On function spaces and polynomial-time computability

Akitoshi Kawamura and Arno Pauly

1 Department of Computer Science, University of Tokyo
Japan
kawamura@is.s.u-tokyo.ac.jp
2 Clare College, University of Cambridge
United Kingdom
Arno.Pauly@cl.cam.ac.uk

Abstract. In the context of second-order polynomial-time computability, we prove that there is no general function space construction. We proceed to identify restrictions on the domain or the codomain that do provide a function space with polynomial-time function evaluation containing all polynomial-time computable functions of that type. As side results we show that a polynomial-time counterpart to admissibility of a representation is not a suitable criterion for natural representations, and that the Weihrauch degrees embed into the polynomial-time Weihrauch degrees.

Keywords: cartesian closed, computational complexity, higher order, computable analysis, admissible representation, Weihrauch reducibility

1 Introduction

Computable analysis (e.g. [30]) deals with computability questions for operators from analysis such as integration, differentiation, Fourier transformation, etc.. In general, the actual computation is envisioned to be performed on infinite sequences over some finite or countable alphabet, this model is then lifted to the spaces of interest by means of representations. Thus, an adequate choice of representations for the various relevant spaces is the crucial foundation for any investigation in computable analysis.

At first, the search for good representations proceeded in a very ad-hoc fashion, exemplified by Turing’s original definition of a computable real number as one with computable decimal expansion [27] and later correction to one with a computable sequence of nested rational intervals collapsing to the number [28].

The development of more systematic techniques to identify good representations had two interlocked main components: One, the identification of admissibility as the central criterion whenever the space in question already carries

3 This choice of a representation, which is indeed a correct one, is credited to Brouwer by Turing.
a natural topology by Kreitz and Weihrauch [18] and later Schröder [26]. Two, the observation that one can form function spaces in the category of represented spaces (e.g. [29], [2]). Using the ideas of synthetic topology [8], this suffices to obtain good representations of spaces just from their basic structure (demonstrated in [21]).

While computable analysis has obtained a plethora of results, for a long time the aspect of computational complexity has largely been confined to restricted settings (e.g. [31]) or non-uniform results (e.g. [17]). This was due to the absence of a sufficiently general theory of second-order polynomial-time computability – a gap which was filled by Cook and the first author in [7]. This theory can be considered as a refinement of the computability theory. In particular, this means that for doing complexity theory, one has to choose well-behaved representations for polynomial-time computation out of the equivalence classes w.r.t. computable translations.

Various results on individual operators have been obtained in this new framework [14, 15, 16, 24], leaving the field at a very similar state as the early investigation of computability in analysis: While some indicators are available what good choices of representations are, an overall theory of representations for computational complexity is missing. Our goal here is to provide the first steps towards such a theory by investigating the role of admissibility and the presence of function spaces for polynomial-time computability.

2 Background on second-order polynomial-time computability

We will use (a certain class of) string functions to encode the objects of interest. We fix some alphabet $\Sigma$. We say that a (total) function $\varphi : \Sigma^* \to \Sigma^*$ is regular if it preserves relative lengths of strings in the sense that $|\varphi(u)| \leq |\varphi(v)|$ whenever $|u| \leq |v|$. We write $\text{Reg}$ for the set of all regular functions. We restrict attention to regular functions (rather than using all functions from $\Sigma^*$ to $\Sigma^*$) to keep the notion of their size (to be defined shortly) simple.

We use an oracle Turing machine (henceforth just “machine”) to convert regular functions to regular functions (Figure 1).

Definition 1. A machine $M$ computes a partial function $F : \subseteq \text{Reg} \to \text{Reg}$ if for any $\varphi \in \text{dom}F$, the machine $M$ on oracle $\varphi$ and any string $u$ outputs $F(\varphi)(u)$ and halts.

Remark 1. For computability, this is equivalent to the model where a Turing machine converts infinite strings to infinite strings. For the discussion of polynomial-time computability, however, we really need to use strings functions in order to encode information efficiently and to measure the input size, as we will see below.

