if it is $\mathcal{A}$-computable.
(iii) There is a sequence of real numbers (not an $\mathcal{A}$-sequence) which is computable in $\mathbf{R}$ but is not $\mathcal{A}$-computable.
(iv) An effective enumeration of all rational numbers is $\mathcal{A}$-computable. (Recall that it is not $\mathcal{A}$-recursive.)

Now, we can define $\mathcal{S}_{\mathcal{A}}$ as the family of $\mathcal{A}$-computable real sequences in the sense above.

Proposition 6.5. (Computability Structure on $\mathcal{A}$) (i) Let $\mathcal{S}_{\mathcal{A}}$ be the family of $\mathcal{A}$-computable sequences as defined in Definition 6.4. Then $\mathcal{S}_{\mathcal{A}}$ is a computability structure for the space $\mathcal{A}$.
(ii) An effective enumeration of all rational numbers is an effective separating set of $\mathcal{A}$ (cf. Definition 3.5).

Proof: (i) C1 and C2 are obvious. C3 is Proposition 6.3.
(ii) From the definition of $\mathcal{A}$-*computable sequences* in Definition 6.4 and (iv) above.

We will now apply the computability structure $\mathcal{S}_{\mathcal{A}}$ in order to demonstrate that some functions which have jump points at integers can be computable in $\mathcal{A}$.

Using the topology of $\mathcal{A}$, we can supply some examples and counter examples to notions of computability in Section 4.

Example 4. $\mathcal{S}_{\mathbf{Z}}$: Let $\mathcal{S}_{\mathbf{Z}}$ be the set of all recursive sequences of integers. It satisfies the axioms C1-C3 in Definition 3.3 with the $\mathcal{A}$-topology. The sequence $\{0, 1, -1, 2, -2, \cdots\}$ serves as an effective approximating set (Definition 3.4) for $\mathcal{S}_{\mathbf{Z}}$. This set is certainly not dense in $\mathbf{A}_{\mathbf{R}}$. This structure is relatively effectively complete (Definition 3.6) but, as stated in (iii) of Proposition 6.2, the whole space is not complete.

Example 5. $\mathcal{S}_{\mathbf{J}}$: Let $\mathcal{S}_{\mathbf{J}}$ be the family of sequences in $\mathcal{S}_{\mathcal{A}}$ which lie in $\mathbf{J}$. It satisfies the axioms C1-C3. A recursive enumeration of all the non-integer rationals will serve as its effective approximating set (Definition 3.4), which is not dense in the space $\mathbf{A}_{\mathbf{R}}$.

Proposition 6.6. (Uniform Computability in $\mathcal{A}$) Let f be a real-valued function on $\mathbf{A}_{\mathbf{R}}$. f is uniformly computable in $\mathcal{A}$ if and only if f preserves sequential computability and is uniformly equi-computable on all J_k for $k \in \mathbf{Z}$, that is, there is a recursive function, say β such that

$$\forall p \forall k \forall x, y \in \mathbf{J}_k (y \in W_{\beta(p)}(x) \Rightarrow |f(x) - f(y)| \leq \frac{1}{2^p}).$$

(See Definition 4.5 for uniform computable function.)

A simple example will be the integer-part function, also known as the Gaußian function.

Proposition 6.7. (Computability of the Gaußian Function) The Gaußian function is uniformly computable in $\mathcal{A}$.

Proof: It suffices to show the necessary condition in Proposition 6.6.

(Preservation of computability) Let $\{x_l\}$ be an $\mathcal{A}$-computable sequence, that is, there is an $\mathcal{A}$-recursive double sequence of rationals $\{r_{li}\}$ which converges effectively to $\{x_l\}$ in $\mathcal{A}$. Notice that the sequence $\{[r_{li}]\}$ is an $\mathcal{A}$-sequence from $\mathcal{J}_Z$ and is a recursive sequence of rational numbers (in $\mathbf{R}$).

Furthermore, it obviously holds that

$$\forall n \forall l \forall i (r_{li} \in W_n(x_l))$$

and hence $\{[x_l]\}$ is $\mathcal{A}$-computable.

(Equi-computability) Put $\beta(p) = 1$. Suppose $x \in J_k$ for a $k \in \mathbf{Z}$ and $y \in V_1^k(x)$. Then $[x] = [y] = k$, and hence $[x] - [y] = 0$. If $x \in \mathbf{J_Z}$ and $y \in V_1^{\mathbf{Z}}(x)$, then $y = x$. β trivially satisfies the requirement.

Proposition 6.8. (Sufficiency for Computability) Consider a real-valued function f on $\mathbf{A_R}$ which has the following properties.

(a) f preserves sequential computability.

(b) There is a recursive function κ such that

$$\forall k, p \in \mathbf{Z} \forall x, y \in \mathbf{J}_k (y \in W_{\kappa(k,p)}(x) \Rightarrow |f(x) - f(y)| \leq \frac{1}{2^p}).$$

Using the topology of $\mathcal{A}$ and applying Propositions 6.6 and 6.8, we can supply some examples and counter-examples to notions of computability in Section 4.

Example 6. Consider the computability structure $\mathcal{S}_{\mathcal{A}}$, and a function f on $\mathbf{A_R}$ such that $f(n) = n$ for $n \in \mathbf{Z}$ and $f(x) = \frac{1}{x-k}$ on $\mathbf{J}_k$. f is *computable* but *not uniformly computable* (Definitions 4.2 and 4.5).

Example 7. Consider the computability structure $\mathcal{S}_{\mathbf{J}}$ in Example 5 above. Let f be the function $f(x) = x$ defined on $\mathbf{A_R}$. Then f is *relatively computable* with respect to $\mathcal{S}_{\mathbf{J}}$ but *not computable* (Definitions 4.1 and 4.2).

This is because, for any sequence $\{e_l\} \in \mathcal{S}_{\mathbf{J}}$ and for every n, $V_n(e_l) \cap \mathbf{J_Z} = \phi$.

Example 8. Take the computability structure $\mathcal{S}_{\mathbf{Z}}$.

(1) Define $f(x) = [x + \frac{1}{2}]$. f is *relatively computable* with respect to $\mathcal{S}_{\mathbf{Z}}$ but *not continuous*. (See Note for Definition 4.1.)

(2) Define $f_n(x) = 1 - \frac{1}{n}$ if $x \in \mathbf{J_Z}$, and $f_n(x) = (-1)^n$ if $x \in \mathbf{J}$. Let $f(x) = 1$. The sequence of functions $\{f_n\}$ *relatively effectively* converges to f but does *not effectively* converge to f. (See Definition 4.3.)

Similarly to $\mathcal{A}$, we can change the topology of a subset of $\mathbf{R}$. Using this idea applied to the interval $[0, 1]$, we can further supply some examples and counter-examples.

Example 9. Let $X = [0,1]$, and let X_A be the set X regarded as the disjoint sum of $\{0\}$ and $(0,1]$. Define $V_l(0) = \{0\}$ and $V_l(x) = B(x, \frac{1}{2^l}) \cap (0,1]$ for $x \in (0,1]$. X_A with $\{V_l\}$ is an effective uniform topological space.

Consider the function $f(x) = \frac{1}{x}$ on $(0,1]$ and $f(0) = 0$. f is computable on X_A, but not uniformly computable (Definitions 4.2 and 4.5).

Example 10. Let us take the same space as in Example 9. Consider a sequence of functions $\{f_n\}$ defined by

$$f_n(x) = \begin{cases} x^{-1+\frac{1}{n}}, & x \in (0,1] \\ 0, & x = 0. \end{cases}$$

$\{f_n\}$ is an effectively equi-computable sequence of functions, effectively converging to the function f in Example 9 (An example of Proposition 4.3).

7 Partial Computable Functions and Future Work

Here we will briefly state how we view the computability problem of partial functions, and also some succeeding work we plan to take up.

A function such as $\tan x$ is defined on the real line except for some points. In the line of [11], computability of a function is generally discussed for total functions, but a function like $\tan x$ certainly possesses some computational attributes. Here we will take up such a subject.

One way of treating such a function in the theory of computability is to regard it as a point in a functional space such as a Banach space or Fréchet space. However, we would like to treat such a function not just as a point in a space but also as an input-output operation in considering computability.

For this purpose, we will extend our notion of computable functions to partial functions.

Definition 7.1. (Partial Computability) Let $\{e_k\}$ be an effective separating set in an effective uniform space with a computability structure, say $\langle X, \mathcal{V}, \mathcal{S}\rangle$. Let η and ξ be recursive functions and let $D = \bigcup_i V_{\eta(i)}(e_{\xi(i)})$ be a subset of X determined by η and ξ. Let f be a real-valued function defined on D. f is said to be *partial computable* if the following holds.

(i) f preserves sequential computability on D, that is, for any computable sequence $\{x_k\}$ in D, $\{f(x_k)\}$ is a computable sequence of real numbers.

(ii) f is computable on D in the sense that there is a recursive function $\delta(i,p)$ satisfying, for every p and every i, if $x \in V_{\eta(i)}(e_{\xi(i)})$, then for every y

$$y \in V_{\eta(i)}(e_{\xi(i)}) \cap V_{\delta(i,p)}(x) \Rightarrow |f(x) - f(y)| \leq \frac{1}{2^p}.$$

Example 11. Consider the real space with the Euclidean topology. $V_n(x) = \{y : |x - y| < \frac{1}{2^n}\}$ gives an effective uniform topology.

The family of computable sequences of real numbers (in $\mathbf{R}$) serves as a computability structure. Let $\{e_i\}$ be a recursive enumeration of rational numbers, which is an effective separating set.

Consider the tan function on the real line.

Notice that $\{\frac{2n+1}{2}\pi\}$, $n = 0, \pm1, \pm2, \cdots$, is a computable sequence of real numbers, which are not rationals, and tan is defined on $D = \bigcup_n (\frac{2n+1}{2}\pi, \frac{2n+3}{2}\pi)$.

Now define ξ to be the identity, so that it is recursive. η is defined as follows. For $i = 1, 2, \cdots$, find the m such that $\frac{2m+1}{2}\pi < e_i < \frac{2(m+1)+1}{2}\pi$, $m = 0, \pm1, \pm2, \cdots$. Put such a unique m as m_i. $\{m_i\}$ is a recursive sequence of integers.

Next, for each i and l, check if $\frac{2m_i+1}{2}\pi < e_i - \frac{1}{2^l}$ and $e_i + \frac{1}{2^l} < \frac{2(m_i+1)+1}{2}\pi$. Define $\eta(i)$ to be the least such l for i. η is recursive.

The D above can be written as $D = \bigcup_i V_{\eta(i)}(e_i)$.

Sequential computability of tan is known.

For computability on D, consider the function on each compact interval $I_i = [e_i - \frac{1}{\eta(i)}, e_i + \frac{1}{\eta(i)}]$. Then we can find a recursive function $\delta(i, p)$ which serves as a modulus of continuity depending on i.

So, the function tan is partial computable.

Example 12. Put $f(x) = \sin(\frac{1}{x})$ on $\mathbf{R} - \{0\}$. We can show that f is partial computable on this domain in a manner similar to Example 11.

In concluding this article, we will list some further topics which we would like to work on in the future.

Foreseeable Subjects. 1. The effective inter-relations between a uniform topology and the corresponding metric.

2. More effective properties on an effective uniform topological space, especially some properties of effective subsets.

3. Application of our theory to Walsh analysis, especially in relation to the results in [7] and [8].

4. Foundational problems, especially with Type 2 Turing machine by Weih/-rauch and his colleagues.

References

1. V. Brattka, *Recursive characterization of computable real-valued functions and relations*, Theoretical Computer Science 162(1996), 45-77.
2. V. Brattka, *A Stability Theorem for Recursive Analysis*,Proceedings of CCA'98 (Brno), Informatik Berichte 235-8/1998(FernUniversit"at), Computability and Complexity in Analysis, ed. by K-I. Ko et. al,(1998), 1-16.
3. V. Brattka and P. Hertling, *Feasible Real Random Access Machines*, Journal of Complexity,14(1998),490-526.
4. N.J.Fine, *On the Walsh Functions*, Trans. Amer. Math. Soc.,65(1949), 372-414.
5. P.Hertling, *Effectivity and effective continuity of functions between computable metric spaces*, Combinatorics, Complexity and Logic, Proceedings of DMTCS'96(1997), 264-275.
6. Y.Kawada and Y.Mimura, *Outline of Modern Mathematics* (in Japanese),Iwanami Shoten, 1965.
7. T.Mori,*On the computability of Walsh functions*, to appear in TCS.
8. T.Mori, *Computabilities of Fine–continuous functions*, Computability and Complexity in Analysis, Proceedings of CCA2000(2000), 299-318.

9. T.Mori,Y.Tsujii and M.Yasugi, *Computability structure on metric spaces*, Combinatorics, Complexity and Logic (Proceedings of DMTCS'96), ed. by Bridges et.al,Springer(1996),351-362.
10. J.Nagata, *Modern General Topology*, 1968, North-Holland Publ.Co., Amsterdam
11. M.B.Pour-El and J.I.Richards, *Computability in Analysis and Physics*, Perspectives in Mathematical Logic, Springer-Verlag, 1989.
12. F. Shipp, W.R.Wade and P.Simon, *Walsh Series*, Adam Hilger, 1990.
13. J.V.Tucker and J.I.Zucker, *Computable functions on stream algebras*, Proc. NATO Advanced Study Institute Int. Summer School at Marktoberdorf on Proof and computation (ed. H. Schwichtenberg), Series F: Computer and Systems Sciences, Springer-Verlag, Berlin(1994),341-382.
14. J.V.Tucker and J.I.Zucker, *Computable functions and semicomputable sets on many-sorted semicomputable sets on many-sorted algebras*, Handbook of Logic in Computer Science,.vol.5 (ed. S. Abramski et al), Oxford University Press (1998).
15. M.Washihara, *Computability and Fréchet space*, Mathematica Japonica, 42(1995),1-13.
16. M.Washihara, *Computability and tempered distributions*, ibid.,50,1(1999),1-7.
17. M.Washihara and M.Yasugi, *Computability and metrics in a Fréchet space*, ibid., 43(1996), 1-13
18. K.Weihrauch, *A foundation for computable analysis*, Combinatorics, Complexity, and Logic(Proceedings of DMTCS'96),ed. by D.S.Bridges et al,(1996), 66-89.
19. K.Weihrauch and X. Zheng, *Computability on continuous, lower semi-continuous and upper semi-continuous real functions*, Computing and Combinatorics, ed. by T.Jiang and D.T.Lee,LNCS,1276(1997),Springer,166-175.
20. M. Yasugi, V. Brattka and M. Washihara, *Computability problems of the Gaußian function*, Manuscript (2001): available at `http://www.kyoto-su.ac.jp/~yasugi/Recent`.
21. M.Yasugi, T.Mori and Y.Tsujii, *Effective Properties of Sets and Functions in Metric Spaces with Computability Structure*, Theoretical Computer Science, 219(1999), 467-486.
22. M.Yasugi and M.Washihara, *Computability structures in analysis*, Sugaku Expositions (AMS), 13(2000), 215-235.
23. M.Yasugi and M.Washihara, *A note on Rademacher functions and computability*, manuscript: available at `http://www.kyoto-su.ac.jp/~yasugi/Recent`.

On Computable Metric Spaces Tietze–Urysohn Extension Is Computable

Klaus Weihrauch

Theoretische Informatik I, FernUniversität, 58084 Hagen, Germany
Klaus.Weihrauch@FernUni-Hagen.de
http://www.informatik.fernuni-hagen.de/thi1/klaus.weihrauch/

Abstract. In this paper we prove computable versions of Urysohn's lemma and the Tietze–Urysohn extension theorem for computable metric spaces. We use the TTE approach to computable analysis [KW85,Wei00] where objects are represented by finite or infinite sequences of symbols and computations transform sequences of symbols to sequences of symbols. The theorems hold for standard representations of the metric space, the set of real numbers, the set of closed subsets and the set of continuous functions. We show that there are *computable procedures* determining the continuous functions from the initial data (closed sets, continuous functions). The paper generalizes results by Yasugi, Mori and Tsujii [YMT99] in two ways: (1) The Tietze–Urysohn extension applies not only to "strictly effectively σ-compact co-r.e." sets but to all co-r.e. closed sets. (2) Not only computable functions exist for computable sets and functions, respectively, but there are *computable procedures* which determine continuous functions from arbitrary closed sets and continuous functions, respectively. These procedures, however, are not extensional on the names under consideration, and so they induce merely multi-valued computable functions on the objects.

1 Introduction

Let (M, d) be a metric space. Urysohn's lemma and the Tietze–Urysohn extension theorem for metric spaces can be formulated as follows [Die60,Eng89].

Theorem 1 (Urysohn's Lemma). *For any pair A, B of disjoint closed subsets of M there exists a continuous function $f : M \to \mathbb{R}$ such that $f(x) = 0$ for $x \in A$, $f(x) = 1$ for $x \in B$ and $0 < f(x) < 1$ for $x \notin A \cup B$.*

Theorem 2 (Tietze–Urysohn Extension Theorem). *Let $A \subseteq M$ be closed and let $f : A \to \mathbb{R}$ be continuous. Then there exists a continuous function $g : M \to \mathbb{R}$ such that $f(x) = g(x)$ for $x \in A$, and $\inf_{x\in A} f(x) = \inf_{x\in M} g(x)$ and $\sup_{x\in A} f(x) = \sup_{x\in M} g(x)$, if $A \neq \emptyset$.*

Some computable versions of the theorems have been proved in the framework of the the Pour-El/Richards approach to computable analysis [PER89], in which

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 357–368, 2001.

computable functions are continuous. Q. Zhou [Zho96] uses a special computable version of Urysohn's lemma (Theorem 3) to prove a computable Tietze–Urysohn extension theorem (Theorem 4).