4 The concept of structure here goes beyond topologies, as witnessed e.g. by the treatment of hyperspaces of measurable sets and functions in [22].
Regular functions map strings of equal length to strings of equal length. Therefore it makes sense to define the size $|\varphi| : \mathbb{N} \to \mathbb{N}$ of a regular function $\varphi$ to be the (non-decreasing) function $|\varphi|(|u|) = |\varphi(u)|$. We will use $\text{Mon}$ to denote the strictly monotone functions from $\mathbb{N}$ to $\mathbb{N}$. For technical reasons, we will tacitly restrict ourselves to those regular functions $\varphi$ with $|\varphi| \in \text{Mon}$, this does not impede generality.

We will make use of a polynomial-time computable pairing function $\langle , \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$, which we want to satisfy $|\langle u, v \rangle| = |u| \times |v|$. This is then lifted to a pairing function on $\text{Reg}$ via $\langle \varphi, \psi \rangle(u) = \langle \varphi(u), \psi(u) \rangle$. Finally, we also lift $\langle , \rangle$ to $\text{Mon}$ (although not as a tupling function) by $\langle f, g \rangle(n) = f(n) \times g(n)$.

Now we want to define what it means for a machine to run in polynomial time. Since $|\varphi|$ is a function, we begin by defining polynomials in a function, following the idea of Kapron and Cook [13]. Second-order polynomials (in type-1 variable $L$ and type-0 variable $n$) are defined inductively as follows: a positive integer is a second-order polynomial; the variable $n$ is also a second-order polynomial; if $P$ and $Q$ are second-order polynomials, then so are $P + Q$, $P \cdot Q$ and $L(P)$. An example is

$$L(L(n \cdot n)) + L(L(n) \cdot L(n)) + L(n) + 4.$$  \hfill (1)

A second-order polynomial $P$ specifies a function, which we also denote by $P$, that takes functions $L \in \text{Mon}$ to another function $P(L) \in \text{Mon}$ in the obvious way. For example, if $P$ is the above second-order polynomial (1) and $L(n) = n^2$, then $P(L)$ is given by

$$P(L)(n) = ((n \cdot n)^2)^2 + (n^2 \cdot n^2)^2 + n^2 + 4 = 2 \cdot n^8 + n^2 + 4.$$  \hfill (2)

As in this example, $P(L)$ is a (usual first-order) polynomial if $L$ is.

**Definition 2.** A machine $M$ runs in polynomial time if there is a second-order polynomial $P$ such that, given any $\varphi \in \text{Reg}$ as oracle and any $u \in \Sigma^*$ as input, $M$ halts within $P(|\varphi|(|u|))$ steps.

---

5. Given some $\varphi \in \text{Reg}$, let $\varphi'$ be defined by $\varphi'(v) = v \varphi(v)$. Then $\varphi' : \text{Reg} \to \text{Reg}$ is polynomial-time computable, and has a polynomial-time computable inverse. Moreover, $|\varphi'| \in \text{Mon}$ for all $\varphi \in \text{Reg}$.

6. While this choice is a bit wasteful, it is useful for technical reasons, and ultimately does not matter for polynomial-time computability.
This defines the class of (polynomial-time) computable functions from Reg to Reg. We can suitably define some other complexity classes related to nondeterminism or space complexity, as well as the notions of reduction and hardness [7].

A representation δ of a set X is formally a partial function from Reg to X that is surjective—that is, for each x ∈ X, there is at least one ϕ ∈ Reg with δ(ϕ) = x. We say that ϕ is a δ-name of x. A represented space is a pair (X, δX) of a set X together with a representation δX of it. For a function f : X → Y between represented spaces X, Y and F :⊆ Reg → Reg, we call F a realizer of f (notation F ⊇ f), iff δY(F(ϕ)) = f(δX(ϕ))) for all p ∈ dom(fδX). A map between represented spaces is called (polynomial-time) computable, iff it has a (polynomial-time) computable realizer.

3 Some properties of second-order polynomials

We will establish some properties of second-order polynomials as the foundation for our further investigations. For this, we first introduce the notion of the second-order degree of a second-order polynomial by deg(1) = 0, deg(n) = 0, deg(P + Q) = deg(P × Q) = max{deg(P), deg(Q)} for P, Q ≠ 0 and deg(L(P)) = deg(P) + 1. Just as the degree of an ordinary polynomial uniquely determines its O-notation equivalence class, we find a similar result for the second-order degree and second-order polynomials. The role of the monomials x^n are taken by the second-order polynomials P_n defined via P_0(p, k) = k and P_{n+1}(p, k) = p(P_n(p)).