Theorem 3 (Zhou). *Let I be a computable rectangle in $\mathbb{R}^q$. For any integer v let $A, B \subseteq I$ be finite unions of closed cubes in the "vth grid" such that $A \cap B = \emptyset$. Then there exists a computable real function $f : I \to \mathbb{R}$ such that $f(x) = 0$ for $x \in A$, $f(x) = 1$ for $x \in B$ and $0 < f(x) < 1$ for $x \notin A \cup B$.*

Theorem 4 (Zhou). *Let I be a computable rectangle in $\mathbb{R}^q$, let K be a recursive closed subset of I, and let $g : K \to \mathbb{R}$ be a computable function. Then there exists a computable function $f : I \to \mathbb{R}$ such that $f(x) = g(x)$ for all $x \in K$.*

(Zhou's generalization to $g : K \to \mathbb{R}^p$ is straightforward.) A more general computable version of Urysohn's lemma for metric spaces with "computability structure" has been proved by Mori/Tsujii/Yasugi [MTY97].

Theorem 5 (Mori/Tsujii/Yasugi). *For any disjoint pair A, B of co-r.e. closed subsets of an "effectively compact metric space" M there exists a computable function $f : M \to \mathbb{R}$ such that $f(x) = 0$ for $x \in A$ and $f(x) = 1$ for $x \in B$.*

In [YMT99] the same authors have proved a more uniform version of the above theorem for computable sequences of co-r.e. sets and a computable version of the Tietze–Urysohn extension theorem:

Theorem 6 (Yasugi/Mori/Tsujii). *Let A be a "strictly effectively σ-compact" metric space M and let $g : A \to \mathbb{R}$ be computable. Then the function g has a computable extension $f : M \to \mathbb{R}$.*

A more general version of Urysohn's lemma for Euclidean space is as follows ([Wei00], Theorem 6.2.10).

Theorem 7 (Weihrauch).

1. *There is a $(\psi^n, \psi^n, \delta^{n1}_{\to})$-computable function $G :\subseteq \mathcal{A} \times \mathcal{A} \to \mathrm{C}(\mathbb{R}^n)$ mapping every disjoint pair $A, B \subseteq \mathbb{R}^n$ of non-empty closed sets to a continuous function $f : \mathbb{R}^n \to \mathbb{R}$ such that*
   $f(x) = 0$ for $x \in A$, $f(x) = 1$ for $x \in B$ and $0 < f(x) < 1$, otherwise.
2. *The multi-valued function $F :\subseteq \mathcal{A} \times \mathcal{A} \rightrightarrows \mathrm{C}(\mathbb{R}^n)$ mapping every disjoint pair $A, B \subseteq \mathbb{R}^n$ of closed sets to continuous functions $f : \mathbb{R}^n \to \mathbb{R}$ such that $f(x) = 0$ for $x \in A$, $f(x) = 1$ for $x \in B$ and $0 < f(x) < 1$, otherwise, is $(\psi^n_>, \psi^n_>, \delta^{n1}_{\to})$-computable.*

In contrast to Theorems 3 to 6 which deal with computable closed sets and computable real valued functions using the Pour-El/Richards approach to computable analysis [PER89], this last theorem formulated in the framework of TTE (Type-2 Theory of Effectivity) [KW85,Wei00] is of different type: there

are *computable operators* mapping arbitrary disjoint pairs of closed sets to continuous functions. In particular, they map computable elements to computable ones. Since the ψ^n-computable closed subsets of $\mathbb{R}^n$ are the recursive closed sets ([Wei00], Def. 5.1.1; [Zho96]) and the $\delta^{n1}_{\to}$-computable real functions are the computable functions ([Wei00], Lemma 6.1.2; [PER89]), the first part generalizes Theorem 3. And since the $\psi^n_>$-computable closed subsets of $\mathbb{R}^n$ are the co-r.e. closed sets ([Wei00], Def. 5.1.1), the second part generalizes Theorem 5 for the Euclidean space.

Using the framework of TTE, in this paper we first generalize Theorem 7 to "computable metric spaces" [Wei93] [Wei00] and then apply it to prove a computable operator version of the Tietze–Urysohn extension theorem for computable metric spaces, which generalizes Theorem 6 in two ways: (1) No compactness restriction is needed. (2) There is a *computable multi-valued function* acting on arbitrary continuous functions with closed domains. Neither in the second generalized version of Urysohn's lemma nor in the generalized version of the Tietze–Urysohn theorem "multi-valued" can be replaced by "single-valued". The theorems are proved for "canonical representations" of the sets under consideration.

2 Computability on Computable Metric Spaces

In this paper we apply the framework of TTE, Type-2 Theory of Effectivity. First we introduce some notations and representations we will use later. For details the reader is referred to [Wei00].

Let Σ be a sufficiently large finite alphabet of symbols and let Σ^* and Σ^ω be the set of finite and infinite, respectively, sequences of symbols. The length of a word $w \in \Sigma^\omega$ is denoted by $|w|$. By $\langle\ ,\ \rangle$ we denote standard pairing functions on the finite or infinite sequences, respectively ([Wei00], Def. 2.1.7). A function $f :\subseteq \Sigma^\omega \to \Sigma^\omega$ is computable, if there is a machine (e.g., a "Type-2 machine") which for any $p \in \mathrm{dom}(f)$ computes forever reading p from the left to the right and writing $f(p)$ one-way from the left to the right. Every computable function is continuous w.r.t. the Cantor topology τ_C on Σ^ω.

In TTE, infinite sequences $p \in \Sigma^\omega$ are used as "names" of abstract objects and computations are performed on names. A representation of a set M is a surjection $\delta :\subseteq \Sigma^\omega \to M$. If $\delta' :\subseteq \Sigma^\omega \to M'$ is another representation, a function $f :\subseteq M \to M'$ is (δ, δ')-computable (-continuous), iff there is a computable (continuous) realizing function $h :\subseteq \Sigma^\omega \to \Sigma^\omega$, such that $f \circ \delta(p) = \delta' \circ h(p)$ for $\delta(p) \in \mathrm{dom}(f)$. A multi-valued function $F :\subseteq M \rightrightarrows M'$ is (δ, δ')-computable (-continuous), iff there is a computable (continuous) realizing function $h :\subseteq \Sigma^\omega \to \Sigma^\omega$, such that $\delta' \circ h(p) \in F \circ \delta(p)$ for $\delta(p) \in \mathrm{dom}(F)$. Generalizations to functions with several arguments are straightforward.

Let $\nu_{\mathbb{Q}} :\subseteq \Sigma^* \to \mathbb{Q}$ and $\mathrm{I} :\subseteq \Sigma^* \to \mathrm{Int}$ be some standard notations of the rational numbers and the open intervals on $\mathbb{R}$ with rational endpoints, respectively. As our standard representation of the real numbers we consider the function $\rho :\subseteq \Sigma^\omega \to \mathbb{R}$ defined by: $\rho(p) = x$, iff $p = w_0\#w_1\#\ldots$, $\overline{\mathrm{I}}(w_{n+1}) \subseteq \mathrm{I}(w_n)$

and $\{x\} = \bigcap_{n\in\mathbb{N}} \mathrm{I}(w_n)$. It is equivalent to the standard Cauchy representation ([Wei00], Def. 4.1.5). Computable metric spaces are introduced in [Wei00], Section 8.1, and the somewhat more general semi-computable metric spaces in [Wei93]).

Definition 8. *A computable metric space is a quadruple* $\mathbf{M} = (M, d, D, \alpha)$*, where* (M, D) *is a separable metric space,* D *is a dense countable subset of* M *and* $\alpha :\subseteq \Sigma^* \to D$ *is a notation of* D *such that*

$$\nu_{\mathbb{Q}}(t) < d(\alpha(u), \alpha(v)) < \nu_{\mathbb{Q}}(w) \quad \text{is r.e. in} \quad t, u, v, w, \in \Sigma^*. \tag{1}$$

In the following let $\mathbf{M} = (M, d, D, \alpha)$ be a fixed computable metric space.

Property (1) means that $d_{D\times D}$ is (α, α, ρ)-computable [Wei00]. Notice that in [Wei93], Property (1) is replaced by the weaker condition

$$d(\alpha(u), \alpha(v)) < \nu_{\mathbb{Q}}(w) \quad \text{is r.e. in} \quad u, v, w \in \Sigma^* \ . \tag{2}$$

As a consequence of (1) or (2), $\mathrm{dom}(\alpha)$ is r.e., and so there is a notation α' of D such that $\alpha \equiv \alpha'$ and $\mathrm{dom}(\alpha')$ is recursive. Therefore we may assume without affecting the computability results below that $\mathrm{dom}(\alpha)$ is recursive.

Define a notation $\mathrm{J} :\subseteq \Sigma^* \to \mathcal{B}$ of the set of open balls in M with center in D and rational radius by

$$\mathrm{J}\langle u, v\rangle := B(\alpha(u), \nu_{\mathbb{Q}}(v)) \quad \text{for} \quad u \in \mathrm{dom}(\alpha) \ \text{ and } \ \nu_{\mathbb{Q}}(v) > 0 \tag{3}$$

and $\mathrm{J}(w)$ does not exist, otherwise. We may assume that # does not occur in w for $w \in \mathrm{dom}(\mathrm{J})$ and that the empty word is not in $\mathrm{dom}(\mathrm{J})$. Since ball inclusion $\mathrm{J}(u) \subseteq \mathrm{J}(v)$ can depend crucially on the internal structure of the metric space and may be not even r.e., we consider *formal balls* with *formal radius* and *formal inclusion* [WS81,Wei93]. For $\langle u, v\rangle, \langle u', v'\rangle \in \mathrm{dom}(\mathrm{J})$ we define the formal radius by $\mathrm{rad}\langle u, v\rangle := \nu_{\mathbb{Q}}(v)$ which is an upper bound of the metric radius of the ball $\mathrm{J}\langle u, v\rangle \subseteq M$, and the formal inclusion relation $\prec \subseteq \mathrm{dom}(\mathrm{J}) \times \mathrm{dom}(\mathrm{J})$ by

$$\langle u, v\rangle \ \prec \ \langle u', v'\rangle \iff d(\alpha(u), \alpha(u')) + \nu_{\mathbb{Q}}(v) < \nu_{\mathbb{Q}}(v') \ . \tag{4}$$

Obviously, the relation $\prec$ is an r.e. subset of $\Sigma^* \times \Sigma^*$ and

$$s \prec t \ \text{ implies } \ \overline{\mathrm{J}}(s) \subseteq \mathrm{J}(t).$$

Various representations of $\mathbb{R}^n$, the set of closed subsets of $\mathbb{R}^n$, and of sets of continuous functions $f :\subseteq \mathbb{R}^n \to \mathbb{R}$ are introduced and compared in [BW99,Wei00]. We generalize some of them from $\mathbb{R}^n$ to M.

Definition 9. *Define a representation* $\delta :\in \Sigma^\omega \to M$ *as follows:* $\delta(p) = x$*, iff there are* $u_0, u_1, u_2 \in \mathrm{dom}(\mathrm{J})$ *such that*

$$p = u_o\#u_1\#\ldots, \quad (\forall n)\ u_{n+1} \prec u_n, \quad \lim_n \mathrm{rad}(u_n) = 0 \quad \text{and} \quad \{x\} = \bigcap_n \mathrm{I}(u_n)\,.$$

The representation δ is equivalent to the Cauchy representation of M ([Wei93]; [Wei00], Section 8.1). In particular, it is admissible ([KW85]; [Wei00], Section 3.2). The distance function $d : M \times M \to \mathbb{R}$ is (δ, δ, ρ)-computable.

Definition 10. *Define a representations* $\psi_> :\subseteq \Sigma^\omega \to \mathcal{A}$ *of the set* $\mathcal{A}$ *of the closed subsets of* M *as follows:*

$$w \in \mathrm{dom(J)},\ \textit{if}\ p \in \mathrm{dom}(\psi_>)\ \ \textit{and}\ \ \#w\#\ \ \textit{is a subword of}\ \ p,\ \ \textit{and}$$
$$\psi_>(p) := M \setminus \bigcup \big\{ \mathrm{J}(w) \mid \#w\#\ \textit{is a subword of}\ \ p \big\}.$$

A $\psi_>$-name p of a closed set $A \in M$ "enumerates" its open complement. The representation $\psi_>$ generalizes $\psi_>^{\mathrm{en}}$ from [Wei00], Def 5.1.9, and δ_{union} from [BW99].

As a generalization of the "admissible Gödel numbering" $\phi : \mathbb{N} \to P^{(1)}$, in TTE there is a representation $\eta^{\omega\omega} : \Sigma^\omega \to F^{\omega\omega}$ satisfying the universal Turing machine theorem ("utm-theorem") and the computable smn-theorem where

$$F^{\omega\omega} = \{ f :\subseteq \Sigma^\omega \to \Sigma^\omega \mid f\ \ \text{continuous and}\ \ \mathrm{dom}(f)\ \ \text{is}\ \ G_\delta \}\,.$$

Every continuous function $f :\subseteq \Sigma^\omega \to \Sigma^\omega$ has an extension in $F^{\omega\omega}$ ([Wei00], Section 2.3). We introduce "canonical" representations of the set $\mathrm{C}(M)$ of continuous functions $f : M \to \mathbb{R}$ and of the set $\mathrm{C_c}(M)$ of continuous partial functions $f :\subseteq M \to \mathbb{R}$ with closed domain.

Definition 11. *1. Define a representation* $\delta_\to :\subseteq \Sigma^\omega \to \mathrm{C}(M)$ *by*

$$\delta_\to(p) = f \iff \eta_p^{\omega\omega}\ \ \textit{is a}\ \ (\delta, \rho)\textit{-realization of}\ \ f\,.$$

2. Define a representation $\delta_{\mathrm{c}} :\subseteq \Sigma^\omega \to \mathrm{C_c}(M)$ *by*

$$\delta_{\mathrm{c}}\langle p, q\rangle = f \iff \eta_p^{\omega\omega}\ \ \textit{is a}\ \ (\delta, \rho)\textit{-realization of}\ \ f\ \ \textit{and}\ \ \mathrm{dom}(f) = \psi_>(q)\,.$$

Since the representation δ is admissible, by the "main theorem" ([Wei00], Theorem 3.2.11), $\delta_\to$ is a representation of $\mathrm{C}(M)$ and δ_{c} is a representation of $\mathrm{C_c}(M)$. The representation $\delta_\to$ generalizes the representation $\delta_\to^{\mathbb{R}^n}$ of the continuous real functions and $\delta_\to = [\delta \to \rho]_M$ ([Wei00], Definitions 6.1.1, 3.3.13). The functions apply : $(f, x) \mapsto f(x)$ for $f \in \mathrm{C}(M)$ or $f \in \mathrm{C_c}(M)$ are $(\delta_\to, \delta, \rho)$-computable or $(\delta_{\mathrm{c}}, \delta, \rho)$-computable, respectively ([Wei00], Lemma 3.3.14). Obviously, $\delta_\to$ is reducible to δ_{c}, $\delta_\to \le \delta_{\mathrm{c}}$ (see [Wei00], Definition 6.1.1 and Section 3.3).

Finally, we introduce the distance representation ψ^{dist} of the set of $\mathcal{A}_0$ of the non-empty closed subsets of M.

Definition 12. *Define a representation* $\psi^{\mathrm{dist}} :\subseteq \Sigma^\omega \to \mathcal{A}_0$ *by*

$$\psi^{\mathrm{dist}}(p) = A \iff \delta_\to(p) = d_A$$

where $d_A(x) := d(x, A) = \inf_{y \in A} d(x, y)$.

The representation ψ^{dist} generalizes ψ^{dist} from [Wei00], Definition 5.1.6, and $\delta_{\mathrm{dist}}^{=}$ from [BW99] (which represent also the empty set). An easy proof shows that ψ^{dist} is reducible to $\psi_>$, $\psi^{\mathrm{dist}} \le \psi_>$. But in general, $\psi_>$ is not even continuously reducible to ψ^{dist}, $\psi_> \not\le_t \psi^{\mathrm{dist}}$ (for example, consider the Euclidean space ([BW99]; [Wei00], Section 5.1).

3 Computable Tietze–Urysohn Extension

Our first computable version of Urysohn's lemma generalizes Theorem 7.1.

Theorem 13 (First Computable Urysohn Lemma). *There is a* $(\psi^{\mathrm{dist}}, \psi^{\mathrm{dist}}, \delta_{\to})$*-computable function* $U :\subseteq \mathcal{A}_0 \times \mathcal{A}_0 \to \mathrm{C}(M)$ *such that for any disjoint pair* (A, B) *of non-empty closed sets* $U(A,B)(x) = 0$ *for* $x \in A$*,* $U(A,B)(x) = 1$ *for* $x \in B$*, and* $0 < U(A,B)(x) < 1$*, otherwise.*

Proof: Define $U(A,B)(x) := d_A(x)/(d_A(x) + d_B(x))$. Then for closed disjoint non-empty sets A, B, the function $U(A,B)$ satisfies the Urysohn conditions. It remains to show that U is computable. For $A \in \mathcal{A}_0$ and $x \in M$ define $F(A,x) := d_A(x)$. We show that F is $(\psi^{\mathrm{dist}}, \delta, \rho)$-computable. By Definitions 11 and 12 we obtain

$$F(\psi^{\mathrm{dist}}(p), \delta(q)) = d_{\psi^{\mathrm{dist}}(p)} \circ \delta(q) = \delta_{\to}(p) \circ \delta(q) = \rho \circ \eta_p^{\omega\omega}(q) = \rho \circ u(p,q)\,,$$

where u is the computable universal function of $\eta^{\omega\omega}$, and so F is $(\psi^{\mathrm{dist}}, \delta, \rho)$-computable. Since addition and division on real numbers are computable (w.r.t. ρ), the function G, $G(A,B,x) := U(A,B)(x)$ is $(\psi^{\mathrm{dist}}, \psi^{\mathrm{dist}}, \delta, \rho)$-computable. By the type conversion theorem ([Wei00], Theorem 3.3.15), the function U is $(\psi^{\mathrm{dist}}, \psi^{\mathrm{dist}}, [\delta \to \rho]_M)$-computable, that is, $(\psi^{\mathrm{dist}}, \psi^{\mathrm{dist}}, \delta_{\to})$-computable. □

Since $\psi_> \not\leq_t \psi^{\mathrm{dist}}$ in general, for an input of a computation $\psi_>$-names are less useful than ψ^{dist}-names. Nevertheless, the "computationally available information" is sufficient to compute a name of an Urysohn function, although in this case there is no longer an extensional function on the names, the computed function is merely multi-valued, i.e. we must admit that equivalent names of two closed sets are mapped to different Urysohn functions. We prepare the proof of the theorem by a lemma.

Lemma 14. *1. The multi-valued function* $F : \mathcal{A} \rightrightarrows \mathrm{C}(M)$*,*

$$F(A) := \big\{ f \in \mathrm{C}(M) \mid A = f^{-1}[\{0\}] \big\},$$

is $(\psi_>, \delta_{\to})$*-computable.*

2. In general, there is no $(\psi_>, \delta_{\to})$*-continuous (single-valued) function* $F_1 : \mathcal{A} \to \mathrm{C}(M)$ *such that* $A = F_1(A)^{-1}[\{0\}]$*.*

Proof: 1. For each $\langle u, v\rangle \in \mathrm{dom}(\mathrm{J})$ define the cone function $c\langle u, v\rangle : M \to \mathbb{R}$ by $c\langle u, v\rangle(x) := \max(0, (\nu_{\mathbb{Q}}(v) - d(\alpha(u), x))/\nu_{\mathbb{Q}}(v))$ with values from $(0; 1]$ in the open ball $\mathrm{J}\langle u, v\rangle$ and value 0 otherwise. For every $p \in \mathrm{dom}(\psi_>)$ and $i \in \mathbb{N}$ define a function $f_{pi} : M \to \mathbb{R}$ by

$$f_{pi} := \begin{cases} 2^{-i} \cdot c(w) & \text{if } \#w\#, w \in \mathrm{dom}(\mathrm{J}), \text{ is the suffix of the first } i \text{ symbols of } p \\ 0 & \text{otherwise.} \end{cases}$$

The function $(p, i, x) \mapsto f_{pi}(x)$ is $(\mathrm{id}_{\Sigma^\omega}, \nu_{\mathbb{N}}, \delta, \rho)$-computable. Let $f_p := \sum_i f_{pi}$. Then $(p, x) \mapsto f_p(x)$ is $(\mathrm{id}_{\Sigma^\omega}, \delta, \rho)$-computable. By the type conversion theorem

([Wei00], Theorem 3.3.15), $p \mapsto f_p$ is $(\mathrm{id}_{\Sigma^\omega}, [\delta \to \rho]_M)$-computable, that is, $(\mathrm{id}_{\Sigma^\omega}, \delta_\to)$-computable. Since for every $p \in \mathrm{dom}(\psi_>)$, $\psi_>(p) = f_p^{-1}[\{0\}]$, the multi-valued function F is $(\psi_>, \delta_\to)$-computable.

2. For a counter-example consider the computable metric space $(\mathbb{R}, d, \mathbb{Q}, \nu_\mathbb{Q})$ with Euclidean distance. Suppose, there is a $(\psi_>, \delta_\to)$-continuous function $F_1 : \mathcal{A} \to \mathrm{C}(M)$ such that $A = F_1(A)^{-1}[\{0\}]$ for all closed $A \subseteq \mathbb{R}$. Since apply : $(f, x) \mapsto f(x)$ is $(\delta_\to, \delta, \rho)$-continuous and $\psi_>$ is $(\mathrm{id}_{\Sigma^\omega}, \psi)$-continuous, the function $p \mapsto F_1 \circ \psi_>(p)(0)$ is $(\mathrm{id}_{\Sigma^\omega}, \rho)$-continuous, and so by the main theorem ([Wei00], Theorem 3.2.11), $(\tau_C, \tau_\mathbb{R})$-continuous, where τ_C is the Cantor topology on Σ^ω, since $\mathrm{id}_{\Sigma^\omega}$ and ρ are admissible. Let $A_0 := \mathbb{R} \setminus (-1; 1)$, $\mathrm{J}(v) = (-1; 1)$ and for all n, $p_n := 0^n\#v\#000\ldots$. Then by (sequential) continuity,
$0 = F_1(\mathbb{R})(0) = F_1 \circ \psi_>(000\ldots)(0) = F_1 \circ \psi_>(\lim_n p_n)(0)$
$= \lim_n F_1 \circ \psi_>(p_n)(0) = \lim_n F_1(A_0)(0) = F_1(A_0)(0) > 0$ (contradiction). □

The following computable version of Urysohn's lemma generalizes Theorem 6 and Theorem 7.2.

Theorem 15 (Second Computable Urysohn Lemma).

1. *The multi-valued function $U :\subseteq \mathcal{A} \times \mathcal{A} \rightrightarrows \mathrm{C}(M)$ mapping every disjoint pair of closed subsets of M to continuous functions $f : M \to \mathbb{R}$ such that $f(x) = 0$ for $x \in A$, $f(x) = 1$ for $x \in B$ and $0 < f(x) < 1$, otherwise, is $(\psi_>, \psi_>, \delta_\to)$-computable.*
2. *In general, there is no $(\psi_>, \psi_>, \delta_\to)$-continuous (single-valued !) function $U_1 :\subseteq \mathcal{A} \times \mathcal{A} \to \mathrm{C}(M)$ such that for disjoint closed sets A, B, $f := U_1(A, B)$ has the property $f(x) = 0$ for $x \in A$ and $f(x) = 1$ for $x \in B$.*

Notice that in Part 2 we exclude not only functions with $0 < f(x) < 1$ for $x \notin A \cup B$.

Proof: 1. By Lemma 14 there is an $(\mathrm{id}_{\Sigma^\omega}, \delta_\to)$-computable function H such that $H(p)^{-1}[\{0\}] = \psi_>(p)$. Define a function $V :\subseteq \Sigma^\omega \times \Sigma^\omega \times M \to \mathbb{R}$ by

$$V(p, q, x) := \frac{H(p)(x)}{H(p)(x) + H(q)(x)} .$$

Since the apply function $(f, x) \mapsto f(x)$ for $\delta_\to$ is $(\delta_\to, \delta, \rho)$-computable, the function V is $(\mathrm{id}_{\Sigma^\omega}, \mathrm{id}_{\Sigma^\omega}, \delta, \rho)$-computable, and so by the theorem on type conversion ([Wei00], Theorem 3.3.15), the function $U' : (p, q) \mapsto f$ where $f(x) = V(p, q, x)$ is $(\mathrm{id}_{\Sigma^\omega}, \mathrm{id}_{\Sigma^\omega}, \delta_\to)$-computable. If $\psi_>(p) = A$, $\psi_>(q) = B$ and $A \cap B = \emptyset$, then $U'(p, q)$ is an Urysohn function for A, B. Therefore the multi-valued function U is $(\psi_>, \psi_>, \delta_\to)$-computable.

2. (cf. the proof of Lemma 14.2) For a counter-example consider the computable metric space $(\mathbb{R}, d, \mathbb{Q}, \nu_\mathbb{Q})$ with Euclidean distance. Suppose, there is a $(\psi_>, \psi_>, \delta_\to)$-continuous function U_1 such that $U_1(A, B)$ is an Urysohn function for all disjoint pairs A, B of closed sets. Since apply : $(f, x) \mapsto f(x)$ is $(\delta_\to, \delta, \rho)$-continuous and $\psi_>$ is $(\mathrm{id}_{\Sigma^\omega}, \psi)$-continuous, the function $(p, q) \mapsto U_1(\psi_>(p), \psi_>(q))(0)$ is $(\mathrm{id}_{\Sigma^\omega}, \mathrm{id}_{\Sigma^\omega}, \rho)$-continuous, and so by the main theorem

([Wei00], Theorem 3.2.11), $(\tau_C, \tau_C, \tau_{\mathbb{R}})$-continuous, where τ_C is the Cantor topology on Σ^ω, since $\mathrm{id}_{\Sigma^\omega}$ and ρ are admissible.

Consider the case $U_1(\emptyset,\emptyset)(0) > 0$. There are $w_0, w_1, \ldots \in \Sigma^*$ and $p \in \Sigma^\omega$ such that for $q := \#w_0\#w_1\#\ldots$, $\psi_>(p) = \emptyset$ and $\psi_>(q) = [-1;1]$. For $n \in \mathbb{N}$ let $q_n := \#w_0\#w_1\#\ldots\#w_n\#\#p$. Then by (sequential) continuity,
$0 < U_1(\psi_>(p), \psi_>(p))(0) = \lim_n U_1(\psi_>(q_n), \psi_>(p))(0)$
$= U_1(\psi_>(\lim_n q_n), \psi_>(p))(0) = U_1(\psi_>(q), \psi_>(p))(0) = 0$
(contradiction). The case $U_1(\emptyset,\emptyset)(0) < 1$ can be treated similarly. □

Since a computable function maps computable elements to computable ones, we obtain the following less effective corollary.

Corollary 16. *For every disjoint pair A, B of co-r.e. closed subsets of M, there is a computable function $f : M \to \mathbb{R}$ such that $f(x) = 0$ for $x \in A$, $f(x) = 1$ for $x \in B$ and $0 < f(x) < 1$, otherwise*

Notice that the corresponding corollary of Theorem 13 follows from Corollary 16.

The next lemmas prepare the proof of the computable Tietze–Urysohn extension theorem. A closed subset $D \subseteq \mathbb{R}$ is co-r.e., iff $\{w \in \mathrm{dom(I)} \mid \overline{\mathrm{I}}(w) \subseteq \mathbb{R} \setminus D\}$ is r.e. ([BW99]; [Wei00], Definition 5.1.1).

Lemma 17. *For every closed co-r.e. set $D \subseteq \mathbb{R}$, the function $H : f \mapsto f^{-1}[D]$ ($f \in \mathrm{C_c}(\mathbb{R})$) is $(\delta_\mathrm{c}, \psi_>)$-computable.*

Proof: Since the formal ball inclusion $\prec$ is r.e., the set of all initial parts $u_0\#u_1\#\ldots\#u_k\#$ of δ-names (see Definition 9) is r.e.. Let $w_0, w_1, \ldots$ be a computable list of this set. Since the set D is co-r.e., there is a computable list $v_0, v_1, \ldots$ of all words $w \in \mathrm{dom(I)}$ such that $\overline{\mathrm{I}}(w) \subseteq \mathbb{R} \setminus D$. Let N be a Type-2 machine ([Wei00], Section 2.1) computing the universal function $(p, q) \mapsto \eta_p^{\omega\omega}(q)$. There is a Type-2 machine M which on input $\langle p, q\rangle \in \mathrm{dom}(\delta_\mathrm{c})$ works in stages $k = 0, 1, 2, \ldots$ as follows.

Stage k: (1) M simulates $|w_k|$ steps of the computation of N on input (p, w_k), let w be the result. If for some $i \le k$, $\#v_i\#$ is a subword of w, then M prints $\#w_{kj}\#$ on the output tape, where $w_k = w_{k0}\#w_{k1}\#\ldots\#w_{kj}\#$.
(2) M prints # and then the word $\#v\#$, if $v \in \mathrm{dom(J)}$ and $\#v\#$ is the suffix of the first k symbols of the sequence q. (End of Stage k)

Suppose, $\langle p, q\rangle \in \mathrm{dom}(\delta_\mathrm{c})$ and let $f := \delta_\mathrm{c}\langle p, q\rangle$. Then $M\langle p, q\rangle \in \mathrm{dom}(\psi_>)$. Since $\eta_p^{\omega\omega}$ is a (δ, ρ)-realization of f, $f(z) = f \circ \delta(s) = \rho \circ \eta_p^{\omega\omega}(s) = \rho \circ N(p, s)$ for any δ-name s of any $z \in \mathrm{dom}(f) = \psi_>(q)$.
(a) Assume $z \notin f^{-1}[D]$. Let $r = u_0\#u_1\#\ldots$ be a δ-name of z. If $z \notin \mathrm{dom}(f)$, then $z \in \mathrm{I}(v)$ for some word $\#v\#$ written in Phase (2) of some stage k. Otherwise, $z \in \mathrm{dom}(f)$ and $f(z) \notin D$. Since $\rho \circ N(p, r) = f(z) \in \mathbb{R} \setminus D$, there is some i such that $\#v_i\#$ is a subword of $N(p, r)$. Then for some $k \ge i$, w_k is a prefix of r and the machine N on input (p, w_k) produces some prefix w of $N(p, r)$ such that $\#v_i\#$ is a subword of w, and so in Stage k the machine M writes a word $\#w_{kj}\#$

such that $z = \delta(r) \in \mathrm{J}(w_{kj})$. Therefore, $z \notin \psi_> \circ M\langle p, q\rangle$.
(b) Assume $z \in f^{-1}D$. Since $z \in \mathrm{dom}(f)$, no interval $\#v\#$ with $z \in \mathrm{J}(v)$ is written in Phase (2) for some k. Suppose, in Phase 1 of Stage k some word $\#w_{kj}\#$ is written by M. For all $y \in \mathrm{J}(w_{kj}) \cap \mathrm{dom}(f)$, $f(y) \in \mathrm{I}(v_i)$ (v_i from the algorithm), and so $f(y) \notin D$, since $\mathrm{I}(v_i) \subseteq \mathbb{R} \setminus D$, hence $z \notin \mathrm{J}(w_{kj})$. And so no interval $\#v\#$ with $z \in \mathrm{J}(v)$ is written in Phase (1) for some k. Therefore, $z \in \psi_> \circ M\langle p, q\rangle$.
From (a) and (b) we conclude $f^{-1}D = \psi_> \circ M\langle p, q\rangle$. Therefore, the function H is $(\delta_\mathrm{c}, \psi_>)$-computable. □

Lemma 18. *The following multi-valued function* $V :\subseteq \mathrm{C_c}(M) \rightrightarrows \mathrm{C}(M)$ *is* $(\delta_\mathrm{c}, \delta_\to)$*-computable:* $g \in V(f)$*, iff* $|f(x)| \leq 1$ *and* $|f(x) - g(x)| \leq 2/3$ *for all* $x \in \mathrm{dom}(f)$ *and* $|g(x)| \leq 1/3$ *for all* $x \in M$*.*

Roughly speaking, $g \in V(f)$, iff $|f|$ is bounded by 1 and g is an approximate extension of f with error $\leq 2/3$. In the proof we apply the computable Urysohn lemma 15.1.
Proof: Define $t : \mathbb{R} \to \mathbb{R}$ by $t(x) := (3x + 1)/2$ such that $t(-1/3) = 0$ and $t(1/3) = 1$. Then

$$F_t : f \mapsto t \circ f \text{ is } (\delta_\mathrm{c}, \delta_\mathrm{c})\text{-computable}, \tag{5}$$
$$F_A : f \mapsto f^{-1}[(-\infty; 0]] \text{ is } (\delta_\mathrm{c}, \psi_>)\text{-computable}, \tag{6}$$
$$F_B : f \mapsto f^{-1}[[1; \infty)] \text{ is } (\delta_\mathrm{c}, \psi_>)\text{-computable}, \tag{7}$$
$$U : (A, B) \rightrightarrows h, \ U \text{ from Theorem 15, is } (\psi_>, \psi_>, \delta_\to)\text{-computable}, \tag{8}$$
$$G_t : f \mapsto t^{-1} \circ f \text{ is } (\delta_\to, \delta_\to)\text{-computable}. \tag{9}$$

Properties (5) and (9) can be proved easily by standard technique, (6) and (7) hold by Lemma 17, and (8) holds by Theorem 15. Define

$$V := G_t \circ U \circ (F_A, F_B) \circ F_t.$$

Then V is $(\delta_\mathrm{c}, \delta_\to)$-computable. Suppose, $f \in \mathrm{C_c}(M)$ and $|f(x)| \leq 1$ for all $x \in \mathrm{dom}(f)$. Then $F_A \circ F_t(f) = \{x \in \mathrm{dom}(f) \mid f(x) \leq -1/3\}$ and $F_B \circ F_t(f) = \{x \in \mathrm{dom}(f) \mid f(x) \geq 1/3\}$. Suppose, $h \in U \circ (F_A, F_B) \circ F_t(f)$. Then $h(x) = 0$, if $x \in \mathrm{dom}(f)$ and $f(x) \leq -1/3$, $h(x) = 1$, if $x \in \mathrm{dom}(f)$ and $f(x) \geq 1/3$, and $0 < f(x) < 1$, if $x \notin \mathrm{dom}(f)$. Therefore, $g := G_t(h)$ satisfies $g(x) = -1/3$, if $x \in \mathrm{dom}(f)$ and $f(x) \leq -1/3$, $g(x) = 1/3$, if $x \in \mathrm{dom}(f)$ and $f(x) \geq 1/3$, and $-1/3 < g(x) < 1/3$, if $x \notin \mathrm{dom}(f)$. Since $|f|$ is bounded by 1, $|f(x) - g(x)| \leq 2/3$ for all $x \in \mathrm{dom}(f)$. □

We are now ready to prove a computable Tietze–Urysohn extension theorem for computable metric spaces.

Theorem 19 (Computable Tietze–Urysohn Extension Theorem). *The multi-valued function* $T : \mathrm{C_c}(M) \rightrightarrows \mathrm{C}(M)$ *mapping every continuous real valued*

function on M with closed domain to some total continuous extension $g : M \to \mathbb{R}$ such that

$$\inf_{x\in A} f(x) = \inf_{x\in M} g(x) \quad \textit{and} \quad \sup_{x\in A} f(x) = \sup_{x\in M} g(x), \quad \textit{if} \quad A := \mathrm{dom}(f) \neq \emptyset, \tag{10}$$

is $(\delta_{\mathrm{c}}, \delta_{\to})$-computable.

Proof: We effectivize the classical proof given, e.g., in [Eng89]. First, we consider arguments $f \in \mathrm{C_c}(M)$ such that $|f(x)| \leq 1$ for $x \in \mathrm{dom}(f)$. Let $f_0, f_1, \ldots \in \mathrm{C_c}(M)$ and $g_1, g_1, \ldots \in \mathrm{C}(M)$ be functions such that

$$f_0 = f, \quad g_{n+1} \in (2/3)^n V\{[(3/2)^n f_n]\} \quad \text{and} \quad f_{n+1} = f_n - g_{n+1} \tag{11}$$

for all n, V from Lemma 18. By induction, for $n \geq 1$,

$$f_n \quad \text{and} \quad g_n \quad \text{are well-defined,} \tag{12}$$
$$|f_n(x)| \leq (2/3)^n \quad \text{for all} \quad x \in \mathrm{dom}(f), \tag{13}$$
$$|g_n(x)| \leq (1/3)(2/3)^{n-1}, \tag{14}$$
$$f_n = f - (g_1 + \ldots + g_n). \tag{15}$$

Therefore, the series $\sum_{i=1}^{\infty} g_i$ converges uniformly to a continuous function $g : M \to \mathbb{R}$ such that $f(x) = g(x)$ for $x \in \mathrm{dom}(f)$ and $|g(x)| \leq 1$ for all $x \in M$.