Lemma 1. For any second-order polynomial Q there are q ∈ Mon and n ∈ N such that Q(p, k) ≤ P_{max{deg(Q),1}}((p, q), (k + 1)^n) for all p ∈ Mon, k ∈ N.

Proof. By induction over the structure of Q.

[Case: deg(Q) = 0 ] In this case Q is an ordinary polynomial q, and we find Q(p, k) = q(k) = P_1((p, q), (k + 1)).

[Case: Q = Q_1 + Q_2, deg(Q_1), deg(Q_2) > 0 ] Let q_1, q_2, n_1, n_2 be suitable choices for the component polynomials. Then we have Q(p, k) = Q_1(p, k) + Q_2(p, k) ≤ P_{deg(Q_1)}((p, q_1), (k + 1)^{n_1}) + P_{deg(Q_2)}((p, q_2), (k + 1)^{n_2}) ≤ (p, q_2)P_{deg(Q_1)}((p, q_1 + q_2), (k + 1)^{max{n_1, n_2}}) + (p, q_2)P_{deg(Q_2)}((p, q_1 + q_2), (k + 1)^{max{n_1, n_2}}) ≤ P_{deg(Q)}((p, q_1 + q_2), (k + 1)^{max{n_1, n_2}}), so q_1 + q_2 and max{n_1, n_2} work as witnesses for Q.

[Case: Q = Q_1 × Q_2, deg(Q_1), deg(Q_2) > 0 ] Let q_1, q_2, n_1, n_2 be suitable choices for the component polynomials. Then we have Q(p, k) = Q_1(p, k) × Q_2(p, k) ≤ P_{deg(Q_1)}((p, q_1), (k + 1)^{n_1}) × P_{deg(Q_2)}((p, q_2), (k + 1)^{n_2}) ≤ (p, q_1)P_{deg(Q_1)}((p, q_1 × q_2), (k + 1)^{max{n_1, n_2}}) × (p, q_2)P_{deg(Q_2)}((p, q_1 × q_2), (k + 1)^{max{n_1, n_2}}) ≤ P_{deg(Q)}((p^2, q_1 × q_2), (k + 1)^{max{n_1, n_2}}) ≤ P_{deg(Q)}((p, q_1 × q_2), (k + 1)^{2max{n_1, n_2}}), so q_1 × q_2 and 2 max{n_1, n_2} work as witnesses for Q.

[Case: Q = L(Q_1), deg(Q_1) = 0 ] As pointed out above, Q_1 is some ordinary polynomial q_1. In particular, there is some n ∈ N such that q_1(k) ≤ (k + 1)^n. We now find Q(p, k) = p(q_1(k)) ≤ p((k + 1)^n) = P_1(p, (k + 1)^n) ≤ P_1((p, q), (k + 1)^n), so q = 1 and n witness the claim.
[Case: \( Q = L(Q_1) \), \( \deg(Q_1) > 0 \)] If \( Q_1(p,k) \leq P_{\deg(Q_1)}((p,q),(k+1)^n) \), then \( Q(p,k) \leq p(P_{\deg(Q_1)}((p,q),(k+1)^n)) \leq (p,q)(P_{\deg(Q_1)}((p,q),(k+1)^n)) = P_{\deg(Q)}((p,q),(k+1)^n) \), so the same witnesses working for \( Q_1 \) also work for \( Q \).

**Lemma 2.** For no \( q \in \text{Mon} \), \( n, m \in \mathbb{N} \) we have \( P_{n+1}(p,k) \leq P_n((p,q),(k+1)^n) \) for all \( p \in \text{Mon} \), \( k \in \mathbb{N} \).