It remains to show g can be computed from f. In the following we use the notation $x \mapsto |x|$ of $\mathbb{N}$ which is equivalent to $\nu_{\mathbb{N}}$. Since the function $(n, h) \mapsto (3/2)^n h$ is $(|\,|, \delta_{\mathrm{c}}, \delta_{\mathrm{c}})$-computable, the function $(n, h) \mapsto (2/3)^n h$ is $(|\,|, \delta_{\to}, \delta_{\to})$-computable, and the function $(h, h') \mapsto h - h'$ is $(\delta_{\mathrm{c}}, \delta_{\to}, \delta_{\mathrm{c}})$-computable, by Lemma 18 the multi-valued function $G_1 : (n, h) \rightrightarrows (2/3)^n V[\{(3/2)^n h\}]$ has a computable $(|\,|, \delta_{\mathrm{c}}, \delta_{\to})$-realization $H_1 : \Sigma^* \times \Sigma^\omega \to \Sigma^\omega$, and the multi-valued function $G : (n, h) \rightrightarrows h - (2/3)^n V[\{(3/2)^n h\}]$ has a computable $(|\,|, \delta_{\mathrm{c}}, \delta_{\mathrm{c}})$-realization $H : \Sigma^* \times \Sigma^\omega \to \Sigma^\omega$.

Define a computable function $H :\subseteq \Sigma^* \times \Sigma^\omega \to \Sigma^\omega$ by primitive recursion ([Wei00], Theorem 2.1.14) as follows:

$$F(\lambda, p) = p, \quad F(wa, p) = H(w, F(w, p)) \quad (w \in \Sigma^*, a \in \Sigma, p \in \Sigma^\omega).$$

For any $p \in \mathrm{dom}(\delta_{\mathrm{c}})$ let $\delta_{\mathrm{c}}(p) = f_{p0} \in \mathrm{C_c}(M)$. For $n \geq 1$ define

$$f_{pn} := \delta_{\mathrm{c}} \circ F(\mathtt{0}^n, p), \quad \text{and} \quad g_{pn} := \delta_{\to} \circ F_1(\mathtt{0}^{n-1}, F(\mathtt{0}^{n-1}, p)).$$

Then the sequences $f_{p0}, f_{p1}, \ldots$ and $g_{p1}, g_{p2}, \ldots$ satisfy (11–15). Since $(h, x) \mapsto h(x)$ is $(\delta_{\to}, \delta, \rho)$-computable, the function $(p, x, n) \mapsto g_{pn}(x)$ is $(\mathrm{id}_{\Sigma^\omega}, \delta, |\,|, \rho)$-computable, and so by the type conversion theorem ([Wei00], Theorem 3.3.15), the function $(p, x) \mapsto (g_{p0}(x), (g_{p1}(x), \ldots)$ is $(\mathrm{id}_{\Sigma^\omega}, \delta, [|| \to \rho]_{\mathbb{N}})$-computable. The limit operation $(x_0, x_1, \ldots) \mapsto \sum_{i=0}^{\infty} x_i$ for real sequences with $|x_n| \leq (2/3)^n$ is $([|\,| \to \rho]_{\mathbb{N}}, \rho)$-computable ([Wei00], Theorem 4.3.7). Therefore, $(p, x) \mapsto g_p(x)$, $g_p := \sum_n g_{pn}$, is $(\mathrm{id}_{\Sigma^\omega}, \delta, \rho)$-computable. By the type conversion theorem ([Wei00], Theorem 3.3.15) the function $p \mapsto g_p$ is $(\mathrm{id}_{\Sigma^\omega}, [\delta \to \rho]_M)$-computable,

that is, $(\mathrm{id}_{\Sigma^\omega}, \delta_{\to})$-computable. Since g_p extends $\delta_{\mathrm{c}}(p)$, the function T_1 which is the restriction of T to functions f with $|f(x)| \leq 1$ for $x \in \mathrm{dom}(f)$ is $(\delta_{\mathrm{c}}, \delta_{\to})$-computable.

Now consider functions for which not necessarily $|f(x)| \leq 1$ for $x \in \mathrm{dom}(f)$. The function $t : x \mapsto x/(1 + |x|)$ is monotone and computable and maps the real line onto the interval $(-1; 1)$. Its inverse $t^{-1} : x \mapsto x/(1 - |x|)$ is monotone and computable and maps the interval $(-1; 1)$ onto the real line. The function $T_- : f \mapsto t \circ f$ is $(\delta_{\mathrm{c}}, \delta_{\mathrm{c}})$-computable and the function $T_+ : h \mapsto t^{-1} \circ h$ is $(\delta_{\to}, \delta_{\to})$-computable. Then $T = T_+ \circ T_1 \circ T_-$, and so T is $(\delta_{\mathrm{c}}, \delta_{\to})$-computable. □

Since computable functions map computable elements to computable ones, we obtain:

Corollary 20. *Every (δ, ρ)-computable function $f :\subseteq M \to \mathbb{R}$ with co-r.e. closed domain has a (δ, ρ)-computable total (δ, ρ)-computable extension $g : M \to \mathbb{R}$ with the same* sup *and* inf.

In Theorem 19 "multi-valued" cannot be replaced by "single-valued". In general, the multi-valued extension function $T : \mathrm{C}_{\mathrm{c}}(M) \rightrightarrows \mathrm{C}(M)$ from Theorem 19 has no $(\delta_{\mathrm{c}}, \delta_{\to})$-continuous choice function $T' : \mathrm{C}_{\mathrm{c}}(M) \to \mathrm{C}(M)$. We prove a slightly stronger version which excludes also extension by functions violating (10).

Lemma 21. *For the computable metric space $\mathbf{M} := (\mathbb{R}, |\,|, \mathbb{Q}, \nu_{\mathbb{Q}})$ there is no $(\delta_{\mathrm{c}}, \delta_{\to})$-continuous function $T' : \mathrm{C}_{\mathrm{c}}(M) \to \mathrm{C}(M)$ such that $T'(f) : \mathbb{R} \to \mathbb{R}$ is a (total) continuous extension of $f :\subseteq \mathbb{R} \to \mathbb{R}$.*

Notice that for this metric space $\mathbf{M}$, δ from Definition 9 is exactly our standard representation ρ of the real numbers.

Proof: Suppose that such a function T' exists.

Let $f_0(0) := 0$ and $x \notin \mathrm{dom}(f_0)$ for all other $x \in \mathbb{R}$ and let $a := T'(f_0)(1)$, the value at 1 of the extension of f_0. Let $f_a(x) := (1+a)x$. There are $\overline{p}, \overline{q}, \overline{r} \in \Sigma^\omega$ such that $\delta_{\to}(\overline{p}) = f_a$, $\psi_>(\overline{q}) = \{0\}$ and $\rho(\overline{r}) = 1$. Since $\psi_>(\#^i\overline{q}) = \psi_>(\overline{q}) = \{0\}$ for all $i \in \mathbb{N}$,

$$(T' \circ \delta_{\mathrm{c}}\langle\overline{p}, \#^i\overline{q}\rangle) \circ \rho(\overline{r}) = T'(f_0)(1) = a, \tag{16}$$

$$(T' \circ \delta_{\mathrm{c}}\langle\overline{p}, \#^\omega\rangle) \circ \rho(\overline{r}) = T'(f_a)(1) = 1 + a. \tag{17}$$

for all $i \in \mathbb{N}$.

By definition there is a continuous function $h :\subseteq \Sigma^\omega \to \Sigma^\omega$ such that $T' \circ \delta_{\mathrm{c}}\langle p, q\rangle = \delta_{\to} \circ h\langle p, q\rangle$. For any $r \in \mathrm{dom}(\rho)$ we obtain $(T' \circ \delta_{\mathrm{c}}\langle p, q\rangle) \circ \rho(r) = (\delta_{\to} \circ h\langle p, q\rangle) \circ \rho(r) = \rho \circ \eta^{\omega\omega}_{h\langle p,q\rangle}(r)$. By the utm-theorem for $\eta^{\omega\omega}$ and continuity of $\rho :\subseteq \Sigma^\omega \to \mathbb{R}$, the function

$$G :\subseteq \Sigma^\omega \to \mathbb{R}, \quad G(q) := (T' \circ \delta_{\mathrm{c}}\langle\overline{p}, q\rangle) \circ \rho(\overline{r}), \quad \text{is continuous.}$$

By (16,17) and continuity of G,

$$a = \lim_{i\to\infty} G(\#^i\overline{q}) = G(\lim_{i\to\infty} \#^i\overline{q}) = G(\#^\omega) = 1 + a$$

(contradiction). □

In [Die60] a Tietze–Urysohn extension is defined explicitly: Suppose, $f :\subseteq M \to \mathbb{R}$ is continuous on its closed domain A and $1 \le f(x) \le 2$ for all $x \in A$. Then the function $g : M \to \mathbb{R}$ with $g(x) = f(x)$ for $x \in A$ and

$$g(x) := \frac{\inf_{y\in A}\big(f(y)d(x,y)\big)}{d(x,A)}$$

otherwise, is a continuous extension of f. By Lemma 21, this proof cannot be effectivized, since g is a single-valued function of f (and A). While for the Urysohn lemma $\psi^{\rm dist}$-names allow single-valued output (Theorem 13 versus Theorem 15), it is an open problem, whether there is some $(\delta_{\rm c0}, \delta_{\to})$-continuous Tietze extension operator, where

$$\delta_{\rm c0}\langle p, q\rangle = f \iff \eta_p^{\omega\omega} \text{ is a } (\delta,\rho)\text{-realization of } f \text{ and } \mathrm{dom}(f) = \psi^{\rm dist}(q)$$

(cf. Definition 11).

References

BW99. Vasco Brattka and Klaus Weihrauch. Computability on subsets of Euclidean space I: Closed and compact subsets. *Theoretical Computer Science*, 219:65–93, 1999.

Die60. J. Dieudonné. *Foundations of Modern Analysis*. Academic Press, New York, 1960.

Eng89. Ryszard Engelking. *General Topology*, volume 6 of *Sigma series in pure mathematics*. Heldermann, Berlin, 1989.

KW85. Christoph Kreitz and Klaus Weihrauch. Theory of representations. *Theoretical Computer Science*, 38:35–53, 1985.

MTY97. Takakazu Mori, Yoshiki Tsujii, and Mariko Yasugi. Computability structures on metric spaces. In Douglas S. Bridges, Cristian S. Calude, Jeremy Gibbons, Steve Reeves, and Ian H. Witten, editors, *Combinatorics, Complexity, and Logic*, Discrete Mathematics and Theoretical Computer Science, pages 351–362, Singapore, 1997. Springer. Proceedings of DMTCS'96.

PER89. Marian B. Pour-El and J. Ian Richards. *Computability in Analysis and Physics*. Perspectives in Mathematical Logic. Springer, Berlin, 1989.

Wei93. Klaus Weihrauch. Computability on computable metric spaces. *Theoretical Computer Science*, 113:191–210, 1993. Fundamental Study.

Wei00. Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.

WS81. Klaus Weihrauch and Ulrich Schreiber. Embedding metric spaces into cpo's. *Theoretical Computer Science*, 16:5–24, 1981.

YMT99. Mariko Yasugi, Takakazu Mori, and Yoshiki Tsujii. Effective properties of sets and functions in metric spaces with computability structure. *Theoretical Computer Science*, 219:467–486, 1999.

Zho96. Qing Zhou. Computable real-valued functions on recursive open and closed subsets of Euclidean space. *Mathematical Logic Quarterly*, 42:379–409, 1996.

Is the Linear Schrödinger Propagator Turing Computable?

Klaus Weihrauch[1] and Ning Zhong[2]

[1] Theoretische Informatik I, FernUniversität, 58084 Hagen, Germany
Klaus.Weihrauch@FernUni-Hagen.de
[2] Clermont College of the University of Cincinnati, Batavia, OH 45103-1785, USA
Ning.Zhong@uc.edu

Abstract. In this note we study Turing computability of the linear inhomogeneous Schrödinger propagator S. We prove: (1) S is computable when the initial functions are from Sobolev spaces. (2) When acting on $L^p(\mathbb{R}^d)$, S is computable, if and only if $p = 2$.

1 Introduction

The Schrödinger equation is the fundamental equation of quantum mechanics. It provides a canonical description for the envelop dynamics of a quasi-monochromatic plane wave propagating in a medium. In this note we consider the inhomogeneous linear Schrödinger equation

$$u_t = i\triangle u + \phi, \quad t \in \mathbb{R},\, x \in \mathbb{R}^d,\, i = \sqrt{-1}$$

with the forcing term $\phi(t, x)$ and the initial condition $u(0, x) = f(x)$, where $\triangle u = \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \cdots + \frac{\partial^2 u}{\partial x_d^2}\right)$. In this note we study Turing computability of the Schrödinger propagator $S : (t, \phi, f) \mapsto u(t, \cdot)$.

The equation describes the movement of a particle of mass equal to $1/2$. The particle is moving in the absence of external forces if the forcing term $\phi \equiv 0$. This is the quantum analogue of Galileo's particle moving in a straight line at constant speed. The physical interpretation, in quantum mechanics, is that the square of the modulus, $|u(t, x)|^2$, is the probability density for finding the particle at time t and place x. The probability interpretation requires that

$$\int_{\mathbb{R}^d} |u(t, x)|^2 dx = 1 \quad \text{for all } t \geq 0$$

for physically relevant solutions. One is immediately led to think that $L^2(\mathbb{R}^d)$, thus Sobolev spaces $H^s(\mathbb{R}^d)$, will play a distinguished role. In fact, the Schrödinger equation is an example where the precise existence and regularity results require Sobolev spaces. We note that the Schrödinger equation is neither elliptic, parabolic, nor hyperbolic.

For another operator, the linear wave operator, computability has already been studied in detail. The following results are established in [PER89,PEZ97]

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 369–377, 2001.

and [WZ98]: (1) The wave propagator $S_W : \mathbb{R} \times C^k(\mathbb{R}^3) \times C^{k-1}(\mathbb{R}^3) \to C^{k-1}(\mathbb{R}^4)$ is $(\rho, \delta_3^k, \delta_3^{k-1}, \delta_4^{k-1})$-computable, that is, there is a Type-2 Turing machine which computes a δ_4^{k-1}-name for $S_W(t, f, g)$, the amplitude at time $t \in \mathbb{R}$ when fed a ρ-name of the time t, a δ_3^k-name of the initial amplitude f, and a δ_3^{k-1}-name of the initial velocity g (where ρ is the standard representation of $\mathbb{R}$ and δ_d^k is the standard representation of $C^k(\mathbb{R}^d)$ [WZ98]; (2) S_W is not $(\rho, \tau_3^{k-1}, \tau_3^{k-1}, \tau_4^{k-1})$-continuous, where τ_d^k is the compact open topology on $C^k(\mathbb{R}^d)$. As a consequence of the First Main Theorem of Pour-El and Richards, there exist computable continuous functions f and g such that the wave at time $t = 1$, $S_W(1, f, g)$, is not computable; (3) The wave propagator is computable on Sobolev spaces.

Like the wave propagator the computability of the solution operator of the Schrödinger equation is closely related to the space of the initial conditions. As is known there exists a nonzero solution $u \in C^\infty(\mathbb{R}_t \times \mathbb{R}_x^d)$ with

$$u_t = i\triangle u \quad \text{and } u = 0 \text{ for all } t < 0.$$

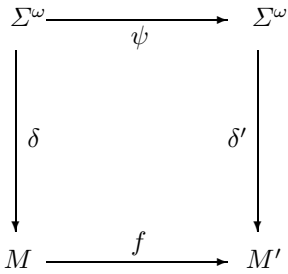
Thus for the Schrödinger equation there is not uniqueness for the initial value problem in the category of all smooth solutions. Hence the continuously kth differentiable functions, $k \in \mathbb{N}$, are not good choices as initial conditions for the Schrödinger equation. Other natural candidates for the initial data are Sobolev functions and L^p functions. In Section 2 below we study computability of the Schrödinger propagator on Sobolev spaces. Like the wave propagator the Schrödinger propagator is computable on Sobolev spaces. In Section 3 we discuss computability of the Schrödinger propagator on L^p spaces. We prove that for any $p \neq 2$, the solution operator of the Schrödinger equation is not computable when acting on L^p.

2 The Schrödinger Operator on Sobolev Spaces

Among the various models of computation proposed for computable analysis we use Type-2 Theory of Effectivity (TTE for short), as our computational model. We shall review some basic definitions. For details on TTE the reader is referred to the textbook [Wei00]. Let Σ be a sufficiently large finite alphabet, Σ^* the set of finite words over Σ with the discrete topology, and Σ^ω the set of infinite words over Σ with the Cantor topology. In TTE, Turing computability is extended from finite words $w \in \Sigma^*$ to infinite sequences $p = (p_0 p_1 p_2 \dots) \in \Sigma^\omega$ of symbols. A Type-2 Turing machine T computes a function (possibly partial) $f_T :\subseteq \Sigma^\omega \to \Sigma^\omega$ in the following way: $f_T(p) = q$, if and only if the machine T reads the input sequence $p = (p_0 p_1 p_2 \dots)$ symbol by symbol from the left to the right and writes the output sequence $q = (q_0 q_1 q_2 \dots)$ symbol by symbol from the left to the right. A notation (representation) of a set M is a surjective map $\delta :\subseteq \Sigma^* \to M$ ($\delta :\subseteq \Sigma^\omega \to M$). For any $x \in M$ and $p \in \Sigma^*$ (or $p \in \Sigma^\omega$), p is called a δ-name of x if $\delta(p) = x$.