**Proof.** Assume the contrary, and let \( q, n, m \) witness the claim. We proceed to construct a \( p \) such that \( P_{n+1}(p,1) = p(P_n(p,1)) > P_n((p,q),2^m) \), thus obtaining a contradiction. If \( n = 0 \), then choosing \( p(k) = (k+1)^m + 1 \) suffices. Next we define a \( p \) in stages which will work for all remaining \( n > 0 \). For \( 1 \leq i \leq 2^m \), let \( p(i) = 2^m + 1 \). This implies \( P_2(p,1) = p(2^m + 1) \) and \( P_2((p,q),2^m) = (2^m + 1,q(2^m)) \). So choosing \( p(2^m + 1) = (2^m + 1,q(2^m)) + 1 \) works. More generally, for \( 2^m + 1 \leq i \leq (2^m + 1,q(2^m)) \) we shall set \( p(i) = (2^m + 1,q(2^m)) + 1 \). Then \( P_3(p,1) = p((2^m + 1,q(2^m)) + 1) \) and \( P_3((p,q),2^m) = ((2^m + 1,q(2^m)) + 1,q((2^m + 1,q(2^m)) + 1) \), so choosing \( p((2^m + 1,q(2^m)) + 1) = ((2^m + 1,q(2^m)) + 1,q((2^m + 1,q(2^m)) + 1) \) provides the contradiction for \( n = 2 \). We can continue in the same fashion indefinitely, thus obtaining the remaining cases.

### 4 Failure of cartesian closure

We shall show that the category of \( \text{Reg} \)-represented spaces and polynomial-time computable functions is not cartesian closed. For this we define the functions \( \Phi_n : \text{Reg} \rightarrow \text{Reg} \) via \( \Phi_0 = \text{id}_{\text{Reg}} \) and \( \Phi_{n+1}(\varphi)(w) = \varphi(\Phi_n)(w) \). Then computing \( \Phi_n(\varphi,w) \) takes time \( O(P_n(|\varphi|,|w|)) \), as already the length of the output provides a lower bound.

**Theorem 1.** Let the second-order polynomial \( P \) witness polynomial-time computability of the function \( F : \subseteq \text{Reg} \times \text{Reg} \rightarrow \text{Reg} \). For no \( \psi \in \text{Reg} \) we may have \( F(\psi,\varphi) = \Phi_{\deg(p)+1}(\varphi) \) for all \( \varphi \in \text{Reg} \).

**Proof.** This is a direct consequence of Lemma 2.

**Corollary 1.** There cannot be an exponential in the category of \( \text{Reg} \)-represented spaces and polynomial-time computable functions.

**Proof.** Any realizer of the evaluation operation would violate Theorem 1.

### 5 Clocked Type-Two machines

Despite the negative result above, we can identify spaces of functions with some of the desired properties of exponentials. The required technical tool is a type-two version of clocked Turing machines. We pick a Universal Turing Machine (UTM) \( M \) which simulates efficiently, this means that on input \( n, \varphi, w \) the time \( M \) needs to compute the output of the \( n \)th Oracle Turing machine on input \( w \).
with oracle \( \varphi \) is bounded by a quadratic polynomial in \( n \) and the time \( T \) needed by the \( n \)th Turing machine itself to compute the output on \( w \) with oracle \( \varphi \). Then \( M \) is extended by a clock evaluating the standard second-order polynomial \( P_m \) on \( |(n, \varphi)|, |w| \) for fixed \( m \) and some \( l \in \mathbb{N} \) encoded as \((x \mapsto x') \in \text{Mon} \) and aborts the computation of \( M \) once the runtime exceeds the value of \( P_m \). Denote the resulting machine with \( M^{T= P_m} \). The runtime of \( M^{T= P_m} \) can be bounded by \( KP_{m+1}^2 + K \) for some constant \( K \in \mathbb{N} \). In particular we find that the second-order degree of the runtime of \( M^{T= P_m} \) is \( m + 1 \).

**Theorem 2.** For any partial function \( f : \subseteq \text{Reg} \rightarrow \text{Reg} \) computable in polynomial time \( P \) with \( \deg(P) \leq m \) there are some \( \psi \in \text{Reg} \), \( n, l \in \mathbb{N} \) such that for any \( \varphi \in \text{dom}(f) \) we find \( f(\varphi) = M^{T= P_m}(\langle n, (\varphi, \psi), x' \rangle) \).

*Proof.* Pick some \( \psi \in \text{Reg} \), \( l \in \mathbb{N} \) such that \( |\psi| \in \text{Mon} \), \( l \) satisfy the criterion in Lemma 1, and some \( n \) that is an index of the machine computing \( f \) in time \( P \). The former guarantees that the clock of \( M^{T= P_m} \) does not abort the computation on valid input; its underlying universal Turing machine then works as intended.