Through a notation or a representation computations on M are defined by means of computations on Σ^* or Σ^ω which can be performed by ordinary or

Type-2 Turing machines. If δ and δ' are representations of M and M' respectively, then a function $\psi :\subseteq \Sigma^\omega \to \Sigma^\omega$ is called a (δ, δ')-realization of $f :\subseteq M \to M'$, if and only if the following diagram commutes,



$$f \circ \delta(p) = \delta' \circ \psi(p)$$

for $p \in \mathrm{dom}(f \circ \delta)$

i.e., $f \circ \delta(p) = \delta' \circ \psi(p)$ for all $p \in \mathrm{dom}(f \circ \delta)$. A function f is called (δ, δ')-continuous (-computable), if and only if it has a continuous (computable) (δ, δ')-realization. The concept can be generalized straightforwardly to $(\delta_1, \ldots, \delta_n, \delta_0)$-realizations of functions $f :\subseteq M_1 \times \ldots \times M_n \to M_0$.

For any representations δ and δ' there is a canonical representation $[\delta \to \delta']$ of the set of (δ, δ')-continuous functions $f : M \to M'$. It is defined by means of a standard representation $\eta : \Sigma^\omega \to F^{\omega\omega}$, the set of (essentially) all continuous partial functions from Σ^ω to Σ^ω, such that $[\delta \to \delta'](p) = f$ if and only if $\eta(p)$ is a (δ, δ')-realization of f (Chap. 2.3 in [Wei00]). η satisfies both the universal Turing machine theorem and the smn-theorem. If both M and M' are topological T_0-spaces with countable bases and both δ and δ' are admissible representations (Def. 3.2.7 in [Wei00]), then $f : M \to M'$ is continuous if and only if it is (δ, δ')-continuous (Thm. 3.2.1 in [Wei00]). In this case $[\delta \to \delta']$ is a representation of $C(M, M')$, the set of all continuous functions from M to M'.

The following theorem on type conversion will be needed (Thm. 3.3.15 in [Wei00]):

Lemma 1. *Let $\delta_i :\subseteq \Sigma^\omega \to M_i$ be a representation of the set M_i, $0 \leq i \leq k$. Let $f : M_1 \times \ldots \times M_k \to M_0$ and define $F(x_1, \ldots, x_{k-1})(x_k) := f(x_1, \ldots, x_k)$. Then f is $(\delta_1, \ldots, \delta_k, \delta_0)$-computable (-continuous), iff F is $(\delta_1, \ldots, \delta_{k-1}, [\delta_k \to \delta_0])$-computable (-continuous).*

Let $\mathbb{N}, \mathbb{Q}$, and $\mathbb{R}$ denote the set of natural numbers, rational numbers, and real numbers respectively. Let $\rho^d :\subseteq \Sigma^\omega \to \mathbb{R}^d$ be the standard representation of $\mathbb{R}^d$ [Wei00]. Roughly speaking, $p \in \Sigma^\omega$ is a ρ^d-name of $x \in \mathbb{R}^d$, if p is a sequence of (names of) rational d-tuples which fast converges to x, where in a metric space, a sequence $\{a_i\}$ fast converges to a, if the distance between a_i and a is less than 2^{-i} for all $i \in \mathbb{N}$. The representations ρ^d are admissible.

In the following we introduce standard representations of L^p-spaces and Sobolev spaces (cf. [WZ98]). For any $1 \leq p < \infty$ the space $L^p(\mathbb{R}^d)$ of L^p-functions is the set of all measurable complex valued functions f such that

$\int_{\mathbb{R}^d} |f(x)|^p dx < \infty$ with the norm $||f||_{L^p} = \{\int_{\mathbb{R}^d} |f(x)|^p dx\}^{1/p}$. Let M be the countable set of rational complex valued finite step functions defined on $\mathbb{R}^d$ as follows:

$$M = \left\{ \sum_{i=0}^{k} c_i \cdot I_{r^i s^i} : k \in \mathbb{N},\ c_i \text{ rational complex},\ r^i, s^i \in \mathbb{Q}^d,\ r^i < s^i \right\}, \quad (1)$$

where $r^i = (r_1, r_2, \cdots, r_d), s^i = (s_1, s_2, \cdots, s_d)$, $r^i < s^i$ if and only if $r_j < s_j$ for all $1 \leq j \leq d$, $(r^i, s^i) = (r_1, s_1) \times (r_2, s_2) \times \cdots (r_d, s_d)$, and $I_{r^i s^i}(x) = 1$ if $x \in (r^i, s^i)$ and $I_{r^i s^i}(x) = 0$ otherwise. The set M is dense in $L^p(\mathbb{R}^d)$. On $L^p(\mathbb{R}^d)$ we consider the Cauchy representation δ_{l^p} which is admissible (see Sec. 8.2 in [Wei00]).

Definition 2. *Let $\nu_M : \Sigma^* \setminus \{\#\} \to M$ be a standard notation of M. The Cauchy representation δ_{L^p} for $L^p(\mathbb{R}^d)$ is defined as follows: For any $f \in L^p(\mathbb{R}^d)$ and $q = (\#q_0\#q_1\#q_2\#\cdots) \in \Sigma^\omega$, $\delta_{L^p}(q) = f$, if and only if for all $i \in \mathbb{N}$, $q_i \in \mathrm{dom}(\nu_M)$ and $||\nu_M(q_i) - f||_{L^p} < 2^{-i}$.*

For any $s \in \mathbb{R}$, the Sobolev space $H^s(\mathbb{R}^d)$ is the set of all functions f such that

$$T_s(f) \in L^2(\mathbb{R}^d) \text{ where } T_s(f) := (1 + |\cdot|^2)^{s/2} \hat{f}(\cdot)$$

($\hat{f}$ denotes the Fourier transform of f) equipped with the norm

$$||f||_{H^s} = ||T_s(f)||_{L^2}.$$

T_s is an isometric mapping and the space $H^s(\mathbb{R}^d)$ is a separable Hilbert space. In [WZ98] a standard representation δ^s is introduced for the Sobolev space as follows:

Definition 3. *For any $s \in \mathbb{R}$ define a representation $\delta^s :\subseteq \Sigma^\omega \to H^s(\mathbb{R}^d)$ by*

$$\delta^s := T_s^{-1} \circ \delta_{L^2}, \quad \text{or}$$
$$\delta^s(p) = f \iff \delta_{L^2}(p) = (1 + |\cdot|^2)^{s/2} \hat{f}(\cdot).$$

Like δ_{L^2}, δ^s is admissible. By the definitions above, computations on $L^2(\mathbb{R}^d)$ are carried out by Type-2 Turing machines with names being sequences of (names of) rational finite step functions, and computations on $H^s(\mathbb{R}^d)$ are simply reduced to computations on $L^2(\mathbb{R}^d)$.

To study computability of the solution operator of the Schrödinger equation, we also need to introduce a representation for the space $C(\mathbb{R}; L^2(\mathbb{R}^d))$, which is the set of all continuous functions from $\mathbb{R}$ to $L^2(\mathbb{R}^d)$ with the compact-open topology. The set of all $D(K, U) := \{H \in C(\mathbb{R}; L^2(\mathbb{R}^d)) : H(K) \subseteq U\}$, where $K \subset \mathbb{R}$ is a compact set and $U \subseteq L^2(\mathbb{R}^d)$ is an open set, is a subbase of the compact-open topology. Recall that the countable set M (see (1)) of all rational

complex valued finite step functions defined on $\mathbb{R}^d$ is dense in $L^2(\mathbb{R}^d)$. Thus the open balls $B(s,r) = \{f \in L^2(\mathbb{R}^d) : ||f - s||_{L^2} < r\}$ with rational radius (in L^2-norm) and centered at an element of M form a countable subbase of $L^2(\mathbb{R}^d)$. Let $\mathcal{B}$ denote this subbase. Let $\mathcal{J}$ be the set of all bounded rational closed intervals in $\mathbb{R}$ and $\mathcal{U}$ the set $\{D(I,B) : I \in \mathcal{I}, B \in \mathcal{B}\}$. Then $\mathcal{U}$ is a countable subbase of $C(\mathbb{R}; L^2(\mathbb{R}^d))$. Thus every function in $C(\mathbb{R}, L^2(\mathbb{R}^d))$ can be approximated by elements in $\mathcal{U}$. The following representation makes use of this fact.

Definition 4. *. Let $\nu_{\mathcal{U}} :\subseteq \Sigma^* \to \mathcal{U}$ be a standard notation (coding) of the countable set $\mathcal{U}$ such that* $\mathrm{dom}(\nu_{\mathcal{U}}) \subseteq (\Sigma \setminus \{\#\})^*$*). Define the* compact-open *representation $\delta_{co} : \Sigma^\omega \to C(\mathbb{R}, L^2(\mathbb{R}^d))$ of $C(\mathbb{R}, L^2(\mathbb{R}^d))$ as follows: $\delta_{co}(p) = H$, iff*

$$q = (\#v_0\#v_1\#v_2\cdots),\ v_i \in \mathrm{dom}(\nu_{\mathcal{U}}) \quad \textit{and}$$

$$\{v_i : i \in \mathbb{N}\} = \{w \in \mathrm{dom}(\nu_{\mathcal{U}}) : H \in \nu_{\mathcal{U}}(w)\}.$$

Since both representations ρ and δ_{L^2} are admissible, $[\rho \to \delta_{L^2}]$ is also a representation of $C(\mathbb{R}, L^2(\mathbb{R}^d))$. The two representations, δ_{co} and $[\rho \to \delta_{L^2}]$, are equivalent, although they are defined by different approaches.

Lemma 5. $\delta_{co} \equiv [\rho \to \delta_{L^2}]$.

Proof. The proof is similar to that of Theorem 6.1.7 in [Wei00]. We omit it. □

For the space $C(\mathbb{R}, H^s(\mathbb{R}^d))$ we use the canonical representation $[\rho \to \delta^s]$. A straightforward proof shows:

Proposition 6. $[\rho \to \delta^s] \ = \ T_s^{-1} \circ [\rho \to \delta_{L^2}] \ = \ T_s^{-1} \circ \delta_{co}$

In words, for any $\phi \in C(\mathbb{R}; H^s(\mathbb{R}^d))$, $q \in \Sigma^\omega$ is a $[\rho \to \delta^s]$-name of ϕ, if and only if q is a $[\rho \to \delta_{L^2}]$-name of $T_s(\phi) = (1 + |\xi|^2)^{s/2}\hat{\phi}_x(t,\xi)$. The representation thus reduces computations on $C(\mathbb{R}; H^s(\mathbb{R}^d))$ to computations on $C(\mathbb{R}; L^2(\mathbb{R}^d))$. After these preparations we can formulate our central theorem.

Theorem 7. *For any $s \in \mathbb{R}$ consider the inhomogeneous linear Schrödinger equation:*

$$u_t = i\triangle u + \phi, \quad t \in \mathbb{R},\ x \in \mathbb{R}^d,\ \phi \in C(\mathbb{R}; H^s(\mathbb{R}^d))$$

with the initial condition $u(0,x) = f(x) \in H^s(\mathbb{R}^d)$. Then the solution operator

$$S : \mathbb{R} \times C(\mathbb{R}; H^s(\mathbb{R}^d)) \times H^s(\mathbb{R}^d) \to H^s(\mathbb{R}^d), \quad (t, \phi, f) \mapsto S(t)(\phi, f) = u(t, \cdot),$$

is $(\rho, [\rho \to \delta^s], \delta^s, \delta^s)$-computable.

Proof. For given initial condition $f \in H^s(\mathbb{R}^d)$, continuous forcing $\phi : \mathbb{R} \to H^s(\mathbb{R}^d)$ and time t the solution $u(t,\cdot) \in H^s(\mathbb{R}^d)$ is given explicitly by

$$\hat{u}_\xi(t,\xi) = e^{-it|\xi|^2}\hat{f}(\xi) + \int_0^t e^{-i|\xi|^2(t-\tau)}\widehat{\phi(\tau)}(\xi)\, d\tau \tag{2}$$

([Sul99]). Denote $u(t,\cdot)$ as $g(\cdot)$. Multiplying (2) by $(1+|\xi|^2)^{s/2}$ and substituting by T_s we obtain:

$$T_s(g)(\xi) = e^{-it|\xi|^2} T_s(f)(\xi) + \int_0^t e^{-i|\xi|^2(t-\tau)} T_s \circ \phi(\tau)(\xi)\, d\tau \tag{3}$$

Let p be a ρ-name of t, q_1 a δ^s-name of f, and q_2 a $[\rho \to \delta^s]$-name of ϕ. From these data we want to compute a δ^s-name r of g. By Proposition 6 a δ^s-name of f (or g respectively) is a δ_{L^2}-name of $T_s(f)$ (or $T_s(g)$ respectively), and a $[\rho \to \delta^s]$-name of ϕ is a $[\rho \to \delta_{L^2}]$-name of $T_s \circ \phi$. We obtain:

$$\begin{aligned}&\delta_{L^2}(r)(\xi)\\ &\quad = e^{-i\rho(p)|\xi|^2} \delta_{L^2}(q_1)(\xi) + \int_0^{\rho(p)} e^{-i|\xi|^2(\rho(p)-\tau)} [\rho \to \delta_{L^2}](q_2)(\tau))(\xi)\, d\tau\end{aligned} \tag{4}$$

which is a computational problem w.r.t. δ_{L^2}. We interrupt the proof of Theorem 7 by some lemmas.

Lemma 8. *1. The function $t \mapsto h$, $h(\xi) := e^{-it|\xi|^2}$, is $(\rho, [\rho^d \to \rho^2])$-computable.*
2. The function $(t,\tau) \mapsto h$, $h(\xi) = e^{-i|\xi|^2(t-\tau)}$, is $(\rho,\rho,[\rho^d \to \rho^2])$-computable.

Lemma 9. *1. Addition $(f,f') \mapsto f+f'$ on $L^2(\mathbb{R}^d)$ is $(\delta_{L^2},\delta_{L^2},\delta_{L^2})$-computable.*
2. Multiplication $(f,g,K) \mapsto fg$ where $fg(x) = f(x)g(x)$ for $f \in L^2(\mathbb{R}^d)$, $g \in C(\mathbb{R}^d)$, and $K \in \mathbb{R}$ with $\forall x |g(x)| < K$ is $(\delta_{L^2}, [\rho^d \to \rho^2], \rho, \delta_{L^2})$-computable.

A modulus of uniform continuity of $f : \mathbb{R} \to L^2(\mathbb{R}^d)$ on $[0;1]$ is a function $\mu : \mathbb{N} \to \mathbb{N}$ such that $||f(x) - f(y)||_{L^2} \le 2^{-n}$ for $|x-y| \le 2^{-m(n)}$ $(x,y \in [0;1])$.

Lemma 10. *There is a computable function $d :\subseteq \Sigma^\omega \to \Sigma^\omega$ such that $d(p) = 0^{\mu(0)}10^{\mu(1)}10^{\mu(2)}1\ldots$ for some modulus μ of uniform continuity of $[\rho \to \delta_{L^2}](p)$ on $[0;1]$ (for all $p \in \mathrm{dom}([\rho \to \delta_{L^2}])$).*

Lemma 11. *The integration*

$$(a,b,h) \mapsto \int_a^b h(\tau)\, d\tau, \quad a,b \in \mathbb{R}, h \in C(\mathbb{R}; L^2(\mathbb{R}^d))$$

is $(\rho,\rho,[\rho \to \delta_{L^2}],\delta_{L^2})$-computable.

We merely outline the proofs.

Proof of Lemma 8: (1) The function $(t,\xi) \mapsto e^{-it|\xi|^2}$ is (ρ,ρ^d,ρ^2)-computable. Apply Lemma 1. (2) (correspondingly)

Proof of Lemma 9: Generalize the proof of Lemma 4.3 in [WZ98] from 3 to d dimensions.

Proof of Lemma 10: By Lemma 5 it suffices to consider δ_{co} instead of $[\rho \to \delta_{L^2}]$. If $\delta_{co}(p) = h$, then p is (encodes) a list of all pairs (I, B) such that $h[I] \subseteq B$. Consider $n \in \mathbb{N}$. Then for every $t \in [0;1]$ there is some pair (I_t, B_t) in the list such that B_t has a radius $< 2^{-n}$ and $t \in I_t/3$, where $[a;b]/3 := (a+(b-a)/3\,;\, b-(b-a)/3)$. Finitely many Intervals $I_{t_1}/3, \ldots, I_{t_j}/3$ cover $[0;1]$. Let $\mu(n)$ be less than the length of every I_{t_i}. Then μ is a modulus of uniform continuity. By systematic search, some Type-2 machine can compute an encoded modulus from p.

Proof of Lemma 11: First consider the special case $a = 0, b = 1$. The integral can be defined by the limit of Riemann sums:

$$\int_0^1 h(\tau)\, d\tau = \lim_{k\to\infty} \sum_{i=1}^{k} h(\frac{i}{k})/k.$$

For obtaining an approximate sum with error $\leq 2^{-n-1}$ (which can be used as the nth term in a Cauchy name), choose $k = m(n+2)$ and compute each $h(\frac{i}{k})$ with precision 2^{-n-2}.
The general case a, b can be reduced to the special one (cf. the proof of Theorem 6.4.1.2 in [Wei00]).