Based on the preceding theorem, we see that rather than a single function space, we obtain a family of function spaces indexed by a natural number corresponding to the second-order degree. Given two \( \text{Reg} \)-represented spaces \( X, Y \) we define the function space \( \mathcal{C}^{T= P_m}(X, Y) \) by letting \( \langle n, \psi, x' \rangle \in \text{Reg} \) be a name for \( f : X \rightarrow Y \) if \( \varphi \mapsto M^{T= P_m}(\langle n, (\varphi, \psi), x' \rangle) \) is a realizator of \( f \). This definition just enforces that \( \text{Eval} : \mathcal{C}^{T= P_m}(X, Y) \times X \rightarrow Y \) is computable with polynomial time bound \( KP_{m+1}^2 + K \).

We can then reformulate Theorem 1 as \( \mathcal{C}^{T= P_m}(\text{Reg, Reg}) \subseteq \mathcal{C}^{T= P_{m+1}}(\text{Reg, Reg}) \) and Theorem 2 as \( f \in \mathcal{C}^{T= P_m}(X, Y) \) for any \( f : X \rightarrow Y \) computable in a polynomial time-bound of \( \deg \leq m \). We can easily obtain an even stronger version of the latter by adapting the proof:

**Corollary 2.** For a function \( f : X \rightarrow Y \) the following properties are equivalent:

1. \( f \) is computable in polynomial time \( P \) with \( \deg(P) \leq m \).
2. \( f \in \mathcal{C}^{T= P_m}(X, Y) \) has a polynomial time computable name.

### 6 Effectively polynomial-bounded spaces

Our next goal is to investigate restrictions we can employ on \( X \) (and later on \( Y \)) in order to force the collapse of the time hierarchy \( \mathcal{C}^{T= P_m}(X, Y) \subseteq \mathcal{C}^{T= P_{m+1}}(X, Y) \).

---

7 A straight-forward adaption of the classical result by Hennie and Stearns [11] provides the existence of such a universal machine.

8 More generally, we could use an arbitrary time-constructible function in place of \( P_m \). That \( P_m \) actually is time-constructible is witnessed by \( \Phi_m \).
Definition 3. We call $X$ effectively polynomially bounded (epb), if it admits a $\text{Reg}$-representation $\delta_X$ such that there is a polynomial time computable function $\chi : \text{dom}(\delta_X) \to \mathbb{N}$ and a monotone bivariate polynomial $Q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ s.t.:

$$\forall \varphi \in \text{dom}(\delta_X) \forall i \in \mathbb{N} \ |\varphi|(i) \leq Q(i, \chi(\varphi))$$

Note that we could alternatively use the estimate $|\varphi(i)| \leq c|\varphi|(c)^mQ(i)$ for some constant $c \in \mathbb{N}$ and a univariate monotone polynomial $Q$.

Theorem 3. Let $X$ be epb. Then for any $m \geq 2$ we find $C^{T=P_2}(X, Y) \cong C^{T=P_m}(X, Y)$ where $\cong$ denotes polytime isomorphic.

Proof. It is sufficient to show only the direction $\subseteq: C^{T=P_m}(X, Y) \to C^{T=P_2}(X, Y)$. Let $M$ be the UTM used in the definition of $C^{T=P_m}(X, Y)$, let $M'$ behave with the oracle $\langle \varphi, \langle \psi, \psi' \rangle \rangle$ in exactly the same way as $M$ does with $\langle \varphi, \psi \rangle$, and then finally use $M'$ to define $C^{T=P_2}(X, Y)$.

The assumption that $X$ is epb allows us to estimate:

$$P_m(|\langle \varphi, \psi \rangle|, k) = |\langle \varphi, \psi \rangle|(|P_{m-1}(|\langle \varphi, \psi \rangle|), k)\rangle\leq c|\varphi|(c)^mQ(|P_{m-1}(|\langle \varphi, \psi \rangle|), k)\rangle\leq (cQ^c \times \psi)|(|P_{m-1}(|\langle \varphi, \psi \rangle|), (k + 1)^c)\rangle \leq (cQ^c \times \psi)|(|P_{m-2}(|\langle \varphi, \psi \rangle|), (k + 1)^c)\rangle \leq (cQ^c \times \psi)|(|P_{m-1}(|\langle \varphi, \psi \rangle|), (k + 1)^c)\rangle \leq P_2(|\langle \varphi, \psi \rangle|, cQ^c \times \psi)|(|P_{m-1}(|\langle \varphi, \psi \rangle|), (k + 1)^c)\rangle$$

Now given $\psi$, we can compute some $\psi'$ with $|\langle \varphi, \psi \rangle|, (cQ^c \times \psi)|^{(m)} \leq |\langle \varphi, \psi \rangle|, \psi'$ in polynomial time (note that $Q$, $c$ and $m$ are all constants here). The $l$ in the original name is replaced by $lc^m$.