Now we return to the proof of Theorem 7. Combining the functions from the lemmas, it follows that a sequence r can be computed from p, q_1 and q_2 satisfying Equation (4). This proves the theorem. □

3 The Schrödinger Propagator on L^p-Spaces

We now return to the linear homogeneous Schrödinger equation

$$u_t = i\triangle u, \quad t \in \mathbb{R},\, x \in \mathbb{R}^d,$$

with the initial condition $u(0, x) = f(x)$. For any $1 \leq p < \infty$ and any $t \in \mathbb{R}$, the Schrödinger propagator $S(t) : L^p(\mathbb{R}^d) \to L^p(\mathbb{R}^d)$ is well defined. Moreover, as a special case of Theorem 7, $S(t)$ is computable on $L^2(\mathbb{R}^d)$ for any computable $t \in \mathbb{R}$, since $L^2(\mathbb{R}^d) = H^0(\mathbb{R}^d)$ and, by a straightforward calculation, the Fourier transform on $L^2(\mathbb{R}^d)$ is $(\delta_{L^2}, \delta_{L^2})$-computable. However when acting on $L^p(\mathbb{R}^d)$ for any $p \neq 2$, the Schrödinger propagator behaves quite differently: It is no longer computable. More precisely we have

Theorem 12. *For any $t \neq 0$, the Schrödinger propagator $S(t) : L^p(\mathbb{R}^d) \to L^p(\mathbb{R}^d)$ is $(\delta_{L^p}, \delta_{L^p})$-computable if and only if $p = 2$.*

Proof. We consider the case when $p < 2$. The same argument applies to the situation where $p > 2$. For any $t \neq 0$, we compute the solution of the Schrödinger equation $u_t = i\triangle u$ with the initial value $g(x) = f(x)/||f||_{L^p}$, where $f(x) =$

$e^{-a|x|^2/2}$ and $a > 0$ is a constant. We note that g is infinitely differentiable, $g \in L^p(\mathbb{R}^d)$, and $||g||_{L^p} = 1$. Since

$$\begin{aligned} S(t)g &= \mathcal{F}^{-1}\left(e^{-it|\xi|^2}\hat{f}\right)/||f||_{L^p} \\ &= \left(a^{-d/2}(1/a+2it)^{-d/2}e^{-|x|^2/(2(1/a+2it))}\right)/||f||_{L^p}, \end{aligned}$$

the L^p-norm of $S(t)g$ can be computed explicitely as follows:

$$\begin{aligned} ||S(t)g||_{L^p}^p &= \int_{\mathbb{R}^d} \left|a^{-d/2}(1/a+2it)^{-d/2}e^{-|x|^2/(2(1/a+2it))}\right|^p dx/||f||_{L^p} \\ &= |(1+2ita)|^{-pd/2} \\ &\qquad \times \int_{\mathbb{R}^d} \left|e^{-a|x|^2/(2(1+4a^2t^2))}e^{ia^2t|x|^2/(1+4a^2t^2)}\right|^p dx/||f||_{L^p} \\ &= |(1+2ita)|^{-pd/2} \int_{\mathbb{R}^d} e^{-pa|x|^2/(2(1+4a^2t^2))}dx/||f||_{L^p} \\ &= |(1+2ita)|^{-pd/2}\left(\frac{1+4a^2t^2}{2\pi ap}\right)^{d/2}/||f||_{L^p} \\ &= \frac{|(1+2ita)|^{-pd/2}\left(\frac{1+4a^2t^2}{2\pi ap}\right)^{d/2}}{\left(\frac{1}{2\pi ap}\right)^{d/2}} \\ &= |1+2ita|^{-pd/2}\left(1+4t^2a^2\right)^{d/2} \\ &\to \infty \end{aligned}$$

as $a \to +\infty$. Hence

$$\sup_{\phi \in L^p(\mathbb{R}^d), ||\phi||_{L^p}=1} \frac{||S(t)\phi||_{L^p}}{||\phi||_{L^p}} = \infty.$$

This shows that the operator $S(t)$ is unbounded on $L^p(\mathbb{R}^d)$. Since $S(t)$ is a linear operator, the unsoundness implies discontinuity. A discontinuous operator is of course not computable.

□

If δ and δ' are representations of M and M' respectively, and f is (δ, δ')-computable, then, by definition, f maps every δ-computable $x \in M$ to a δ'-computable element $f(x) \in M'$. There is a computability property which is strictly weaker that (δ, δ')-computability. A map $f : M \to M'$ is called a (δ, δ')-computable invariant if it maps every δ-computable element in M to a

δ'-computable element in M'. In [Bra99] Brattka showed that there exist (δ, δ')-computable invariants which are not (δ, δ')-computable.

Theorem 12 shows that $S(t)$ is not $(\delta_{L^p}, \delta_{L^p})$-computable for any $t \neq 0$ and $p \neq 2$. Can $S(t)$ be a $(\delta_{L^p}, \delta_{L^p})$-computable invariant? In the following we use the First Main Theorem by Pour-El and Richards [PER89] to show that $S(t)$ is not a $(\delta_{L^p}, \delta_{L^p})$-computable invariant either, if p is a computable real number.

Corollary 13. *For any $t \neq 0$ and computable $p \neq 2$, the Schrödinger propagator $S(t) : L^p(\mathbb{R}^d) \to L^p(\mathbb{R}^d)$ is not a $(\delta_{L^p}, \delta_{L^p})$-computable invariant.*

Proof. As is known classically, $S(t)$ is a closed linear operator. By the proof of Theorem 12, $S(t)$ is unbounded. By Theorem 9.3.3 in [Wei00], the sequences computable w.r.t. δ_{L^p} are the computable sequences w.r.t. "intrinsic" L^p-computability [PER89]. Applying the First Main Theorem of Pour-El and Richards [PER89], $S(t)$ must map some δ_{L^p}-computable initial $L^p(\mathbb{R}^d)$ function f to a solution $S(t)f \in L^p(\mathbb{R}^d)$ which is not δ_{L^p}-computable.

□

References

BN73. José Barros-Neto. *An introduction to the theory of distributions*, volume 14 of *Pure and Applied Mathematics*. Marcel Dekker Inc., New York, 1973.

Bra99. Vasco Brattka. Computable invariance. *Theoretical Computer Science*, 210:3–20, 1999.

PER89. Marian B. Pour-El and J. Ian Richards. *Computability in Analysis and Physics*. Perspectives in Mathematical Logic. Springer, Berlin, 1989.

PEZ97. Marian Pour-El and Ning Zhong. The wave equation with computable initial data whose unique solution is nowhere computable. *Mathematical Logic Quarterly*, 43(4):499–509, 1997.

Rau91. Jeffrey Rauch. *Partial Differential Equations*, volume 128 of *Graduate Texts in Mathematics*. Springer, New York, 1991.

Sul99. Catherine Sulem and Pierre-Louis Sulem. *The Nonlinear Schrödinger Equation*, volume 128 of *Applied Mathematical Sciences*. Springer, New York, 1999.

Wei00. Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.

WZ98. Klaus Weihrauch and Ning Zhong. The wave propagator is Turing computable. In Ker-I Ko, Anil Nerode, Marian B. Pour-El, Klaus Weihrauch, and Jiří Wiedermann, editors, *Computability and Complexity in Analysis*, volume 235 of *Informatik Berichte*, pages 127–155. FernUniversität Hagen, August 1998. CCA Workshop, Brno, Czech Republic, August, 1998.

A Computable Spectral Theorem

Martin Ziegler[1⋆] and Vasco Brattka[2⋆⋆]

[1] Heinz Nixdorf Institute, Fachbereich 17
University of Paderborn, 33095 Paderborn, Germany
ziegler@uni-paderborn.de
[2] Theoretische Informatik I, Informatikzentrum
FernUniversität, 58084 Hagen, Germany
vasco.brattka@fernuni-hagen.de

Abstract. Computing the spectral decomposition of a normal matrix is among the most frequent tasks to numerical mathematics. A vast range of methods are employed to do so, but all of them suffer from instabilities when applied to degenerate matrices, i.e., those having multiple eigenvalues. We investigate the spectral representation's effectivity properties on the sound formal basis of computable analysis. It turns out that in general the eigenvectors cannot be computed from a given matrix. If however the size of the matrix' spectrum (=number of different eigenvalues) is known in advance, it *can* be diagonalized effectively. Thus, in principle the spectral decomposition can be computed under remarkably weak non-degeneracy conditions.

1 Introduction

The Spectral Theorem for normal matrices is one of the most important theorems of linear algebra. It ensures the existence of an appropriately rotated coordinate system in which a normal operator becomes diagonal.

Theorem 1 (Spectral Theorem). *Let $A \in \mathbb{C}^{n\times n}$ be a normal matrix. There exists an orthogonal basis $(x_1, ..., x_n)$ of $\mathbb{C}^n$ and complex numbers $\lambda_1, ..., \lambda_n \in \mathbb{C}$ such that $Ax_i = \lambda_i x_i$ for each $i = 1, ..., n$.*

The Spectral Theorem has a large number of applications in mathematics, computer science, engineering and other disciplines of which we just mention the following:

- Mathematically it yields a nice normal form for normal linear operators.
- The Spectral Theorem induces an easy-to-use calculus for *functions* of self-adjoint matrices.
- It enables the explicit solvability of vector-valued linear differential equations.

⋆ Work partially supported by DFG Grant Me 872/7-3.
⋆⋆ Work partially supported by DFG Grant BR 1807/4-1.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 378–388, 2001.

- In quantum physics, it provides the basic tool for describing a measurement process ('collapse of the system into an eigenstate').
- Whether some dynamical systems are stable or instable depends on the eigenvalues of their dynamics.
- In mechanical engineering it is particularly important to align a rotating body strictly along its mass centroid axis in order to avoid dynamic imbalances which might otherwise destroy its bearing.
- We also mention the numerous applications to computer science, e.g. in graph theory [4] and combinatorial optimization [12].
- Finally, from the complexity theoretic point of view it can increase efficiency to diagonalize a matrix A in order to compute A^n [7,3].

However the Spectral Theorem only asserts the *existence* of the spectral decomposition — actually *finding* it is a different task. Numerical mathematics offers a vast range of methods and software libraries for doing so in the *non-degenerate* case, that is, provided the λ_i are pairwise different. For the general case on the other hand, no satisfying algorithm has been found yet: all known methods suffer from numerical instabilities and convergence problems if eigenvalues coincide.

The present work puts these experiences onto the formal basis of computable analysis, which is the theory of real number computation as it has been developed by Turing, Banach and Mazur, Grzegorczyk, Lacombe, Pour-El and Richards, Kreitz and Weihrauch, Ko and many others [16,11,5,10,13,9,8]. We will follow Weihrauch's approach to computable analysis (the so-called *Type-2 Theory of Effectivity*) since it offers a very uniform frame for computations on real numbers, functions and subsets [17]. Using this theory we prove that the spectral resolution of non-degenerate normal matrices can be computed but in the general case it *cannot*. The reason for this intractability lies in the discontinuous behaviour of eigenvectors: even infinitely small perturbations of the input matrix (e.g. due to floating point approximations) may entirely change the eigenvectors. On the other hand, our following main result says that it suffices to know the cardinality of the spectrum $\sigma(A)$ in advance, in order to compute the spectral resolution of A (later on we will give a precise reformulation as Theorem 13).

Theorem 2 (Computable Spectral Theorem). *Given as input a normal matrix $A \in \mathbb{C}^{n\times n}$ and the cardinality of its spectrum $|\sigma(A)|$, we can compute an orthogonal basis $(x_1, ..., x_n)$ of $\mathbb{C}^n$ and complex numbers $\lambda_1, ..., \lambda_n \in \mathbb{C}$ such that $Ax_i = \lambda_i x_i$ for each $i = 1, ..., n$.*

Our work differs from that of Pour-El and Richards [13] in that we consider *uniform* computability. This means that we investigate computability of the spectral representation as (multi-valued) function of the input matrix A. In contrast, Pour-El and Richards look for sufficient and necessary conditions on A such that the eigenvalues and eigenvectors be objects which are computable as points, i.e., without considering the computability of their dependence on A itself. Furthermore the counter-examples of Pour-El and Richards rely on so

called *ad hoc* structures which have been defined for *infinite*-dimensional Hilbert spaces and thus do not apply to the finite-dimensional case we consider.

We close this section with a short survey on the organization of this paper. In the following section we recall some basic definitions from computable analysis and in Section 3 we will present relevant parts of our previous results from [18,2], which have been developed for the canonical Euclidean space $\mathbb{R}^n$. Here we will discuss the transfer of these results to the complex case $\mathbb{C}^n$. In Section 4 we will present basic results on eigenvalues and eigenvectors which are mainly based on the Computable Fundamental Theorem of Algebra. In Section 5 we discuss and prove our main result, the Computable Spectral Theorem.

2 Computable Analysis

In this section we briefly present some basic notions from computable analysis (based on the approach of Type-2 Theory of Effectivity) and some direct consequences of well-known facts. For a precise and comprehensive reference we refer the reader to [17]. Roughly speaking, a partial real number function $f :\subseteq \mathbb{R}^n \to \mathbb{R}$ is computable, if there exists a Turing machine which transfers each sequence $p \in \Sigma^\omega$ that represents some input $x \in \mathbb{R}^n$ into some sequence $F_M(p)$ which represents the output $f(x)$. Since the set of real numbers has continuum cardinality, real numbers can only be represented by infinite sequences $p \in \Sigma^\omega$ (over some finite alphabet Σ) and thus, such a Turing machine M has to compute infinitely long. But in the long run it transfers each input sequence p into an appropriate output sequence $F_M(p)$. It is reasonable to allow only one-way output tapes for infinite computations since otherwise the output after finite time would be useless (because it could possibly be replaced later by the machine). It is straightforward how this notion of computability can be generalized to other sets X with a corresponding *representation*, that is a surjective partial mapping $\delta :\subseteq \Sigma^\omega \to X$.

Definition 3 (Computable Functions). Let δ, δ' be representations of X, Y, respectively. A function $f :\subseteq X \to Y$ is called (δ, δ')–*computable*, if there exists some Turing machine M such that $\delta' F_M(p) = f\delta(p)$ for all $p \in \mathrm{dom}(f\delta)$.

Here, $F_M :\subseteq \Sigma^\omega \to \Sigma^\omega$ denotes the partial function, computed by the Turing machine M. It is straightforward how to generalize this definition to functions with several inputs and it can even be generalized to multi-valued operations $f :\subseteq X \rightrightarrows Y$, where $f(x)$ is a subset of Y instead of a single value. In this case we replace the condition in the definition above by $\delta' F_M(p) \in f\delta(p)$. We can also define the notion of (δ, δ')–*continuity* by replacing F_M by a continuous function $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ (w.r.t. the Cantor topology on Σ^ω).

Already in case of the real numbers it appears that the defined notion of computability sensitively relies on the chosen representation of the real numbers. The theory of *admissible* representations completely answers the question how to find "reasonable" representations of topological spaces (more precisely, of T_0-spaces with countable bases, see [17]). Let us just mention that for admissible

representations δ, δ' each (δ, δ')-computable function is necessarily continuous (w.r.t. the final topologies of δ, δ').

An example of an admissible representation of the real numbers is the so-called *Cauchy representation* $\rho :\subseteq \Sigma^\omega \to \mathbb{R}$, where roughly speaking, $\rho(p) = x$ if p is an (appropriately encoded) sequence of rational numbers $(q_i)_{i\in\mathbb{N}}$ which converges rapidly to x, i.e. $|q_k - x| \leq 2^{-k}$ for all k. By standard coding techniques this representation can easily be generalized to a representation of the n-dimensional Euclidean space $\rho^n :\subseteq \Sigma^\omega \to \mathbb{R}^n$ and to a representation of $m \times n$ matrices $\rho^{m\times n} :\subseteq \Sigma^\omega \to \mathbb{R}^{m\times n}$. From the computational point of view we can identify the set of complex numbers $\mathbb{C}$ with $\mathbb{R}^2$ and in this way we obtain canonically a representation $\rho^n_{\mathbb{C}} := \rho^{2n}$ of $\mathbb{C}^n$. Analogously, we consider $\rho^{m\times n}_{\mathbb{C}}$ as a representation of the set of $m \times n$–matrices $\mathbb{C}^{m\times n}$. A vector $x \in \mathbb{C}^n$ or a matrix $A \in \mathbb{C}^{m\times n}$ will be called *computable*, if it has a computable $\rho^n_{\mathbb{C}}$–, $\rho^{m\times n}_{\mathbb{C}}$–*name*, i.e. if there exists a computable $p \in \Sigma^\omega$ such that $x = \rho^n_{\mathbb{C}}(p)$ or $A = \rho^{m\times n}_{\mathbb{C}}(p)$, respectively. A function $f :\subseteq \mathbb{C}^n \to \mathbb{R}$ is called just *computable*, if it is $(\rho^n_{\mathbb{C}}, \rho)$-computable. Analogous notions are used over the real numbers.

If δ, δ' are admissible representations of T_0-spaces with countable bases X, Y, respectively, then there exists a canonical representation $[\delta, \delta'] :\subseteq \Sigma^\omega \to X \times Y$ of the product $X \times Y$ and a canonical representation $[\delta \to \delta'] :\subseteq \Sigma^\omega \to C(X, Y)$ of the space $C(X, Y)$ of the total continuous functions $f : X \to Y$. We just mention that these representations allow evaluation and type conversion (which correspond to an utm- and smn-Theorem). Evaluation means that the evaluation function $C(X, Y) \times X \to Y, (f, x) \mapsto f(x)$ is $([[\delta \to \delta'], \delta], \delta')$-computable and type conversion means that a function $f : Z \times X \to Y$ is $([\delta'', \delta], \delta')$-computable, if and only if the canonically associated function $f' : Z \to C(X, Y)$ with $f'(z)(x) := f(z, x)$ is $(\delta'', [\delta \to \delta'])$-computable. As a direct consequence we obtain that matrices $A \in \mathbb{C}^{m\times n}$ can effectively be identified with linear mappings $f \in \mathrm{Lin}(\mathbb{C}^n, \mathbb{C}^m)$, see Proposition 4.1 and 4.2 below. Especially, a matrix A is computable, if and only if the corresponding linear mapping is a computable function.

To express weaker computability properties, we will use two further representations $\rho_<, \rho_> :\subseteq \Sigma^\omega \to \mathbb{R}$. Roughly speaking, $\rho_<(p) = x$ if p is an (appropriately encoded) list of all rational numbers $q < x$. (Analogously, $\rho_>$ is defined with $q > x$.) It is known that a mapping $f :\subseteq X \to \mathbb{R}$ is (δ, ρ)-computable, if and only if it is $(\delta, \rho_<)$– and $(\delta, \rho_>)$-computable [17]. The $(\rho^n_{\mathbb{C}}, \rho_<)$–, $(\rho^n_{\mathbb{C}}, \rho_>)$-computable functions $f : \mathbb{C}^n \to \mathbb{R}$ are called *lower, upper semi-computable*, respectively. (Analogous notions are used for functions $f : \mathbb{R}^n \to \mathbb{R}$).

Occasionally, we will also use some standard representation $\nu_{\mathbb{N}}, \nu_{\mathbb{Q}}$ of the natural numbers $\mathbb{N} = \{0, 1, 2, ...\}$ and the rational numbers $\mathbb{Q}$, respectively.