It is worthwhile pointing out that the function spaces for computability do not only contain the computable functions as elements, but comprise exactly the continuous functions as discussed very well in [1], yielding a structure dubbed category extension in [21,20]. This is due to the fact that the (partial) functions $f : \subseteq \mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}$ arising as sections of computable (partial) functions $F : \subseteq \mathbb{N}^\mathbb{N} \times \mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}$ are just the continuous functions.

In a similar way, we shall investigate which functions appear in a space $C^{T=P_2}(X, Y)$ for epb $X$. It turns out that (a modification of) uniform continuity plays a central role. A connection between run-time bounds and the modulus of continuity was also found for multivalued functions in [23].

Definition 4. We call a partial function $f : \subseteq \text{Reg} \to \text{Reg}$ polytime-locally uniformly continuous, if there is a polynomial-time computable function $\chi : \subseteq \text{Reg} \to \mathbb{N}$, such that $\text{dom}(f) \subseteq \text{dom}(\chi)$ and any $f|_{\chi^{-1}(n)}$ is uniformly continuous.

Theorem 4. Let $X \subseteq \text{Reg}$ be epb. Then for $f : X \to \text{Reg}$ the following are equivalent:
1. \( f \) is polytime-locally uniformly continuous
2. \( f \in C^T = P^2(X, \text{Reg}) \)

Proof. 1. \( \Rightarrow \) 2. Given Theorem 3 and Corollary 2, it suffices to show that such an \( f \) is polynomial-time computable relative to some oracle \( \psi \). We start by some \( \Lambda \in \text{Mon} \) such that \( i \mapsto \Lambda(\langle n, i \rangle) \) is a modulus of continuity of \( f|_{\chi^{-1}({\{n\}})} \). Then \( f(\varphi)(u) \) depends only on values \( \varphi(w) \) with \( |w| \leq \Lambda(\langle \chi(\varphi), |u| \rangle) \), and we may encode this dependency in some table \( \psi \). In order to write the query to \( \psi \), the machine needs time \( 2^\Lambda(\langle \chi(\varphi), |u| \rangle) \). By providing \( \langle 2^\Lambda, \psi \rangle \) as an oracle, this time is made available.

2. \( \Rightarrow \) 1. By continuing the estimate from the proof in Theorem 3 we obtain an upper bound for the evaluation of \( f \) given its \( C^T = P^2(X, \text{Reg}) \)-name depending only on \( \psi \), \( l \) and \( \chi(\varphi) \), but beyond that not on \( \varphi \). In particular, for fixed \( \chi(\psi) \), there is a bound \( \lambda : \mathbb{N} \rightarrow \mathbb{N} \) such that to compute \( f(\varphi)(w) \), \( \varphi \) is only queried on inputs \( v \) with \( |v| \leq \lambda(|w|) \) – but this is uniform continuity.

Note that the same argument used for 1. \( \Rightarrow \) 2. in the preceding proof also establishes that \( C^T = P^2(\mathbb{R}, \mathbb{R}) \) contains all the continuous functions, where \( \mathbb{R} \) is represented as suggested in [7].

7 Padding and polytime admissibility

In this section we shall explore two distinct but similar arguments based on using padding-like concepts on the codomain of a function in order to make time bounds irrelevant. This technique both reveals polynomial-time admissibility as a far too restrictive concept (as opposed to computable admissibility) and allows us to draw some conclusions about degree structures.

We define a \( \text{Reg} \)-representation \( \pi \) of Cantor space via \( \pi(\varphi)(i) = \varphi(0^n)(i) \) if \( |\varphi(0^n)| \geq i \). Note that \( \varphi \in \text{dom}(\pi) \) requires \( |\varphi| \) to be unbounded. Now any Cantor-representation \( \delta \) can be turned into a \( \text{Reg} \)-representation by composing with \( \pi \), and by this we obtain a strong correspondence between computability and polynomial-time computability.

Proposition 1. A function \( f : X \rightarrow (Y, \delta_Y) \) is computable if and only if \( f : X \rightarrow (Y, \delta_Y \circ \pi) \) is polynomial-time computable.

Proof. The map \( \pi \) is computable, this provides one direction. For the other direction, note that a computation providing a result in \( (Y, \delta_Y \circ \pi) \) can safely be delayed as long as required to stay within any given time bound.

Weihrauch reducibility (e.g. [6,11,12]) is a computable many-one reduction between multivalued functions that serves as the basis of a metamathematical research programme. Likewise, a reduction that could be called polynomial-time Weihrauch reducibility has been investigated by some authors (e.g. [3,7]). In [19,20] abstract principles were demonstrated that provide a very similar degree structure for both. Let \( (\mathcal{M}, \oplus, +, \times) \) and \( (\mathcal{P}, \mathcal{S}, +, \times) \) be the corresponding degree structures for Weihrauch reducibility and polynomial-time Weihrauch reducibility. We then find:
Corollary 3. \((\mathbb{N}, \oplus, +, \times)\) embeds as a substructure into \((\mathbb{N}, \oplus, +, \times)\).

The characterization of admissibility that admits a translation into the setting of computational complexity is due to Schröder [25] (see also [21]). Given the Sierpiński space \(S\) and the function space \(C(-, -)\), we find that there is a canonic map \(\kappa_S : X \to C(C(X, S), S)\) with \(\kappa_S(x)(f) = f(x)\). A space \(X\) is called computably admissible, if \(\kappa_S\) admits a computable partial inverse.

The space \(S\) has the underlying set \(\{\top, \bot\}\), and the representation \(\delta_S : \text{Reg} \to S\) defined by \(\delta_S(\varphi) = \top\) iff \(\exists w . \varphi(w) = 1\). By the same argument as Proposition 1, any computable function into \(S\) is computable in polynomial time – in fact, even linear time suffices. Thus, just as in Section 6 we can use the space \(C^{T=1}_1(X, S)\) as a function space and subsequently obtain a definition of polynomial-time admissibility by calling \(X\) polynomial-time admissible iff the (polynomial-time computable) map \(\kappa_S : X \to C^{T=1}_1(X, S)\) has a polynomial-time computable partial inverse. However, this notion is of limited use:

Proposition 2. If \(x \in X\) for polynomial-time admissible \(X\) has a computable name, then it has a polynomial-time computable name.

Proof. As polynomial-time computable functions preserve polynomial-time computable names, this follows from a function \(f : C^{T=1}_1(X, S) \to S\) being polynomial-time computable iff it is computable together with Corollary 2.

Note that this implies that all the representations suggested in [7] fail to be polynomial-time admissible, despite appearing to be very reasonable choices.

8 Conclusions

The trusted techniques developed for the theory of represented spaces and computable functions are insufficient to fully comprehend polynomial-time computability. Function spaces are not always available, and even where they are, they might differ from the familiar one of the continuous functions. Instead, some form of uniform continuity will be appear as the central notion.

What can be used as a guiding principle for the choice of representations is the epb property. If compatible with other criteria, choosing a representation that makes a space epb also makes function spaces well-behaved. For example, separable metric spaces are traditionally represented by encoding points by fast converging sequences of basic elements. For computability theory it does not matter what fast means – for complexity theory it does. A sensible choice could be: As fast as possible while retaining the epb property. Whether this already determines a representation up to polynomial-time equivalence is open, though.

\footnote{This observation was also made by Férée and Hoyrup [10] (see also [9]), and they suggested to use higher-order functionals on the machine level to retain spaces of continuous function with efficient evaluation. However, as shown by Schröder, this would change the notion of computability, too.}
References