Moreover, we will also need a representation for the space $\mathcal{L}^n_{\mathbb{C}}$ of linear subspaces $V \subseteq \mathbb{C}^n$. Since all linear subspaces are non-empty closed spaces, we can use well-known representations of the hyperspace $\mathcal{A}^n_{\mathbb{C}}$ of all closed non-empty subsets $A \subseteq \mathbb{C}^n$ (cf. [1,17]). One way to represent such spaces is via the distance function $d_A : \mathbb{C}^n \to \mathbb{R}$, defined by $d_A(x) := \inf_{a\in A} d(x, a)$, where $d : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{R}$ denotes the canonical metric of $\mathbb{C}^n$, defined by $d(x, y) := |x - y|$. Altogether, we

define three representations $\psi^n_{\mathbb{C}}, \psi^n_{\mathbb{C}<}, \psi^n_{\mathbb{C}>} :\subseteq \Sigma^\omega \to \mathcal{A}^n_{\mathbb{C}}$. We let $\psi^n_{\mathbb{C}}(p) = A$, if and only if $[\rho^n_{\mathbb{C}} \to \rho](p) = d_A$. In other words, p encodes a set A w.r.t. $\psi^n_{\mathbb{C}}$, if it encodes the distance function d_A w.r.t. $[\rho^n_{\mathbb{C}} \to \rho]$. Analogously, let $\psi^n_{\mathbb{C}<}(p) = A$, if and only if $[\rho^n_{\mathbb{C}} \to \rho_>](p) = d_A$ and let $\psi^n_{\mathbb{C}>}(p) = A$, if and only if $[\rho^n_{\mathbb{C}} \to \rho_<](p) = d_A$. One can prove that $\psi^n_{\mathbb{C}<}$ encodes "positive" information about the set A (all open rational balls $B(q,r) := \{x \in \mathbb{C}^n : d(x,q) < r\}$ which intersect A can be enumerated), and $\psi^n_{\mathbb{C}>}$ encodes "negative" information about A (all closed rational balls $\overline{B}(q,r)$ which do not intersect A can be enumerated). The final topology induced by $\psi^n_{\mathbb{C}}$ on $\mathcal{A}^n_{\mathbb{C}}$ is the *Fell topology*. It is a known fact that a mapping $f :\subseteq X \to \mathcal{A}^n_{\mathbb{C}}$ is $(\delta, \psi^n_{\mathbb{C}})$-computable, if and only if it is $(\delta, \psi^n_{\mathbb{C}<})$– and $(\delta, \psi^n_{\mathbb{C}>})$-computable [17]. We mention that

1. the operation $(f, A) \mapsto f^{-1}(A) \subseteq \mathbb{C}^n$ is $\big([[\rho^n_{\mathbb{C}} \to \rho^m_{\mathbb{C}}], \psi^m_{\mathbb{C}>}], \psi^n_{\mathbb{C}>}\big)$-computable,
2. the operation $(f, B) \mapsto \overline{f(B)} \subseteq \mathbb{C}^m$ is $\big([[\rho^n_{\mathbb{C}} \to \rho^m_{\mathbb{C}}], \psi^n_{\mathbb{C}<}], \psi^m_{\mathbb{C}<}\big)$-computable.

From these properties one can deduce some computability properties of kernel and image, see Proposition 4.3 and 4.4 below.

A closed set $A \subseteq \mathbb{C}^n$ is called *recursive*, if it is empty or if there is a computable $p \in \Sigma^\omega$ such that $A = \psi^n_{\mathbb{C}}(p)$. Thus, the non-empty recursive subsets $A \subseteq \mathbb{C}^n$ are exactly those with computable distance function $d_A : \mathbb{C}^n \to \mathbb{R}$. We will apply all defined notions analogously to closed subsets of $\mathbb{R}^n$ and we will denote the corresponding representations simply without index "$\mathbb{C}$" (cf. [18,2]).

3 Computable Linear Algebra

In our previous papers [18,2] we have investigated some basic computability properties of linear algebra on the canonical Euclidean vector space $\mathbb{R}^n$. The purpose of this section is to transfer these results to the complex vector space $\mathbb{C}^n$. We will argue that all proofs can be transfered in a one-to-one manner without any essentially new aspects.

In the following we assume that the unitary vector space $\mathbb{C}^n$ is endowed with the canonical inner product, defined by $x \cdot y := \sum_{i=1}^n x_i \overline{y_i}$ for vectors $x = (x_1, ..., x_n), y = (y_1, ..., y_n) \in \mathbb{C}^n$. Here $\overline{z} := a - ib$ denotes the *conjugate* of the complex number $z = a + ib \in \mathbb{C}$. Correspondingly, we will use the norm $|x| := \sqrt{x \cdot x} = \sqrt{\sum_{i=1}^n x_i \overline{x_i}}$. The following proposition reformulates results from [18] for the complex case.

Proposition 4. *Consider the following canonical mappings from linear algebra:*

1. $\mathrm{Lin}(\mathbb{C}^n, \mathbb{C}^m) \to \mathbb{C}^{m\times n}$ *is* $\big([\rho^n_{\mathbb{C}} \to \rho^m_{\mathbb{C}}], \rho^{m\times n}_{\mathbb{C}}\big)$*-computable,*
2. $\mathbb{C}^{m\times n} \to \mathrm{Lin}(\mathbb{C}^n, \mathbb{C}^m)$ *is* $\big(\rho^{m\times n}_{\mathbb{C}}, [\rho^n_{\mathbb{C}} \to \rho^m_{\mathbb{C}}]\big)$*-computable,*
3. $\ker : \mathbb{C}^{m\times n} \to \mathcal{A}^n_{\mathbb{C}}$ *is* $\big(\rho^{m\times n}_{\mathbb{C}}, \psi^n_{\mathbb{C}>}\big)$*-computable, but neither* $(\rho^{m\times n}_{\mathbb{C}}, \psi^n_{\mathbb{C}<})$*-computable, nor –continuous,*
4. $\mathrm{span} : \mathbb{C}^{m\times n} \to \mathcal{A}^m_{\mathbb{C}}$ *is* $(\rho^{m\times n}_{\mathbb{C}}, \psi^m_{\mathbb{C}<})$*-computable, but neither* $(\rho^{m\times n}_{\mathbb{C}}, \psi^m_{\mathbb{C}>})$*-computable, nor –continuous,*

5. $\det : \mathbb{C}^{n\times n} \to \mathbb{R}$ *is* $\big(\rho_{\mathbb{C}}^{n\times n}, \rho\big)$*-computable,*
6. $\mathrm{rank} : \mathbb{C}^{m\times n} \to \mathbb{R}$ *is* $\big(\rho_{\mathbb{C}}^{m\times n}, \rho_<\big)$*-computable, but neither* $\big(\rho_{\mathbb{C}}^{m\times n}, \rho_>\big)$*-computable, nor –continuous,*
7. $\dim :\subseteq \mathcal{A}_{\mathbb{C}}^n \to \mathbb{R}$ *is* $\big(\psi_{\mathbb{C}<}^n, \rho_<\big)$*– and* $\big(\psi_{\mathbb{C}>}^n, \rho_>\big)$*-computable.*

All the proofs given in [18] can be transfered to the complex case one-to-one. This is mainly due to the fact that topologically and computationally we can identify $\mathbb{C}$ with $\mathbb{R}^2$. Wherever the rational numbers $\mathbb{Q}$ have been used, we substitute $\mathbb{Q}[i]$ for $\mathbb{Q}$ (and we replace S^{n-1} by the border of the unit ball of $\mathbb{C}^n$ in the proof of Lemma 8 in [18]).

While Proposition 4.3 shows that the solution space of a (homogeneous) linear equation $Ax = 0$ does not depend continuously on the matrix A (w.r.t. $\psi_{\mathbb{C}}^n$), we have proved in [2] that we can compute the solution space if its dimension is known in advance. We reformulate this result for the complex case.

Theorem 5 (Computable Solvability of Linear Equations). *The function*

$$S :\subseteq \mathbb{C}^{m\times n} \times \mathbb{R} \to \mathcal{A}_{\mathbb{C}}^n, \quad (A, d) \mapsto \ker(A)$$

with $\mathrm{dom}(S) := \{(A, d) : d = \dim \ker(A)\}$ *is* $\big([\rho_{\mathbb{C}}^{m\times n}, \rho], \psi_{\mathbb{C}}^n\big)$*-computable.*

Again the proof can be transfered directly. Whenever the Euclidean norm and the canonical inner product of $\mathbb{R}^n$ has been used in [2], we replace them by the corresponding operations of $\mathbb{C}^n$ (the Cauchy-Schwarz inequality holds analogously in this case). Also the Gram-Schmidt orthogonalization process can be used in the complex case analogously. Finally, we mention a corollary, which shows that we can also effectively find a basis, in case that the dimension of the space is known.

Corollary 6. *The multi-valued mapping*

$$B :\subseteq \mathcal{A}_{\mathbb{C}}^n \times \mathbb{R} \rightrightarrows \mathcal{A}_{\mathbb{C}}^n, (V, d) \mapsto \big\{\{b_1, ..., b_d\} \subseteq \mathbb{C}^n : (b_1, ..., b_d) \textit{ is a basis of } V\big\}$$

with $\mathrm{dom}(B) := \{(V, d) : d = \dim(V)\}$ *is* $([\psi_{\mathbb{C}}^n, \rho], \psi_{\mathbb{C}}^n)$*-computable.*

4 Eigenvalues, Eigenvectors, and Eigenspaces

Recall that the *adjoint* of a complex $n\times n$-matrix $A = \big(a_{ij}\big) \in \mathbb{C}^{n\times n}$ is defined by $A^* = \big(\overline{a}_{ji}\big)$. A number $\lambda \in \mathbb{C}$ with $Az = \lambda z$ for some non-zero vector $z \in \mathbb{C}^n$ is called *eigenvalue* of A and z a corresponding *eigenvector*. The set of eigenvalues, called *spectrum* $\sigma(A)$, is precisely the set of zeros of the *characteristic polynomial* $\det(zI - A) \in \mathbb{C}[z]$, where $I \in \mathbb{C}^{n\times n}$ denotes the $n \times n$-unit matrix. The set of eigenvectors to $\lambda \in \sigma(A)$ — extended by $0 \in \mathbb{C}^n$ — forms a vector space, the *eigenspace* $\ker(\lambda I - A)$. Our first observation states that given a matrix A, we can compute its characteristic polynomial.

Proposition 7. *Given a complex $n \times n$-matrix $A \in \mathbb{C}^{n\times n}$, we can compute its characteristic polynomial*

$$z^n + \sum_{i=0}^{n-1} a_i z^i := \det(zI - A).$$

More precisely, the mapping $\chi : \mathbb{C}^{n\times n} \to \mathbb{C}^n, A \mapsto (a_0, ..., a_{n-1})$ is $(\rho_{\mathbb{C}}^{n\times n}, \rho_{\mathbb{C}}^n)$-computable.

The proof is a straightforward calculation using *Leibniz' formula*

$$\det(B) = \sum_{\sigma\in\mathcal{S}_n} \operatorname{sign}(\sigma) \prod_{i=1}^{n} b_{\sigma(i)i},$$

for matrices $B = (b_{ij}) \in \mathbb{C}^{n\times n}$, where $\mathcal{S}_n$ denotes the set of all permutations $\sigma : \{1, ..., n\} \to \{1, ..., n\}$.

In the next step a computable version of the Fundamental Theorem of Algebra, due to Specker [15], can help us to compute the eigenvalues of a given matrix. It says that, on input of a polynomial $p \in \mathbb{C}[z]$ of degree $n \in \mathbb{N}$, its n complex zeroes can be computed.

Theorem 8 (Computable Fundamental Theorem of Algebra). *Consider the unique normed polynomial having exactly the zeros $\lambda_1, ..., \lambda_n \in \mathbb{C}$ (including multiplicities):*

$$z^n + \sum_{i=0}^{n-1} a_i z^i := \prod_{i=1}^{n} (z - \lambda_i)$$

The mapping $\mathbb{C}^n \ni (\lambda_1, ..., \lambda_n) \mapsto (a_0, ..., a_{n-1}) \in \mathbb{C}^n$ is surjective and has a multi-valued $(\rho_{\mathbb{C}}^n, \rho_{\mathbb{C}}^n)$-computable right inverse $Z : \mathbb{C}^n \rightrightarrows \mathbb{C}^n$.

See Exercise 11 in Section 6.3 of [17] for a sketch of the proof. A combination of this theorem with the previous Proposition 7 and the fact that $\mathbb{C}^n \to \mathcal{A}_{\mathbb{C}}^1, (z_0, ..., z_{n-1}) \mapsto \{z_0, ..., z_{n-1}\}$ is $(\rho_{\mathbb{C}}^n, \psi_{\mathbb{C}}^1)$-computable, directly yields the computability of the spectrum mapping.

Corollary 9. *The map $\sigma : \mathbb{C}^{n\times n} \to \mathcal{A}_{\mathbb{C}}^1, \quad A \mapsto \sigma(A)$ is $(\rho_{\mathbb{C}}^{n\times n}, \psi_{\mathbb{C}}^1)$-computable.*

Computing eigenspaces is slightly more difficult. As we have seen in Theorem 5 we can compute the kernel of a matrix, provided that we know its dimension in advance. Thus, we directly obtain the following corollary of Theorem 5 (applied to $\lambda I - A$) on computing eigenspaces.

Corollary 10. *Given a matrix $A \in \mathbb{C}^{n\times n}$ together with some eigenvalue $\lambda \in \mathbb{C}$ and $d = \dim\ker(\lambda I - A)$, we can compute the eigenspace $\ker(\lambda I - A)$. More precisely,*

$$E :\subseteq \mathbb{C}^{n\times n} \times \mathbb{C} \times \mathbb{R} \to \mathcal{A}_{\mathbb{C}}^n, \quad (A, \lambda, d) \mapsto \ker(\lambda I - A),$$

with $\operatorname{dom}(E) := \{(A, \lambda, d) : \lambda$ is eigenvalue of A and $d = \dim\ker(\lambda I - A)\}$, is $([\rho_{\mathbb{C}}^{n\times n}, \rho_{\mathbb{C}}, \rho], \psi_{\mathbb{C}}^n)$-computable

Considering a single computable matrix $A \in \mathbb{C}^{n\times n}$ together with some eigenvalue $\lambda \in \mathbb{C}$ (which necessarily is computable too by Corollary 9), the discrete dimension is always available as a further (computable) input; thus, we obtain the following corollary.

Corollary 11. *Each computable real or complex $n \times n$-matrix A has a recursive spectrum $\sigma(A)$, especially, all eigenvalues are computable. Moreover, the corresponding eigenspaces are all recursive and each eigenvalue admits a computable eigenvector.*

5 The Spectral Theorem

Recall that a complex $n \times n$-matrix $A \in \mathbb{C}^{n\times n}$ is called *self-adjoint*, if $A = A^*$ and *normal*, if $AA^* = A^*A$. For normal matrices A, the dimension $d \in \mathbb{N}$ of the eigenspace $\ker(\lambda I - A)$ of some eigenvalue $\lambda \in \mathbb{C}$ equals the algebraic *multiplicity* of the zero λ of the characteristic polynomial $\det(zI - A)$, i.e., the i-th derivative $d^i/dz^i \det(zI - A) \in \mathbb{C}[z]$ vanishes at $z = \lambda$ for $i = 0, ..., d-1$ but not for $i = d$.

Now our goal is to prove a computational version of the classical Spectral Theorem 1. Unfortunately, it turns out that the spectral resolution cannot be computed directly from a given self-adjoint matrix $A \in \mathbb{C}^{n\times n}$. The chief snag is that, although the spectrum $\sigma(A)$ can be computed from A, its cardinality $|\sigma(A)|$ does not continuously depend on A. Even worser, the following proposition shows that the eigenvectors of real symmetric 2×2-matrices A do not depend continuously on A. For the proof, we borrow an example of Rellich [14] which can also be found in [6].

Proposition 12. *There exists no $(\rho^{2\times 2}, \rho^{2\times 2})$–continuous multi-valued function*

$$F :\subseteq \mathbb{R}^{2\times 2} \rightrightarrows \mathbb{R}^{2\times 2}$$

such that each $(x, y) \in F(A)$ is an orthogonal basis of $\mathbb{R}^2$ consisting of eigenvectors of A, and $A \in \mathrm{dom}(F)$ whenever $A \in \mathbb{R}^{2\times 2}$ is symmetric.

Proof. First of all, it suffices to prove the statement for ortho*normal* bases instead of ortho*gonal* bases, since each orthogonal basis (x, y) can continuously be normalized to $(x/|x|, y/|y|)$. Let us assume, that there exists a $(\rho^{2\times 2}, \rho^{2\times 2})$–continuous multi-valued function $F :\subseteq \mathbb{R}^{2\times 2} \rightrightarrows \mathbb{R}^{2\times 2}$ which solves the problem for orthonormal bases. Now consider the continuous function $A : \mathbb{R} \to \mathbb{R}^{2\times 2}$, defined by

$$A(\varepsilon) := \exp(-1/\varepsilon^2) \begin{pmatrix} \cos(2/\varepsilon) & \sin(2/\varepsilon) \\ \sin(2/\varepsilon) & -\cos(2/\varepsilon) \end{pmatrix}, \quad A(0) := \begin{pmatrix} 0\,0 \\ 0\,0 \end{pmatrix} .$$

In case $\varepsilon > 0$ the eigenvalues of $A(\varepsilon)$ are $\exp(-1/\varepsilon^2)$ and $-\exp(-1/\varepsilon^2)$; the corresponding orthonormal basis of eigenvectors of A is uniquely determined up to order and orientation and consists of the vectors

$$x(\varepsilon) := \begin{pmatrix} \cos(1/\varepsilon) \\ \sin(1/\varepsilon) \end{pmatrix}, \quad y(\varepsilon) := \begin{pmatrix} \sin(1/\varepsilon) \\ -\cos(1/\varepsilon) \end{pmatrix}.$$

Thus, the eigenspaces and eigenvectors of $A(\varepsilon)$ rotate faster and faster if ε tends to 0. By assumption $F \circ A : \mathbb{R} \rightrightarrows \mathbb{R}^{2\times 2}$ is $(\rho, \rho^{2\times 2})$–continuous. Let us fix some $z := (x, y) \in FA(0)$, which could be any orthonormal basis of $\mathbb{R}^2$. Then there are arbitrarily small $\varepsilon > 0$ such that $z(\varepsilon) := (x(\varepsilon), y(\varepsilon))$ is far away from z, no matter how $x(\varepsilon)$ and $y(\varepsilon)$ are oriented or ordered, e.g. $|z - z(\varepsilon)| > \frac{1}{2}$. Thus, $F \circ A$ cannot be $(\rho, \rho^{2\times 2})$–continuous. Contradiction! □

Any substantial information about the degree of degeneracy however does enable computability of the spectral representation. Especially the additional knowledge of the cardinality of the spectrum suffices, as our following result shows.