1. Bauer, A.: Realizability as the connection between computable and constructive mathematics. Tutorial at CCA 2004 (notes)
2. Bauer, A.: A relationship between equilogical spaces and type two effectivity. Mathematical Logic Quarterly 48(1), 1–15 (2002)
3. Beame, P., Cook, S., Edmonds, J., Impagliazzo, R., Pitassi, T.: The relative complexity of NP search problems. Journal of Computer and System Science 57, 3–19 (1998)
4. Brattka, V., de Brecht, M., Pauly, A.: Closed choice and a uniform low basis theorem. Annals of Pure and Applied Logic 163(8), 968–1008 (2012)
5. Brattka, V., Gherardi, G.: Effective choice and boundedness principles in computable analysis. Bulletin of Symbolic Logic 1, 73 – 117 (2011).
6. Brattka, V., Gherardi, G.: Weihrauch degrees, omniscience principles and weak computability. Journal of Symbolic Logic 76, 143 – 176 (2011).
7. Cook, S., Kawamura, A.: Complexity theory for operators in analysis. ACM Transactions on Computation Theory 4(2) (2012)
8. Escardó, M.: Synthetic topology of datatypes and classical spaces. Electronic Notes in Theoretical Computer Science 87 (2004)
9. Férey, H., Gomaa, W., Hoyrup, M.: Analytical properties of resource-bounded real functionals. preprint (2013), http://www.loria.fr/~hoyrup/functionals.pdf
10. Férey, H., Hoyrup, M.: Higher-order complexity in analysis. CCA 2013
11. Hennie, F.C., Stearns, R.E.: Two-tape simulation of multitape turing machines. J. ACM 13(4), 533–546 (1966)
12. Higuchi, K., Pauly, A.: The degree-structure of Weihrauch-reducibility. Logical Methods in Computer Science 9(2) (2013)
13. Kapron, B.M., Cook, S.A.: A new characterization of type-2 feasibility. SIAM Journal on Computing 25(1), 117–132 (1996)
14. Kawamura, A.: Lipschitz continuous ordinary differential equations are polynomial-space complete. Computational Complexity 19(2), 305–332 (2010)
15. Kawamura, A., Müller, N., Rösnick, C., Ziegler, M.: Parameterized Uniform Complexity in Numerics: from Smooth to Analytic, from NP-hard to Polytime. arXiv 1211.4974 (2012)
16. Kawamura, A., Ota, H., Rösnick, C., Ziegler, M.: Computational complexity of smooth differential equations. In: Proc. of MFSC 2012. LNCS, vol. 7464, pp. 578–589. Springer (2012)
17. Ko, K.I.: Polynomial-time computability in analysis. Birkhäuser (1991)
18. Kreitz, C., Weihrauch, K.: Theory of representations. Theoretical Computer Science 38, 35–53 (1985)
19. Pauly, A.: Many-one reductions between search problems. arXiv 1102.3151 (2011), http://arxiv.org/abs/1102.3151
20. Pauly, A.: Multi-valued functions in computability theory. In: Cooper, S., Dawar, A., Löwe, B. (eds.) How the World Computes, Lecture Notes in Computer Science, vol. 7318, pp. 571–580. Springer (2012), http://dx.doi.org/10.1007/978-3-642-30870-3{(_)}357
21. Pauly, A.: A new introduction to the theory of represented spaces. http://arxiv.org/abs/1204.3763 (2012)
22. Pauly, A., de Brecht, M.: Towards synthetic descriptive set theory: An instantiation with represented spaces. arXiv 1307.1850
23. Pauly, A., Ziegler, M.: Relative computability and uniform continuity of relations. Journal of Logic and Analysis 5 (2013)
24. Rösnick, C.: Closed sets and operators thereon. CCA 2013
25. Schröder, M.: Admissible Representations for Continuous Computations. Ph.D. thesis, FernUniversität Hagen (2002)
26. Schröder, M.: Extended admissibility. Theoretical Computer Science 284(2), 519–538 (2002)
27. Turing, A.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the LMS 2(42), 230–265 (1936)
28. Turing, A.: On computable numbers, with an application to the Entscheidungsproblem: Corrections. Proceedings of the LMS 2(43), 544–546 (1937)
29. Weihrauch, K.: Type 2 recursion theory. Theoretical Computer Science 38, 17–33 (1985)
30. Weihrauch, K.: Computable Analysis. Springer-Verlag (2000)
31. Weihrauch, K.: Computational complexity on computable metric spaces. Mathematical Logic Quarterly 49(1), 3 – 21 (2003)

Acknowledgements

The second author is grateful to Anuj Dawar, Carsten Rösnick and Martin Ziegler for valuable discussions pertaining to the topic of the paper. This work is supported in part by the Japanese Grant-in-Aid for Scientific Research (Kakenhi) 24106002, and the Marie Curie International Research Staff Exchange Scheme Computable Analysis, PIRSES-GA-2011- 294962.