Theorem 13 (Computable Spectral Theorem). *There exists a multi-valued* $([\rho_{\mathbb{C}}^{n\times n}, \rho], [\rho_{\mathbb{C}}^{n\times n}, \rho_{\mathbb{C}}^{n}])$*-computable function*

$$R :\subseteq \mathbb{C}^{n\times n} \times \mathbb{R} \rightrightarrows \mathbb{C}^{n\times n} \times \mathbb{C}^n$$

with $\mathrm{dom}(R) := \{(A, s) : A \text{ normal and } s = |\sigma(A)|\}$ *such that for any normal* $A \in \mathbb{C}^{n\times n}$ *with* $s = |\sigma(A)|$ *and* $((x_1, ..., x_n), (\lambda_1, ..., \lambda_n)) \in R(A, s)$ *we obtain*

$$Ax_i = \lambda_i x_i$$

and $(x_1, ..., x_n)$ *is an orthogonal basis of* $\mathbb{C}^n$.

That means that our algorithm has to know in advance the number of pairwise different eigenvalues. Indeed from this, the multiplicities of all eigenvalues can be deduced. This follows from the following technical lemma (where, for technical simplicity, multiplicities occur as complex numbers).

Lemma 14. *The function* $T :\subseteq \mathbb{C}^{n+1} \to \mathcal{A}_{\mathbb{C}}^2$, *with*

$$T(\lambda_1, ..., \lambda_n, s) := \{(\lambda, d) : \lambda = \lambda_i \text{ for precisely } d \text{ values of } i = 1, ..., n\}$$

with $\mathrm{dom}(T) := \big\{(\lambda_1, ..., \lambda_n, s) : s = |\{\lambda_1, ..., \lambda_n\}|\big\}$ *is* $(\rho_{\mathbb{C}}^{n+1}, \psi_{\mathbb{C}}^2)$*-computable.*

Proof. We sketch the proof which is a straightforward exercise. Let us assume that $(\lambda_1, ..., \lambda_n) \in \mathbb{C}^n$ and $s = |\{\lambda_1, ..., \lambda_n\}|$ are given w.r.t. $\rho_{\mathbb{C}}$. Since the set $\{(x, y) : x \neq y\}$ is an r.e. open subsets of $\mathbb{C}^2$, we can compare all pairs (λ_i, λ_j) with $i \neq j$ until we can distinguish exactly s pairwise different "clusters" of values λ_j. At this stage we know that all d values λ in a cluster have to coincide and we can produce a name of the set of all pairs (λ, d) w.r.t. $\psi_{\mathbb{C}}^2$. □

This idea actually enables us to complete the proof of the computable Spectral Theorem.

Proof (of Theorem 13). Let us assume that a normal matrix $A \in \mathbb{C}^{n\times n}$ is given w.r.t. $\rho_{\mathbb{C}}^{n\times n}$ and $s = |\sigma(A)|$ is given w.r.t. ρ. By applying Proposition 7 and the computable version of the Fundamental Theorem of Algebra 8, we can compute a tuple $(\lambda_1, ..., \lambda_n) \in \mathbb{C}^n$ of eigenvalues. By the proof of the previous lemma

we can compute the set $T(\lambda_1, ..., \lambda_n, s)$ of all pairs (λ, d) of eigenvalues λ of A together with their multiplicities d, i.e. $d = \dim \ker(\lambda I - A)$. For each of these pairs (λ, d) we apply Corollary 10 in order to compute the corresponding eigenspace $E(A, \lambda, d) = \ker(\lambda I - A)$ w.r.t. $\psi_{\mathbb{C}}^n$. By applying Corollary 6 we can compute a basis $(y_1, ..., y_d) \in \mathbb{C}^n$ of the eigenspace $E(A, \lambda, d)$. Now we employ the Gram-Schmidt orthogonalization process to compute an orthogonal basis $(z_1, ..., z_d) \in \mathbb{C}^n$ by

$$z_1 := y_1, \quad z_{j+1} := y_{j+1} - \sum_{i=1}^{j} \frac{y_{j+1} \cdot z_i}{|z_i|^2} z_i$$

for all $j = 1, ..., d-1$. Since eigenspaces for different eigenvalues are orthogonal to each other we can simply put the s bases of all s eigenspaces $E(A, \lambda, d)$ together and we obtain an orthogonal basis $(x_1, ..., x_n)$ of $\mathbb{C}^n$. Finally, we mention that we can arrange the tuple of eigenvalues $(\lambda'_1, ..., \lambda'_n)$ in a corresponding order such that $A\lambda'_i = \lambda'_i x_i$ for all $i = 1, ..., n$. □

We immediately obtain the following weaker computable version of the Spectral Theorem.

Corollary 15. *If $A \in \mathbb{C}^{n \times n}$ is a computable and normal matrix, then there are computable vectors $x_1, ..., x_n \in \mathbb{C}^n$ and computable numbers $\lambda_1, ..., \lambda_n \in \mathbb{C}$ such that $(x_1, ..., x_n)$ is an orthogonal basis of $\mathbb{C}^n$ and $Ax_i = \lambda_i x_i$ for each $i = 1, ..., n$.*

6 Conclusion

In this paper we have continued our project to investigate computability properties in linear algebra with rigorous methods from computable analysis. Using previous results from [18,2] we have proved an effective version of the finite-dimensional Spectral Theorem. One could continue this project in several different directions: on the one hand, there are other normal forms in linear algebra which have not yet been studied from the point of view of computability. On the other hand, one could ask for more general cases, such as unitary vector spaces endowed with an inner product different from the canonical one, or infinite-dimensional Hilbert spaces. Moreover, it would be a fascinating project to establish connections with applications of the Spectral Theorem.

References

1. Vasco Brattka and Klaus Weihrauch. Computability on subsets of Euclidean space I: Closed and compact subsets. *Theoretical Computer Science*, 219:65–93, 1999.
2. Vasco Brattka and Martin Ziegler. Computability of linear equations. unpublished, 2000.
3. Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic complexity theory*, vol. 315 of *Grundl. der math. Wissenschaften*. Springer, Berlin, 1997.

4. Fan R.K. Chung. *Spectral Graph Theory*. CBMS Regional Conference Series, vol. 92, American Mathematical Society, Providence 1997.
5. Andrzej Grzegorczyk. On the definitions of computable real continuous functions. *Fundamenta Mathematicae*, 44:61–71, 1957.
6. Tosio Kato. *Perturbation Theory for Linear Operators*, vol. 132 of *Grundlehren der mathematischen Wissenschaften*. Springer, Berlin, 2. edition, 1976.
7. Donald E. Knuth. *The art of computer programming*. Computer Science and Information Processing. Addison-Wesley, Reading, 1981. Volume 2, Seminumerical algorithms.
8. Ker-I Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1991.
9. Christoph Kreitz and Klaus Weihrauch. A unified approach to constructive and recursive analysis. In M.M. Richter, E. Börger, W. Oberschelp et al. eds., *Computation and Proof Theory*, vol. 1104 of *LNM*, 259–278, Springer, Berlin, 1984.
10. Daniel Lacombe. Les ensembles récursivement ouverts ou fermés, et leurs applications à l'Analyse récursive. *Compt. Rend. Acad. des Sci. Paris*, 246:28–31, 1958.
11. S. Mazur. *Computable Analysis*, vol. 33. Razprawy Matematyczne, Warsaw, 1963.
12. Bojan Mohar and Svatopluk Poljak. Eigenvalues in Combinatorial Optimization. In R. A. Brualdi, S. Friedland and V. Klee, (eds.) *Combinatorial and Graph-Theoretic Problems in Linear Algebra*, pages 107–151, The IMA Volumes in Mathematics and Its Applications, vol. 50, Springer, Berlin, 1993.
13. Marian B. Pour-El and J. Ian Richards. *Computability in Analysis and Physics*. Perspectives in Mathematical Logic. Springer, Berlin, 1989.
14. Franz Rellich. Störungstheorie der Spektralzerlegung I., Analytische Störung der isolierten Punkteigenwerte eines beschränkten Operators, in *Math. Ann.* 113:600-619, 1937.
15. Ernst Specker. The fundamental theorem of algebra in recursive analysis. In B. Dejon and P. Henrici, editors, *Constructive Aspects of the Fundamental Theorem of Algebra*, pages 321–329, London, 1969. Wiley-Interscience.
16. Alan M. Turing. On computable numbers, with an application to the "Entscheidungsproblem". *Proc. of the London Math. Soc.*, 42(2):230–265, 1936.
17. Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.
18. Martin Ziegler and Vasco Brattka. Computing the dimension of linear subspaces. In V. Hlavác, K.G. Jeffery and J. Wiedermann, editors, *SOFSEM'2000: Theory and Practice of Informatics*, vol. 1963 of LNCS, 450–458, Springer, Berlin, 2000.

Exact Real Arithmetic Systems: Results of Competition

Jens Blanck

University of Wales Swansea, Singleton Park, Swansea, SA2 8PP, UK

Abstract. A competition between systems for doing exact real number computations was held in September 2000. We present the results obtained and give a short evaluation of the different approaches used.

Introduction

Computable Analysis is flourishing theoretically, and during the last decade there have also been some practical advances. Systems for doing exact real number computations have appeared. Although still in their infancy, some of these systems are capable of non-trivial practical computations. The systems use very different approaches and it was therefore decided to have a competition between existing systems for exact real number computations as part of the 4th workshop of Computability and Complexity in Analysis at Swansea (CCA 2000). The competition was organised by David Lester.

The aims of the competitions were: To establish the state of art in exact real number computations; and to be a forum where competitors may debate ideas and implementation techniques from different systems. This was a first attempt at staging such a competition. The problems set were simple calculations that would check basic capabilities of the systems.

The contestants are listed in Table 1.

Table 1. The contestants.

Competitor	**System**	**Acronym**
David Lester (Manchester)	Manchester Arithmetic Package	MAP
Norbert Müller (Trier)	iterative Real RAM	iRRAM
Marko Krznaric (Imperial College)		IC
Tom Kelsey (St Andrew)		Kelsey
Paul Zimmermann (INRIA Lorraine)	MPFR[1]	MPFR

The person running the system for the competition has been entered as the competitor. The system name has also been listed in case a specific name was known. The entries will henceforth be referred to by the acronyms listed in the

[1] Multiple Precision Floating-point Reliable library.

J. Blanck, V. Brattka, P. Hertling (Eds.): CCA 2000, LNCS 2064, pp. 389–393, 2001.

table. The competitor has been the main developer of their systems except for IC and MPFR, which are joint projects. See [1,2,3,4,5]. MPFR was a remote entry. All contestants except Kelsey were running their systems on AMD 800 MHz machines with 256 Mbyte RAM. Since Maple was not available on these computers, Kelsey was forced to run his system on a 233 MHz Pentium Laptop.

There were two sets of problems. One set containing the precirculated problems testing the minimum capability of the systems. The other set was announced to the contestants shortly before the competition.

The competition was supervised by a committee consisting of Günter Hotz, Daniel Richardson, Dieter Spreen, and the author.

Results

The results of the first set of problems is in Table 2. The tabulated timings are in seconds. Note that the table is given for different accuracy depending on implementation. The MAP, iRRAM, and MPFR systems computed the results to 10000 decimal places. The IC system is for 1000 decimal places, and Kelsey's system is for 100 decimal places. Also note that MPFR did not restart the system after each example as the rest of the contestants did; this explains why MPFR gets much shorter times for cosine than for sine. IC withdrew from the competition after two problems.

Table 2. Level 0 problems.

To Calculate	MAP	iRRAM	MPFR	IC	IC[2]	Kelsey
$\sqrt{\pi}$	.05	.13	.75	.78	.13	
$\log \pi$	16.73	1.10	1.70	30.00	.39	89.22
$\sin e$	2.03	1.30	4.00		.30	111.26
$\cos e$	2.10	1.30	.09		.28	126.90
$\sin(\sin(\sin 1))$	5.79	.99	4.63		2.39	106.72
$\cos(\cos(\cos 1))$	5.88	.99	4.65		1.94	111.83
e^{e^e}	2.12	1.60	1.05		1.90	
$\log(1+\log(1+\log(1+\pi)))$	32.42	1.37	.73		6.25	102.49
$\log(1+\log(1+\log(1+e)))$	78.60	1.53	.95		8.96	123.29
$\log(1+\log(1+\log(1+\log(1+\pi))))$	112.73	1.90	1.29		10.19	141.36
$\log(1+\log(1+\log(1+\log(1+e))))$	179.87	2.03	1.51		11.67	136.19
$\sin 10^{50}$	2.02	1.48	49.28		90.33	139.52
$\cos 10^{50}$	2.03	1.47	.08		90.81	
e^{1000}	1.15	2.26	.39		1.11	
$\arctan 10^{50}$	.17	.20			10.79	

The results of the second set of problems is in Table 3. The contestants were given one hour to run these problems. The second set was not attempted by all

[2] These results for IC were supplied at a later date and are for 100 decimal places and run on a 233 MHz Pentium laptop.

contestants. MPFR did not get this set of problems. Kelsey did not compute any of them. Again, the accuracy was 10000 decimal places except for IC giving 1000 decimal places.

The logistic map problem was to compute 1000 iterations of the logistic map

$$x_{n+1} = \frac{15}{4}(x_n - x_n^2)\,,$$

with starting value $x_0 = \frac{1}{\pi}$, to ten decimal places.

Table 3. CCA 2000 problems.

To Calculate	MAP	iRRAM	IC	IC[3]
$e^{\pi\sqrt{163}}$	1.07	2.43	26.72	1.40
$\sqrt[3]{\sqrt[5]{\frac{32}{5}} - \sqrt[5]{\frac{27}{5}} - \frac{1+\sqrt[5]{3}-\sqrt[5]{9}}{\sqrt[5]{25}}}$	199.53	3.74		11.47
$\sin\left(\frac{3\log 640320}{\sqrt{163}}\right)$	15.87	1.31		3.28
Logistic map	>500.00	0.07		4h

Correctness

A comprehensive checking of correctness has not been performed, although correctness of the computed results is obviously very important. Ideally, each system should be accompanied by a proof of the correctness of the system. Such proofs exists for IC and an earlier version of MAP.

As a quick check of correctness, MAP and iRRAM results have been checked for some of the problems and agree up to 10000 decimal places. As these two systems are implemented in different ways this is a good indication that they actually compute the correct result. This consensus approach to validate results is of course only an indication of correctness, not a proof.

Evaluation

The three systems, iRRAM, MAP, and MPFR, have in common that they compute on *fast converging dyadic Cauchy sequences*. The iRRAM uses an iterative bottom-up approach. It starts with a predefined precision of the inputs. The subexpressions are then evaluated bottom-up and only the guaranteed precision after the operation is forwarded to the next subexpression. If the precision of the

[3] These results for IC were supplied at a later date and are for 100 decimal places and run on a 233 MHz Pentium laptop.

result is not sufficient, it recomputes the whole expression with increased precision. The method used in MPFR is also bottom-up but it uses floating point calculations with *directed* rounding. MPFR rounds the expression both up and down and if the resulting interval is too wide, it increases the input precision. In contrast, MAP uses a top-down approach. The precision needed for each subexpression is derived from the expression tree. The MAP does not need to do any recomputations.

The IC uses *linear fractional transformations* (LFT). In this approach, trees of LFTs are constructed, which are then normalised to a normal form. Some operations are derived from algorithms used on continued fractions, since these can be encoded into LFTs. This approach has the advantage that the operations have the best possible convergence speed. However, it also means that IC suffers from the rapidly growing coefficients that come with long continued fractions.

Kelsey uses symbolic computations in Maple. It is clearly slower than the other approaches. This can only partly be explained by the inferior hardware that it was run on.

That IC did not perform very well must be taken as indication that the LFT approach is not as efficient as the Cauchy sequence approach. This probably also applies to approaches using pure continued fractions.

There is not much to separate the Cauchy sequence based approaches for many of the expressions tested. However, the consistency of the iRRAM made it the clear choice as winner in this competition.

Winner. The winner of the competition was Norbert Müller's iRRAM.

Criticism

The value of the results presented herein should not be overestimated. It is of course always hard to make such a competition fair and conclusive. However, there are some issues that really should be addressed. There was no attempt to deduce the performance of the systems as a function of the size of the problem. By problem size one can consider both the precision required and the size of the expression. (The logistic map example can be seen as a really big expression if the iterations are unfolded.) One should also be careful with the choice of problems. It may be that one approach is particularly well-suited to certain problems.

Another shortcoming of the Swansea 2000 competition was that only calculator style problems were included. For the next competition it would be nice to see some other types of problem, e.g., integration, differential equations, linear algebra (the iRRAM already does matrix operations). This would require the competitors to implement other data types apart from the reals, e.g., the space of continuous functions.

References

1. A. Edalat and J.P. Potts, A New Representation for Exact Real Numbers, *Electronical Notes in Theoretical Computer Science* 6 (1997),
2. P. Gowland and D. Lester, A Survey of Exact Arithmetic Implementations, in: J. Blanck, V. Brattka and P. Hertling (editors), *Computability and Complexity in Analysis*, Lecture Notes in Computer Science, Springer, this volume.
3. MPFR group, The MPFR library, `http://www.mpfr.org/`.
4. N. Müller, The iRRAM: Exact Arithmetic in C++, in: J. Blanck, V. Brattka and P Hertling (editors), *Computability and Complexity in Analysis*, Lecture Notes in Computer Science, Springer, this volume.
5. P.J. Potts, Exact Real Arithmetic Using Möbius Transformations. PhD Thesis, University of London, Imperial College, 1998.

Author Index